ARUNAI ENGINEERING COLLEGE

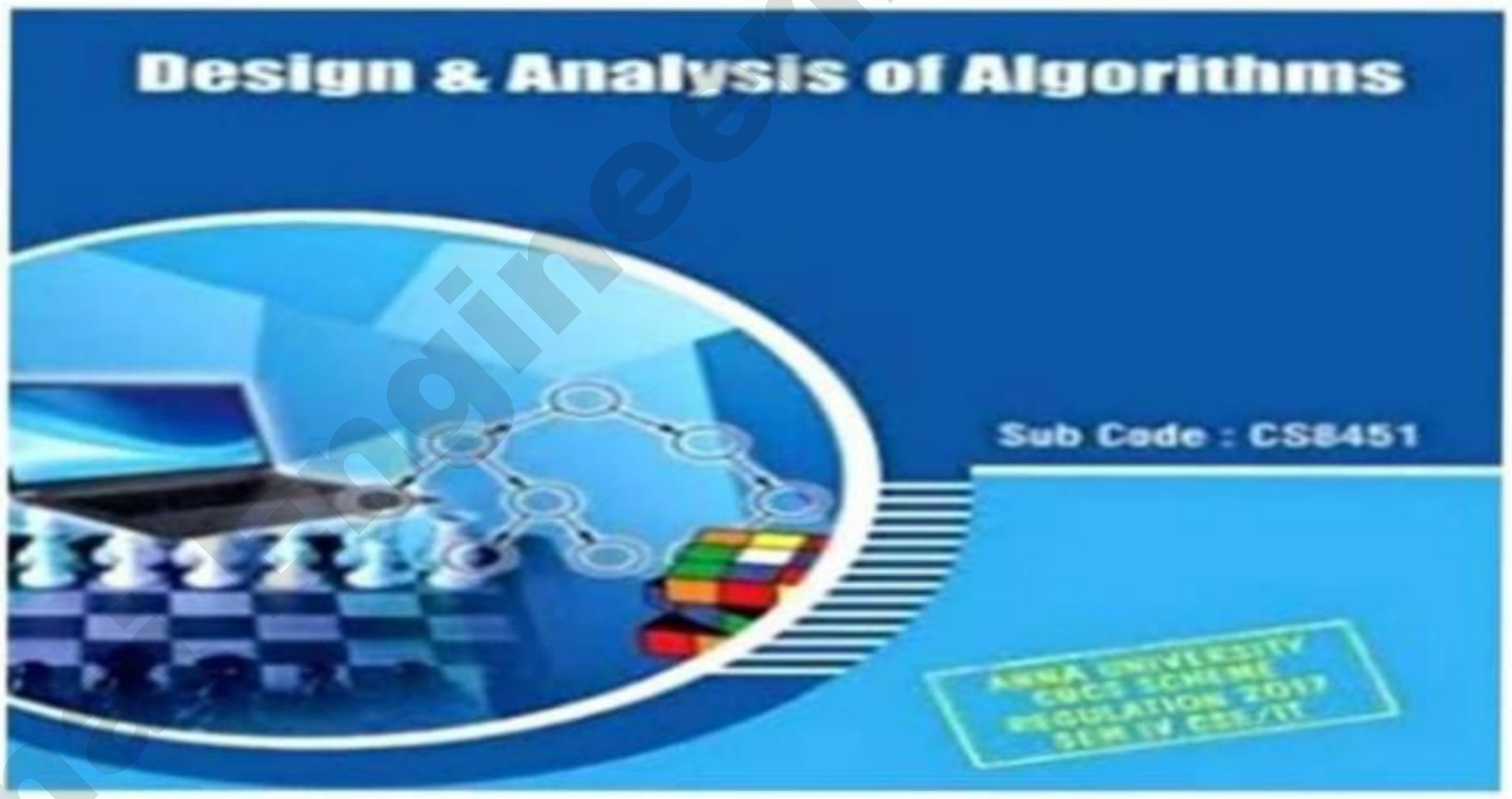
(Affiliated to Anna University)
Velu Nagar, Tiruvannamalai —606603
Phone: 04175-237419/236799/237739
www.arunai.org

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BACHELOR OF ENGINEERING

Second Year

Fourth Semester



CS8451-Design & Analysis Of Algorithm

Lecture By – Mrs. Karthika .D , AP/CSE

Scanned by TapScanner

CS8451 DESIGN AND ANALYSIS OF ALGORITHMS L T P C

3 0 0 3

UNIT I INTRODUCTION 9

Notion of an Algorithm – Fundamentals of Algorithmic Problem Solving – Important Problem Types – Fundamentals of the Analysis of Algorithmic Efficiency –Asymptotic Notations and their properties. Analysis Framework – Empirical analysis – Mathematical analysis for Recursive and Non-recursive algorithms – Visualization

UNIT II BRUTE FORCE AND DIVIDE-AND-CONQUER 9

Brute Force – Computing an – String Matching – Closest-Pair and Convex-Hull Problems – Exhaustive Search – Travelling Salesman Problem – Knapsack Problem – Assignment problem. Divide and Conquer Methodology – Binary Search – Merge sort – Quick sort – Heap Sort – Multiplication of Large Integers – Closest-Pair and Convex – Hull Problems.

UNIT III DYNAMIC PROGRAMMING AND GREEDY TECHNIQUE 9

Dynamic programming – Principle of optimality – Coin changing problem, Computing a Binomial Coefficient – Floyd's algorithm – Multi stage graph – Optimal Binary Search Trees – Knapsack Problem and Memory functions. Greedy Technique – Container loading problem – Prim's algorithm and Kruskal's Algorithm – 0/1 Knapsack problem, Optimal Merge pattern – Huffman Trees.

UNIT IV ITERATIVE IMPROVEMENT 9

The Simplex Method – The Maximum-Flow Problem – Maximum Matching in Bipartite Graphs, Stable marriage Problem.

UNIT V COPING WITH THE LIMITATIONS OF ALGORITHM POWER 9

Lower – Bound Arguments – P, NP NP- Complete and NP Hard Problems. Backtracking – n-Queen problem – Hamiltonian Circuit Problem – Subset Sum Problem. Branch and Bound – LIFO Search and FIFO search – Assignment problem – Knapsack Problem – Travelling Salesman Problem – Approximation Algorithms for NP-Hard Problems – Travelling Salesman problem – Knapsack problem.

UNIT I

Scanned by TapScanner

UNIT-I

① What is an Algorithm? [APR/MAY '17]

⇒ Algorithm is a sequence of unambiguous instructions for solving a specific computational problem.

Problem
↓
Algorithm
↓
Input → Computer → output.

2) Write an algorithm to compute the greatest common divisor of 2 numbers. [APR/MAY '17] [MAY '18]

(OR)

Give the Euclid's algorithm for computing gcd (m,n) [MAY '16]

The GCD of 2 non-negative, not both 0 integers m and n, denoted as gcd(m,n) is defined as the largest integer that divides both m and n evenly (i.e) remainder is zero.

Euclid's Algorithm for gcd (m,n)

Step 1: if n=0, return the value of m as answer & stop; otherwise proceed to step 2.

Scanned by TapScanner

Step 2 : Divide m by n and assign the value of remainder to r.

Step 3 : Assign the value of n to m & the value of r to n. Goto step 1.

eg: gcd (60, 24) = gcd (24, 60 mod 24)
= gcd (24, 12)
= gcd (12, 24 mod 12)
= gcd (12, 0)
= 12 //

3. State the transpose symmetry property of O and Ω. [NOV/DEC '19]

$f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$

Example: If $f(n) = n$ and $g(n) = n^2$ then n is $O(n^2)$ and n^2 is $\Omega(n)$

Proof:

Necessary Part:

$f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))$ By the definition of Big-oh (O).

$\Rightarrow f(n) \leq C \cdot g(n)$ for some positive constant $C \Rightarrow g(n) \geq (1/c) \cdot f(n)$ By the definition of omega (Ω), $g(n) = \Omega(f(n))$.

Sufficient Part

$g(n) = \Omega((f(n)) \Rightarrow f(n) = O(g(n))$ By the definition of omega (Ω), for some positive constant $C \Rightarrow g(n) \geq c \cdot f(n) \Rightarrow f(n) \leq (1/c) \cdot g(n)$ By the definition of Big-oh (O), $f(n) = O(g(n))$.

Scanned by TapScanner

4 Define recursion.

$$x(n) = x(n-1) + n \text{ for } n > 0$$

⇒ An equation, such as it called a recurrence equation or recurrence relation (or simply a recurrence). An initial condition can be given for a value of n other than 0, eg. for n=1 and for some recurrences more than on value needs to be specified by initial conditions (eg) for recurrence $F(n) = F(n-1) + F(n-2)$ defining the fibonacci numbers.

5) How do you measure the efficiency of an algorithm.

Space Complexity:

⇒ The space complexity of an algorithm is the amount of memory it needs to run to completion.

Time Complexity:

⇒ The time complexity of an algorithm is the amount of time (or the number of steps) it needs to run to completion.

6) Design an algorithm to compute the area and circumference of a circle. [Nov/Dec '16]

Step 1: Read the input of the radius r.

Scanned by TapScanner

Step 2 Calculate Area = Pi * r * r and
circumference = 2 * Pi * r.

Step 3 : Print the o/p of Area and circumference
of the circle.

(7) Compare the orders of growth of $n(n-1)/2$ & $2n^2$ [May/June '16]

(i) $\frac{n(n-1)}{2} \in O(n^2)$

$\therefore \frac{n(n-1)}{2} \leq c\,n(n)$

$\Rightarrow \frac{n(n-1)}{2} \leq \frac{c\,n(n)}{2}$

$\Rightarrow \frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2$

$\therefore c = 1/2 \;,\; n_0 = 0$

$c = 1 \;,\; n = 4 \Rightarrow 8 - 2 \leq 16$

$c = 1 \;,\; n = 2 \Rightarrow 1 \leq 4.$

ii) $\frac{n(n-1)}{2} \geq c \cdot n^2$

$n-1 \geq n/2$

$\therefore$ for $n \geq 2$

$\frac{n(n-1)}{2} \geq n/2\,(n-1)$

$\geq \frac{n}{2} \times \frac{n}{2}$ ($\because n-1 \geq n/2$)

$= 1/4\, n^2$

$\therefore \epsilon = 1/4 \quad n_0 = 2.$

AEC

Scanned by TapScanner

(8) what is Big 'O' Notation?

The Big 'oh' notation provides an upper bound for the function f. A function f(n) is said to be in O(g(n)), if there exist some positive constant C and some non negative number No, such that,

$f(n) <= (g(n)$, for all $n >= no$.

9) what is Pseudo Code?

Pseudo Code is a mixture of Natural Language and programming language constructs such as functions, loops, decision making statements ... etc.

(10) Define amortized efficiency?

In some situations a single operation can be expensive, but the total time for the entire sequence of n such operations always significanlty better that the worst case efficiency of that single operation multiplied by n. This is called amortized efficiency.

(11) List the features of efficient algorithm.

⇒ free of ambiguity * Definiteness finiteness
⇒ Efficient in execution time
⇒ Concise and compact completeness.

Scanned by TapScanner

12) Define order of Algorithm:

⇒ The order of algorithm is a standard notation of an algorithm that has been developed to represent function that bound the computing time for algorithms.

⇒ The order of an algorithm is a way of defining its efficiency. It is usually referred as O-notation.

13) What is the substitution Method?

One of the methods for solving any such recurrence relation is called the substitution Method.

Types
① Forward substitution
② Backward substitution

14) What is a basic operation?

A basic operation is one that best characterizes the efficiency of the particular algorithm of interest.

15) List the desirable properties of algorithm?

Precision – The steps are precisely stated (defined).

Uniqueness – results of each step are uniquely defined and only depend on the i/p and the result of the preceding steps.

Scanned by TapScanner

Finiteness – The alg stops after a finite no of instructions are executed.

I/P – The algorithm receives I/P

O/P – The algorithm produces O/P.

Generality – The algorithm applies to a set of inputs.

16) Differentiate : Mathematical and Empirical analysis

Mathematical Analysis	Empirical Analysis
* The algorithm is analyzed with the help of mathematical derivations and there is no need of specific i/p. * The principal weakness is limited applicability * The principal strength is it is independent of any i/p	* The alg is analyzed by taking some sample of input and no mathematical derivation is involved. * The principal strength is it is applicable for any alg. * The principal weakness is it depends upon the sample i/p.

17 What is algorithm Visualization ?

Algorithm Visualization can be defined as the use of images to convey some useful information about algorithms.

Two principal variations are static algorithm visualization

Dynamic Algorithm Visualization.

Scanned by TapScanner

18) How can you classify Algorithms.

⇒ Among several ways to classify algorithms, the 2 principal alternatives are,

1) To group algorithms according to types of problem they solve completely.

2) To group algorithms according to underlying design techniques they are based upon.

19) List 5 of basic efficiency classes.

* log n logarithmic
* n linear
* n log n n-log-n
* n2 quadratic
* 2n exponential.

20) what is the formula used to calculate the algorithm's running time?

⇒ The running time T(n) of a program implementing the algorithm on a computer is given by the formula

$$T(n) = C_{op} \times C(n)$$ where Cop is the time of execution of an algorithm's basic operations C(n) is the no. of. times the basic operation is executed.

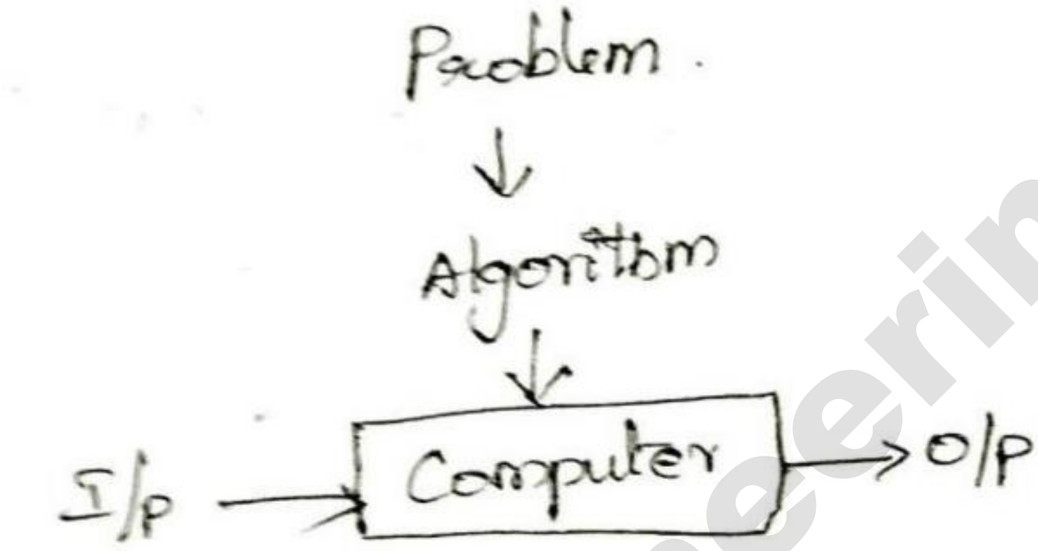
Scanned by TapScanner

UNIT - I

INTRODUCTION : Notion of an algorithm.

Algorithm

An algorithm is a sequence of unambiguous instructions for solving a problem, (ie) for obtaining a required o/p for any legitimate input in a finite amount of time.



1. Specify the Euclid's algorithm, the consecutive integer checking alg and the middle school alg for computing the greatest common divisor of two integers.

⇒ To illustrate the notion of an alg, let us consider three methods for solving the same problem. Computing the GCD of 2 integers:

⇒ 3 Methods are

1. Euclid's alg
2. Consecutive integer checking alg
3. Middle - school procedure.

Scanned by TapScanner

Euclid's Algorithm

* Euclid's algorithm is based on applying repeatedly the equality

gcd (m,n) = gcd (n, m mod n)

* where m mod n is the remainder of the division m by n) until m mod n is equal to 0, the last value of m is also the gcd of the initial m and n

Example : gcd (60, 24)

gcd (60, 24) = gcd (24, 60mod 24)
= gcd (24, 12)
= gcd (12, 24 mod 12)
= gcd (12, 0)
= 12.

Euclid's algorithm of computing gcd (m,n)

steps:

step 1 : If n=0, return the value of m as the answer and stop ; otherwise proceed to step 2.

Step 2 : Divide m by n and assign the value of the remainder to r.

Step 3 : Assign the value of n to m and the value of r to n. Go to step 1.

Scanned by TapScanner

ALGORITHM (Euclid (m,n)) in a pseudocode :

// compute gcd (m,n) by Euclid's alg

// Input : Two non-negative and non-zero integers
m and n.

// output : Greatest common divisor of m and n.

```
while n ≠ 0 do
        r ← m mod n
        m ← n
        n ← r
return m
```

2. Consecutive integer checking algorithm for computing gcd (m,n).

⇒ A common divisor cannot be greater than the smaller of the two numbers which we will denote by t.

$$t = \min\{m, n\}$$

⇒ check whether t divides both m and n; if it does, t is the answer; if it does not, we simply decrease t by 1 and try again.

Scanned by TapScanner

Steps:

Step 1 : Assign the value of min {m,n} to t.

Step 2 : Divide m by t. If the remainder of this division is 0, go to step 3; otherwise go to step 4.

Step 3 : Divide n by t. If the remainder of This division is 0, return the value of t as the answer and stop; otherwise proceed to step 4.

Step 4 : Decrease the value of t by 1. Go to step 2.

Example:

Consider m=12, n=8

t = min(12,8)

Set value of t=8 initially.

check m mod t = 0

and n mod t = 0

if not, then decrease t by 1 and again with this new t value check whether

m mod t = 0

and. n mod t = 0

⇒ Thus we go on checking whether m mod t and n mod t both are resulting 0 or not.

Scanned by TapScanner

Example : gcd (12, 8)

m	n	Explanation
12 mod 8 = 4	8 mod 8 = 0	12 mod 8 is not equal to zero. So 8 is not a gcd Set new t = t - 1 t = 8 - 1 t = 7.
12 mod 7 = 5	8 mod 7 = 1	Both 12 mod 7 & 8 mod 7 are not equal to zero. So 7 is not a gcd Set new t = t - 1 new t = 7 - 1 t = 6
12 mod 6 = 0	8 mod 6 = 2	12 mod 6 is equal to zero But 8 mod 6 is not equal to zero. So 6 is not a gcd. Set new t = t - 1 t = 6 - 1 ie t = 5
12 mod 5 = 7	8 mod 5 = 3	Both 12 mod 5 & 8 mod 5 are not equal to zero. So 5 is not a gcd Set new t = t - 1 = 5 - 1 t = 4.
12 mod 4 = 0	8 mod 4 = 0	Both 12 mod 4 & 8 mod 4 are equal to zero. t = 4 is a gcd.

gcd (12, 8) = 4.

Scanned by TapScanner

3. Middle school procedure for computing gcd(m,n).

steps:

Step 1 : Find the prime factors of m

Step 2 : Find the prime factors of n

Step 3 : Identify all the common factors in the two prime expansions found in step 1 and step 2.

Step 4 : Compute the product of all the common factors and return it as the gcd of the numbers given.

Thus for the numbers 120 and 972 we get,

$$120 = 2 \times 2 \times 2 \times 3 \times 5$$

$$72 = 2 \times 2 \times 2 \times 3 \times 3$$

$$\boxed{\gcd(120, 72) = 2 \times 2 \times 2 \times 3 = 24.}$$

Scanned by TapScanner

PART - B

② Discuss in detail about fundamentals of algorithm problem solving.

(or)

Discuss the sequence of steps in designing and analyzing an algorithm.

Steps

1. Understand the problem
2. Decide on
   (a) Computational means
   (b) Exact vs approximate solving
   (c) Data structure(s)
   (d) Algorithm design techniques
3. Design an algorithm
4. Prove correctness
5. Analyze the algorithm
6. Code the algorithm.

Algorithm Design and Analysis process

① Understanding the problem

* While understanding the problem statements, read the problem description carefully & ask questions

Scanned by TapScanner

ask questions for clarifying the doubts about the problem.

* After understanding the problem statements find out what are the necessary inputs for solving that problem.

* The input to the algorithm is called instance of the problem.

* It is very important to decide the range of inputs so that the boundary values of algorithm get fixed.

* The algorithm should work correctly for all valid i/ps.

2. Decision Making

* To analyse the input and need to decide certain issues like,

(a) Computational means

(b) Exact Vs Approximate solving

(c) Data structures

(d) Algorithm design techniques

(a) Computational means

* It is necessary to know the computational capabilities of devices on which the algorithm will be running.

Scanned by TapScanner

① Sequential Algorithm
② Parallel Algorithm

(b) Exact Vs Approximate solving.

⇒ If the problem needs to be solved exactly or correctly then we can use exact algorithm.

⇒ If we want to solve a complex alg problem, we will not get the exact solution. That situation is called approximation algorithm.

* Example Travelling Salesman problem.

(c) Data structure(s)

⇒ Data structure and algorithm work together and there are Independent. Hence choice of proper data structure is required to before designing the actual algorithm.

(d) Algorithm Design Techniques.

⇒ An algorithm design Techniques is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

Techniques are,

* Brute-force * Divide & Conquer

Scanned by TapScanner

* Decrease and Conquer * Transform & Conquer
* Dynmii Programming * Greedy Technique.

3) Specification of Algorithm

* There are different ways by which we can specify an algorithm.

(a) Using natural language

(b) Pseudo code

(c) flow chart

Example: Write an algorithm to perform addition of two numbers.

(a) Using Natural Language.

* It is very simple to specify an algorithm using natural language. But many times specification of alg by natural language is not clear.

1. Read the first number say 'a'
2. Read the second number say 'b'
3. Add the two numbers and store the result in a variable.
4. Display the result.

Scanned by TapScanner

(b) Pseudo code Method

ALGORITHM Sum(a,b)

// Problem Description : This performs addition of two numbers

// Input : Two integers a and b

// Output : Addition of two integers.

C ← a+b.

return C.

(c) Flow chart:

- (START) Start state
- Transition
- Processing (or) Assignment stmts
- Input-O/P Statement
- Conditional Statement
- (STOP) Stop state

4. Proving an Algorithm's Correctness

⇒ Prove that the algorithm yields a required result for every legitimate input in a finite amount of time.

5. Analysis of an Algorithm

⇒ (a) Time efficiency (b) Space efficiency.
(c) Simplicity (d) Generality.

Scanned by TapScanner

6. Coding an Algorithm.

⇒ It is done by suitable programming language.

(2) Explain the important problem types in detail.

① Sorting
② Searching
③ String Processing
④ Graph problems
⑤ Combinational Problems
⑥ Geometric Problems
⑦ Numerical Problems.

① Sorting.

* Sorting means arranging the elements in increasing order (or) decreasing order.

* The sorting can be done on numbers, characters strings or employees record.

* For sorting any record we need to choose certain piece of information based on which Sorting can be done.

Eg: Keeping the employees record in sorting order we will rearrange the employees record as per employee ID.

⇒ Two properties of sorting algorithms are,
(a) stable property (b) In place property.

Scanned by TapScanner

(a) Stable property:

* A sorting algorithm is called stable if it preserves the relative order of any two equal elements in its input.

(b) In place property

* An algorithm is said to be in place if it does not require extra memory, except, possibly, for a few memory unit.

2) Searching:

* Searching is an activity by which we can find out the desired element from the list. The element which is to be searched is called search key.

* eg: Sequential search (Straight forward technique)
Fibonacci search
Binary search

3. String Processing

* A string is a sequence of characters from an alphabet. Strings of particular interrupt are text strings, which comprise letters, numbers and special characters.

Scanned by TapScanner

A. Graph problems

*A Graph can be thought of as a collection of points called vertices, some of which are connected by line segments called edges.

eg: Graph traversal algorithms
Shortest-path algorithms
Topological sorting for graphs

5. Combinational problems

*The combinational problems are related to the problems like computing combinations and permutation.

⇒There is no algorithm available which can solve these problems in finite amount of time.

⇒Many of these problems fall in the catogory of unsolvable problems.

6. Geometric Problems

Geometric algorithms deal with geometric objects such as points, lines and polygons.

7. Numerical problems

*Numerical problems are problems that involve mathematical objects of continuous nature.

*The majority of such mathematical problems can be solved only approximately.

Scanned by TapScanner

Amortized Efficiency.

* It applies not to a single run of an algorithm but rather to a sequence of operations performed on the same data structure.

* Amortized analysis means finding the average running time per operation over a worst case sequence of operations.

4 Explain briefly Big-oh Notation, omega Notation & Theta Notations [Imp/Important]

Asymptotic Notations and their properties:

* Algorithm's efficiency is determined by the order of growth of that algorithm. order of growth is determined by the basic operation count C(n).

⇒ 5 notations

① O (big oh)
② Ω (big omega)
③ Θ (big theta)
④ Little oh notation (o)
⑤ Little omega notation (ω)

⇒ t(n) and g(n) are non-negative functions defined on the set of natural numbers.

t(n) → The algorithm's running time
g(n) → A some simple function to compare the count.

Scanned by TapScanner

Informal Introduction

$O(g(n))$ is the set of all functions with a smaller or same order of growth as $g(n)$.

$n \in O(n^2)$, $100n+5 \in O(n^2)$, $\frac{1}{2}n(n-1) \in O(n^2)$

⇒ The first two functions are linear and hence have a smaller order of growth than $g(n) = n^2$, while the last one is quadratic and hence has the same order of growth as n^2, on the other hand,

$$n^3 \notin O(n^2),\ 0.00001n^3 \notin O(n^2),\ n^4+n+1 \notin O(n^2)$$

⇒ The functions n^3 and $0.00001n^3$ are both cubic and hence have a higher order of growth than n^2 and so has the fourth-degree polynomial n^4+n+1.

⇒ The second notation, $\Omega(g(n))$, stands for the set of all functions with a larger or same order of growth as $g(n)$.

$$n^3 \in \Omega(n^2),\ \frac{1}{2}n(n-1) \in \Omega(n^2),\ \text{but } 100n+5 \notin \Omega(n^2)$$

⇒ Finally, $\Theta(g(n))$ is the set of all functions that have the same order of growth as $g(n)$.

⇒ Thus every quadratic function an^2+bn+c with $a>0$ is in $\Theta(n^2)$, but so are, among infinitely many others $n^2+\sin n$ and $n^2+\log n$.

Scanned by TapScanner

Big-oh - Notation (O)

→ Method of representing the upper bound of alg's running time.

→ Used to define the worst-case complexity & give longest amount of time taken by the alg to complete.

→ Concerned with large values of n.

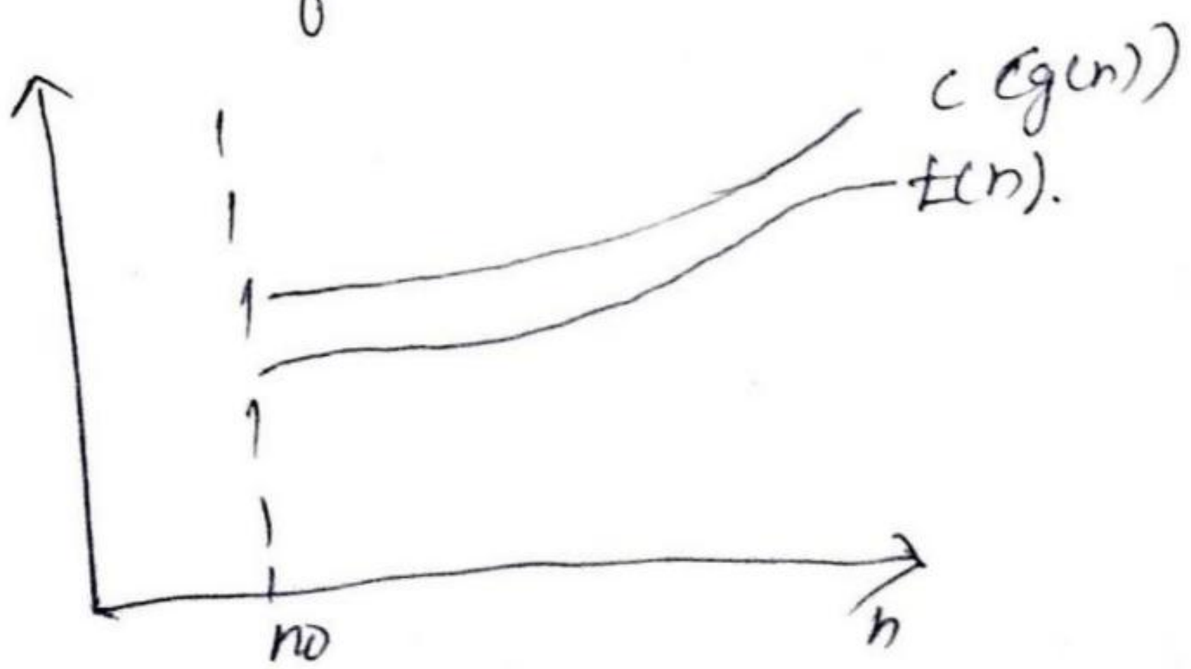
→ O(g(n)) is the set of all functions with a smaller or same order of growth as g(n).

Definition

* A function $t(n)$ is said to be in $O(g(n))$, denoted as $t(n) \in O(g(n))$, if $t(n)$ is bounded above by some constant multiple of $g(n)$, for all large n.

* (i.e) if there exist some positive constant C and some non-negative integer n_0 such that $t(n) \leq c\, g(n)$ $\forall\, n \geq n_0$.

O(g(n)): class of functions t(n) that grow no faster than g(n). Big-oh puts asymptotic upper bound on a function.



Scanned by TapScanner

Eg:

Consider function $f(n) = 2n+2$ and $g(n) = n^2$. Then we have to find some constant C, So that $f(n) \leq C * g(n)$.

As $f(n) = 2n+2$ and $g(n) = n^2$ then we find C for $n=1$ then

$$f(n) = 2n+2$$
$$= 2\times1 + 2 = 4$$

$g(n) = n^2$ (ie) $f(n)$ is not less than $g(n)$
$= 1$

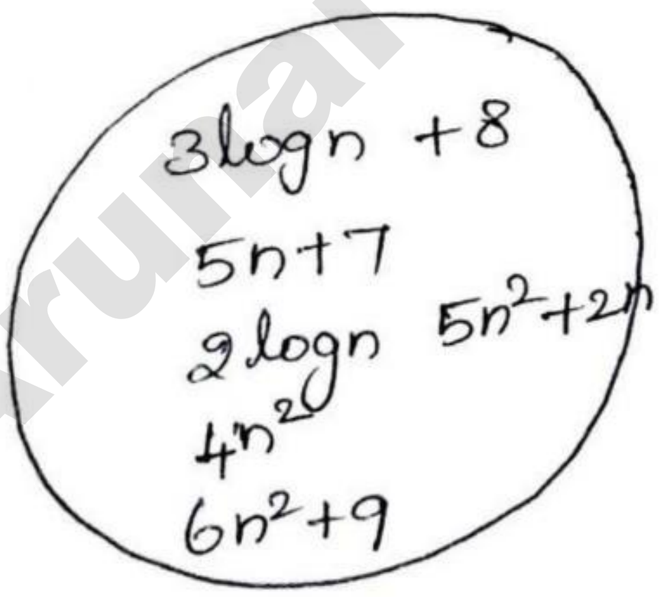
if $n=2$ $f(n) = 2\times2 + 2$ $g(n) = n^2$
$= 4+2 = 6$ $= 2^2 = 4$

(ie) $f(n) \not< g(n)$

if $n=3$ $f(n) = 2\times3+2$ $g(n) = 3^2 = 9$
$= 8$

$f(n) < g(n)$ is true.

Hence we conclude for $n>2$, we obtain $f(n) < g(n)$



Represents the classes of linear, logarithmic, quadratic functions that belong to $O(n^2)$.

Scanned by TapScanner

Big - Omega Notation

→ Used to describe the best case running time of algorithms & concerned with large values of n.

Definition

→ A function $t(n)$ is said to be in $\Omega(g(n))$, denoted as $t(n) \in \Omega(g(n))$, if $t(n)$ is bounded below by some positive constant multiple of $g(n)$ for all large n.

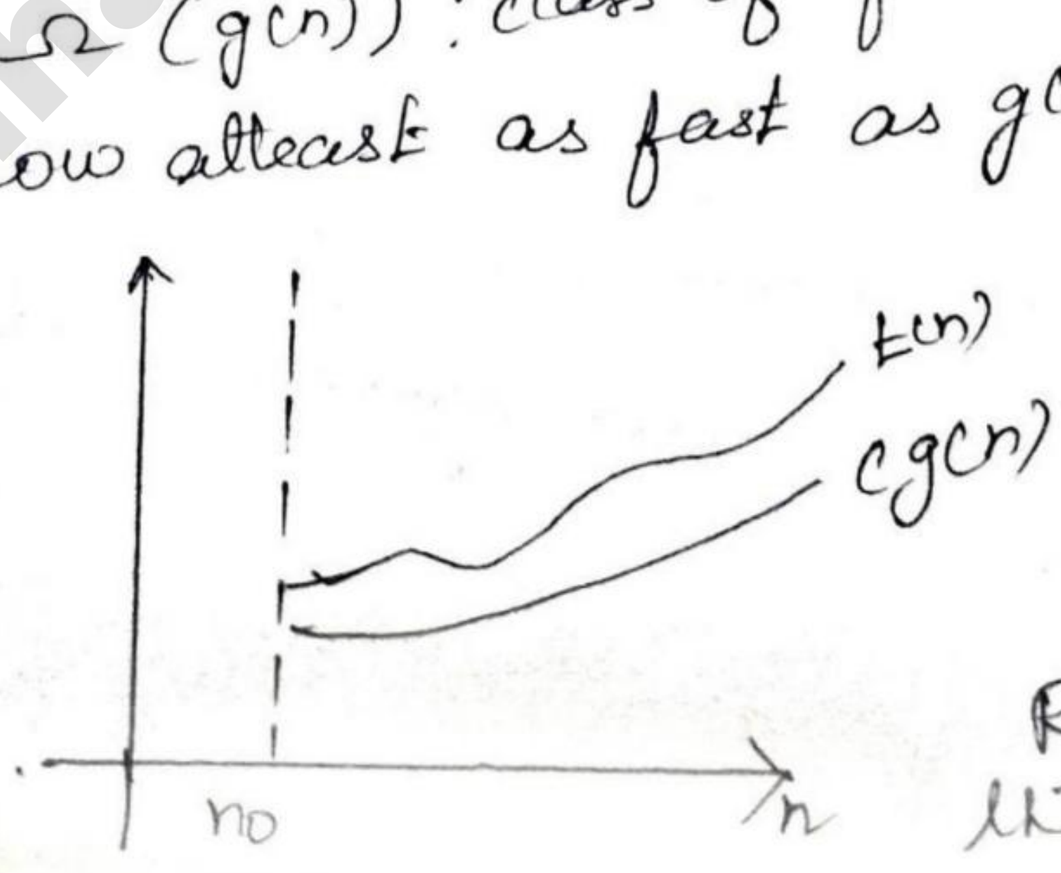
(ie) There exist some positive constant c & some non negative integer n_0, such that

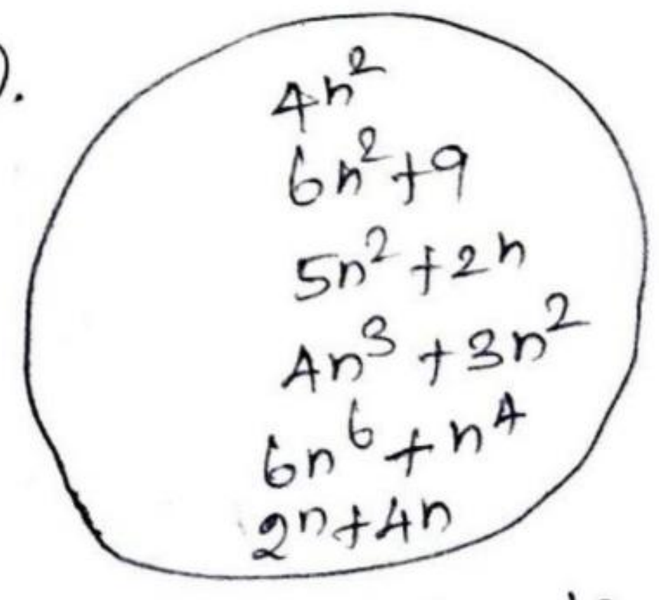
$$t(n) \geq c\,g(n) \quad \forall\ n \geq n_0.$$

→ It represent the lower bound of the resources required to solve a pbm.

→ $\Omega(g(n))$ stands for the set of all functions with a larger or same order of growth as $g(n)$

→ $\Omega(g(n))$: class of functions $t(n)$ that grow atleast as fast as $g(n)$.





Represents the functions that belongs to $\Omega(n^2)$.

Scanned by TapScanner

eg Consider $f(n) = 2n^2+5$ and $g(n) = 7n$

Then if $n=0$

$f(n) = 2(0)^2+5 = 5$

$g(n) = 7(0)$
$= 0$
$f(n) > g(n)$

if $n=1$

$f(n) = 2+5 = 7$ $\quad f(n) = g(n)$

$g(n) = 7(1) = 7$

if $n=2$

$f(n) = 4+5 = 9$ $\quad f(n) \leq g(n)$

$g(n) = 7(2) = 14$

if $n=3$

$f(n) = 2(3^2)+5 = 23$ $\quad f(n) > g(n)$

$g(n) = 7(3) = 21$

$\therefore 2n^2+5 \in \Omega(n)$

Similarly any $n^3 \in \Omega(n^2)$.

Theta Notation (Θ)

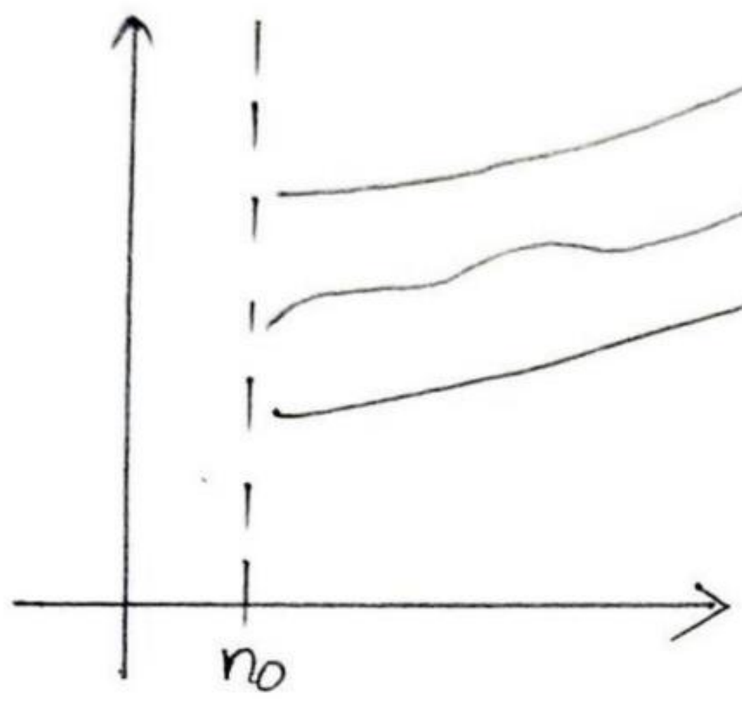
$\Rightarrow$ A function $t(n)$ is said to be in $\Theta(g(n))$, denoted by $t(n) \in \Theta g(n)$, if $t(n)$ is bounded both above & below by some positive constant multiples of $g(n)$ for all large n.

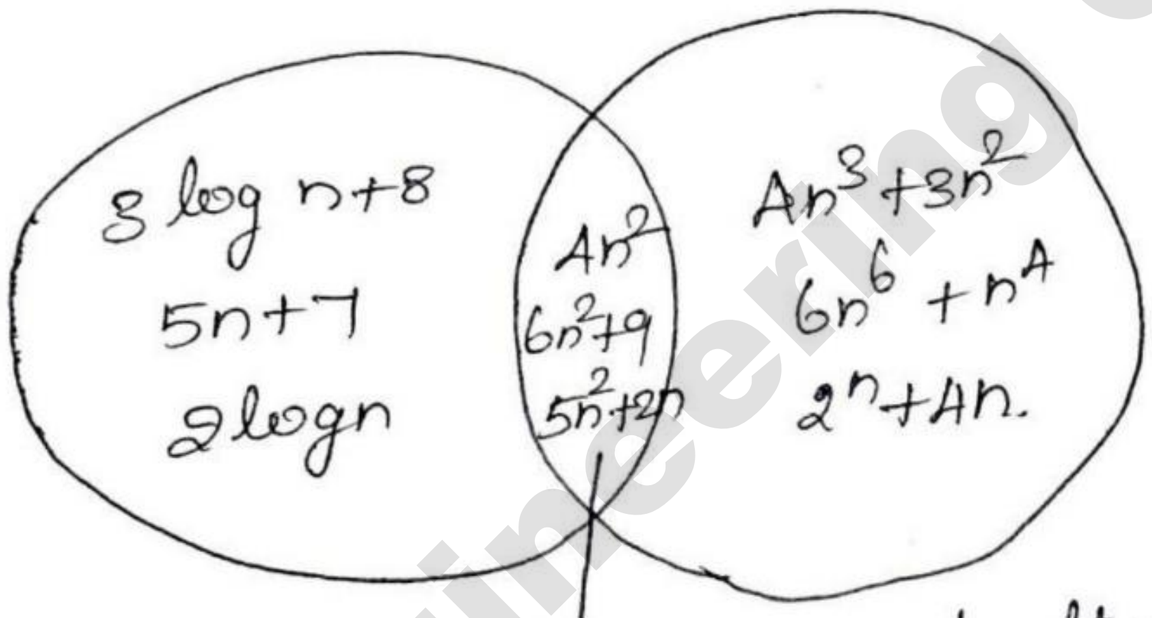
i.e) if there exist some positive constant C_1 & C_2 & some non-negative integer n_0 such that

Scanned by TapScanner

$C_2 g(n) \leq t(n) \leq C_1 g(n) \quad \forall \; n > n_0.$

$\Theta(g(n)) = O(g(n))$ n $\sim$ $(g(n))$





Represents the functions that belongs to $O(n^2)$.

Little - oh notation : (o)

→ Used to describe worst case analysis of algorithms & concerned with small values of n.

Definition :

→ A function $t(n)$ is said to be in $O(g(n))$, denoted $t(n) \in o(g(n))$, if there exist some

Scanned by TapScanner

positive constant C & some non-negative integer n such that

$$f(n) \leq C * g(n).$$

→ $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$

Little omega Notation : (ω)

→ Notation used to describe the best-case analysis of algorithms & concerned with small values of n.

→ function $f(n) = \omega(g(n))$ iff

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \text{ (or)}$$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0.$$

Scanned by TapScanner

(5) Write an algorithm for determining the uniqueness of an array. Determine the time complexity of your algorithm.

Algorithm

// check whether all the elements in a given array are distinct.
// Input : An array A[0...n-1]
// output : Returns "true" if all the elements in A are distinct and "false" otherwise

```
for i ← 0 to n-2 do
    for j ← i+1 to n-1 do
        if A[i] = A[j] return false
```

Mathematical Analysis:

Step 1 : The input size is n. (ie) The total number of elements in the array.

Step 2 : The basic operation will be comparison of two elements.

step 3 : The number of comparisons, will depend upon the input n. We will limit our investigation to the worst case only.

Scanned by TapScanner

Step 4 : The worst case input is ~~given~~ when ~~an~~ we require largest number of comparisons for the array of size n. The worst case time is denoted by $C_{worst}(n)$. There are two types of worst case inputs.

(i) when there are no equal elements in the array.

ii) The last two elements are equal in the array.

⇒ For such type of inputs one comparison is made for each value of j (inner loop) ranging from i+1 to n-1.

⇒ This inner loop will be repealed for each value of (outer loop) and i varies from 0 to n-2.

So we get,

$$C_{worst}(n) = \text{Outer loop} \times \text{Inner loop}$$

$$C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=j+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} ((n-1) - (i+1) + 1)$$

$$\left\{ \text{Ref} : \sum_{\text{lower limit}}^{\text{upper limit}} 1 = \left\{ (\text{upper limit}) - (\text{lower limit}) + 1 \right\} \right\}$$

Scanned by TapScanner

$$= \sum_{i=0}^{n-2} n-1-i-1+1$$

$$= \sum_{i=0}^{n-2} (n-1-i)$$

$$= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i$$

Ref:

$$\sum_{i=0}^{n} i = 0+1+2+\cdots n = \frac{n(n+1)}{2}$$

$$= (n-1)[n-2-0+1] - \frac{(n-2)[(n-2)+1]}{2}$$

$$= (n-1)(n-1) - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)(n-1) - \frac{(n-1)(n-2)}{2}$$

$$= \frac{(n-1)}{2}[2(n-1) - (n-2)]$$

$$= \frac{(n-1)}{2}[2n-2-n+2]$$

$$= \frac{(n-1)}{2} n$$

$$= \frac{n^2-n}{2}$$

$$\simeq \frac{1}{2} n^2$$

$C_{worst}(n) \in \Theta(n^2)$.

AEC

Scanned by TapScanner

b) Use the most effici. appropriate notation to indicate the time efficiency class of a sequential search
(i) in the worst case ii) in the best case
iii) in the average case.

(i) Worst Case Efficiency / Worst Case Time Complexity.

⇒ If an algorithm takes maximum amount of time to run to completion for a specific set of input, then it is called worst time complexity.

Example,

⇒ While searching a particular element by using linear search we get the desired element at the end of the list then it is called worst case time complexity.

$$C_{worst}(n) = n$$

* The algorithm guarantees that for any instance of input which is of size n, the running time will not exceed $C_{worst}(n)$.

Best Case Efficiency / Best Case Time Complexity.

* If an algorithm takes minimum amount of time to run to completion for a specific set of i/p then it is called best time complexity

Scanned by TapScanner

Example:

* while searching a particular element by using Sequential search we get the desired element at first place itself then it is called best case time complexity.

* If the key element is present at first location in the list (x[0---n-1]) then algorithm runs for a very short time and thereby we will get the best case time complexity.

$$C_{best}(n) = 1$$

Average Case Efficiency / Average Case Time Complexity-

* This type of complexity gives information about the behaviour of an algorithm on specific or random input.

* Let the algorithm is for sequential search and p be a probability of getting successful search, n is the total no. of elements in the list.

* The first match of the element will occur at i^{th} location - Hence probability of occuring first match is p/n for every i^{th} element.

Scanned by TapScanner

* The probability of getting unsuccessful search is (1-p):

$C_{avg}(n)$ = Probabilies of successful search (For elements 1 to n in the list) + Probability of unsuccessful search.

$$= \left[1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + 3 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}\right] + n \cdot (1-p)$$

$$= \frac{p}{n}\left[1+2+3+\cdots+i+\cdots+n\right] + n(1-p)$$

$$= \frac{p}{n}\,\frac{n(n+1)}{2} + n(1-p)$$

$$= \frac{p(n+1)}{2} + n(1-p)$$

$$C_{avg}(n) = \frac{p(n+1)}{2} + n(1-p)$$

⇒ The general formula yields some quite reasonable answers.

For eg if $p = 1$,

$$C_{avg}(n) = 1\,\frac{(n+1)}{2} + n(1-1)$$

$$= \frac{n+1}{2}$$

(ie) The search is successful.

(ie) On average, about half of the list's elements will be inspected, by the algorithm.

Scanned by TapScanner

For example if $p=0$

$$C_{avg}(n) = \frac{0(n+1)}{2} + n(1-0)$$

$$= 0+n = n$$

$$C_{avg}(n) = n$$

(ii) The average no. of. key comparisons will be n, because the algorithm will inspect all n elements.

(iii) The search is unsuccessful.

⇒ Computing average case time complexity is difficult than computing worst case and best case time complexities.

⇒ Average-case efficiency cannot be obtained by taking the average of the worst case and the best-case efficiencies.

(f) Briefly explain the mathematical analysis of recursive and non-recursive algorithm.

General plan for Analysing Efficiency of Non-Recursive Algorithms

1. Decide on a parameter indicating an input's size.
2. Identify algorithm's basic operations.

Scanned by TapScanner

8) Check whether the no. of times the basic operation is executed depends only on the size of an ip.

* If it also depends on some additional property, the worst-case, average case and if necessary the best case efficiencies have to be investigated separately.

4) Set up a sum expressing the no. of times the algorithm's basic operation is executed.

5) Using standard formulas and rules of sum manipulation, either find a closed form formula for the count or at the very least, establish its order of growth.

Summation Formulas and Rules

1. $\sum_{i=1}^{n} i = 1+1+1+\cdots+1 = n \in \Theta(n)$

2. $\sum_{i=1}^{n} i = 1+2+3+\cdots+n = \frac{n(n+1)}{2} \approx \frac{1}{2}n^2 \in \Theta(n^2)$

3. $\sum_{i=1}^{n} i^k = 1+2^k+3^k+\cdots+n^k = \frac{n^{k+1}}{k+1} \in \Theta(n^{k+1})$

4. $\sum_{i=1}^{n} a^i = a+a^2+\cdots+a^n = \frac{a^{n+1}-1}{a-1} \in \Theta(a^n)$

5. $\sum_{i=1}^{n}(a_i \pm b_i) = \sum_{i=1}^{n} a_i \pm \sum_{i=1}^{n} b_i$

6. $\sum_{i=1}^{n} ca_i = c\sum_{i=1}^{n} a_i$

7. $\sum_{i=k}^{n} 1 = n-k+1$, where n & k are upper and lower limits resp.

Scanned by TapScanner

Example :

Counting number of bits in a positive decimal integer.

ALGORITHM Binary (n)

//Input : A positive decimal integer n.

//output : The no. of binary digits in n's binary representation.

```
Count ← 1
while n>1 do
    Count ← Count+1
        n ← ⌊n/2⌋
    return count
```

Mathematical Analysis

1. The input size is n
2. The basic operation is denoted by while loop. And it is each time checking whether n>1. The while loop will be executed for the no. of time at which n>1 is true.
   * It will be executed once more than n>1. is false. But when n>1 is false the statements inside while loop won't get executed.
3. The value of n is about halved on each repetition of the loop.

Scanned by TapScanner

A. The exact formula for the no. of times the comparison $n>1$ will be executed is actually.

$$\lfloor \log_2 n \rfloor + 1$$

↑ floor value

* Time complexity of the algorithm for finding the no. of bits of a given number is $\Theta(\log_2 n)$

Mathematical Analysis of Recursive Algorithms:

* If an algorithm calls itself again and again for solving the problem then these algorithms were called Recursive Algorithms.

1) Decide on a parameter indicating an input's size.
2) Identify the algorithm's basic operation.
3) Check whether the no. of times the basic operation is executed can vary on different inputs of the same size:
   * If it can, the worst case, avg case and best case efficiencies must be investigated separately.
4) Set up a recurrence relation, with an appropriate initial condition, for the no. of times the basic operation is executed.

Scanned by TapScanner

5. Solve the recurrence or atleast ascertain the order of growth of its solution.

Example

1. Computing factorial of 'n' recursively.
2. Tower of Hanoi puzzle.
3. Finding the no. of binary digits present in n's binary representation
4. Generating 'n' fibonacci numbers recursively.

Example

Finding the no. of binary digits in the binary representation of a positive decimal integer.

ALGORITHM Binary(n)

```
//Input : A positive decimal integer n
//output : The no.of. binary digits in n's binary
           representation.
    Count ← 1
    while n > 1 do
        Count ← Count + 1
        n ← ⌊n/2⌋
    return count
```

Mathematical Analysis

1. Input size = n.
2. Basic operation is division by 2.

Scanned by TapScanner

3. Number of additions made is $A(n)$.

A The recurrence relation is

$$A(n) = A+1 \text{ for } n>1$$

Initial condition is, $A(1)=0$.

5 The standard approach to solve a recurrence is to solve if only for $n=2^k$ and take advantage of the theorem called the smoothness rule.

$n=2^k$ in the recurrence recurrence relation we get,

$$A(2^k) = A(2^{k-1}) + 1 \text{ for } k>0$$

$$= [A(2^{k-2}) + 1] + 1$$

$$= A(2^{k-2}) + 1 + 1$$

$$= A(2^{k-2}) + 2$$

$$= [A(2^{k-3}) + 1] + 2$$

$$= A(2^{k-3}) + 1 + 2$$

$$= A(2^{k-3}) + 3$$

- - - -

- - - -

$$= A(2^{k-i}) + i$$

- - -

- - -

$$= A(2^{k-k}) + k$$

$$= A(i^0) + k \Rightarrow A(1) + k$$

$$\left\{ A(2^{k-1}) = A(2^{k-2}) + 1 \right.$$

Scanned by TapScanner

UNIT II

Scanned by TapScanner

UNIT - II

① State the convex hull problem.

A shape or set is convex if for any two points that are part of the shape, the whole connecting line segment is also part of the shape.

② Outline the knapsack problem?

The knapsack with maximum capacity W and a set S consisting of n items. Each item i has some weight w_i and benefit value V_i. The knapsack has to be packed to achieve maximum total value.

⇒ It is mathematically defined as,

$$\max \sum_{i \in T} V_i \text{ subject to } \sum_{i \in T} w_i \leq W$$

③ Write the brute force algorithm to string matching.

Algorithm BruteforceStringMatch (T[0...n-1], P[0...m-1])

// The algorithm implements brute-force string matching

// Input : An array T[0...n-1] of n characters rep^t a text;

// an array P[0...m-1] of m characters rep^t a pattern;

// O/P : The position of the first character in the text.

// that starts the first matching substring if the

// search is successful and -1 otherwise.

Scanned by TapScanner

```
for  i ← 0 to n-m do
       j ← 0
       while j < m and p[j] = T[i+j] do
       j ← j+1
     if j = m return i
     return -1
```

4. Write the brute force algori

A. What is the time and space complexity of merge sort?

MergeSort time Complexity is O(nlgn) which is a fundamental knowledge.

MergeSort Space complexity will always be O(n) including with arrays.

5. what are the differences between dynamic pgming & divide and conquer approaches?

Divide and Conquer	Dynamic programming
* It is Recursive	* It is Non Recursive.
* It does more work on subproblems & hence has more time consumption.	* It solves subproblems only once and then stores in the table
* It is top-down approach	* It is a Bottom-up approach
* In this subproblems are independent of each other	* In this subproblems are independent
eg; Merge Sort & Binary Search	(eg) Matrix Multiplication

Scanned by TapScanner

6) what is an exhaustive Search?

* A brute force solution to a problem involving Searching for an element with a special property, usually among combinatorial objects such as permutations, combinations or subsets of a set is termed as exhaustive Search.

7) Give the general plan of divide and conquer algorithms.

* The general strategy of Divide and conquer method is to divide and conquer the problem in to smaller instances of the same problem & then solve (Conquer) the smaller instances recursively and finally combine the solutions to obtain the solution of original input.

8) what is the closest pair problem?

* The closest pair problem is to find the two closest points in a set of n points.

9) Write an algorithm for brute force closest-pair problem.

ALGORITHM Bruteforce ClosestPair (P)
// Finds distance between two closest points in the plane by brute force.

Scanned by TapScanner

// Input : A list p of n(n>=2) points $P_1(x_1,y_1)$ ... $P_n(x_n,y_n)$
// output : the distance b/w the closest pair of points.

```
d ← ∞
for i ← 1 to n-1 do
    for j ← i+1 to n do
        d ← min(d, sqrt((x_i - x_j)^2 + (y_i - y_j)^2))
return d.
```

10) what is worst case complexity of binary search?

The worst case Complexity of binary Search is $O(\log n)$.

11) Design a brute-force algorithm for computing the value of a polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x \ldots a_0$ at a given point x_0 and determine its worst-case efficiency.

Algorithm Better Brute force Polynomial Evaluation (P[0..n], x).

// The alg computes the value of polynomial P at a
// given point x, by the "lowest-to highest term"
// alg.
// I/P Array P[0..n] of the coefficients of a
// polynomial of degree n, from the lowest to the
// highest, and a number x.

Scanned by TapScanner

6) what is an exhaustive Search?

* A brute force solution to a problem involving Searching for an element with a special property, usually among combinatorial objects such as permutations, combinations or subsets of a set is termed as exhaustive Search.

7) Give the general plan of divide and conquer algorithms.

* The general strategy of Divide and conquer method is to divide and conquer the problem in to smaller instances of the same problem & then solve (Conquer) the smaller instances recursively and finally combine the solutions to obtain the solution of original input.

8) what is the closest pair problem?

* The closest pair problem is to find the two closest points in a set of n points.

9) Write an algorithm for brute force closest-pair problem.

ALGORITHM BruteforceClosestPair (P)
// Finds distance between two closest points in the plane by brute force.

Scanned by TapScanner

```
//o/p : the value of the polynomial at the point x.
p ← P[0];
Power ← 1 for i ← 1 to n do
power ← power * x
p ← p + P[i] * power
return P
```

Worst case efficiency $\in n^2$.

12) List out the 4 steps in Strassen's Method.

1. Divide the i/p matrices A and B into n/2 * n/2 sub matrices.
2. Using $\Theta(n^2)$ Scalar additions and subtractions, Compute 14 n/2 * n/2 matrices A1, B1, A2, B2 ... A7B7.
3. Recursively compute the 7 matrix products $P_i = A_iB_i$ for i=1,2,7.
4. Compute the desired sub matrices r, s, t, u of the result matrix C by adding and/or subtracting various combinations of the Pi matrices, using only $\Theta(n^2)$ Scalar additions and subtractions.

13) Illustrate the Assignment Problem.

* There are n people who need to be assigned to execute n jobs as one person per job. Each person is assigned to exactly one job and each job is assigned to exactly one person.

Scanned by TapScanner

14) Define profiling?

* Profiling is an important resource the empirical analysis of an algorithm running time. Measuring n different segments of program can pinpoint a bottleneck in the pgm's performance that can be missed by an abstract deliberation about the alg's basic operations. The process of getting such data is called profiling.

15) what is decrease and conquer technique?

* The decrease and conquer technique is based on exploiting the relationship b/w a solution to a given instance of a pbm and solution to a smaller instance of the same problem.

16) Write the efficiency of heap sort alg.

Heap sort is an in-place alg. Time complexity of heapify is O(Log n). Time Complexity of CreateAndBuildHeap() is O(n) and overall time complexity of HeapSort is O(nLogn)

17) what is efficiency of selection Sort

O(n2) time Complexity.

Scanned by TapScanner

18) Define Recursive Call?

An algorithm is said to be recursive if the same algorithm invoked in the body.

2 types (1) Direct Recursive
(2) Indirect Recursive

19) what is the Quick Sort and write the Analysis for the Quick sort?

* In quick Sort, the division into subarrays is made so that the sorted subarrays do not need to be merged later. In analysing QUICKSORT, we can only make the number of element comparisons C(n). It is easy to see that the frequency count of other operations is of the same order as C(n).

20) List the strength and weakness of brute force alg.

Strength:

(a) wide applicability
(b) Simplicity
(c) Reasonable alg for Some problems (eg matrix multiplication, sorting)

Weaknesses

(a) Rarely yields efficient alg.
(b) Some brute-force algs are unacceptably slow not as constructive as Some other design techniques.

Scanned by TapScanner

PART-B

(11) Explain convex – Hull problems by Brute force Method.

Definition of Convex Set

* A set of points (finite or infinite) in the plane is called convex. if for any two points P and Q in the set, the entire line segment with the end points at P and Q belongs to the set.

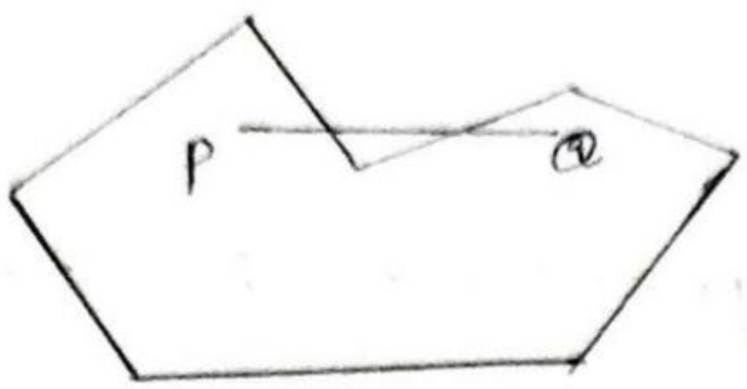
Definition of Convex Hull.

* The Convex Hull of a set S of points is the Smallest convex set containing S.

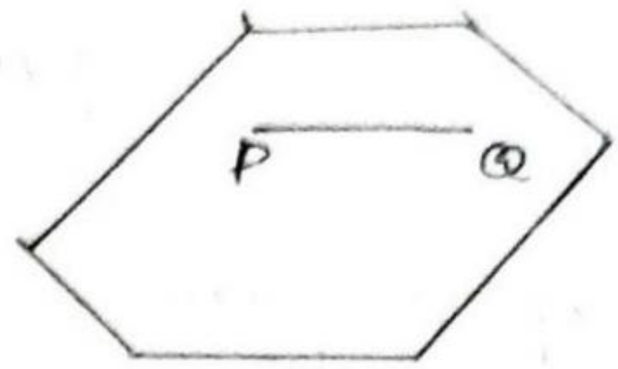
Explanation of Convex Hull.

* To explain what a convex hull is, A subset S of a space is tilled convex if and only if for any pair of points P and Q in the set S, the line section from P and Q is enclosed completely in the set S.

Scanned by TapScanner



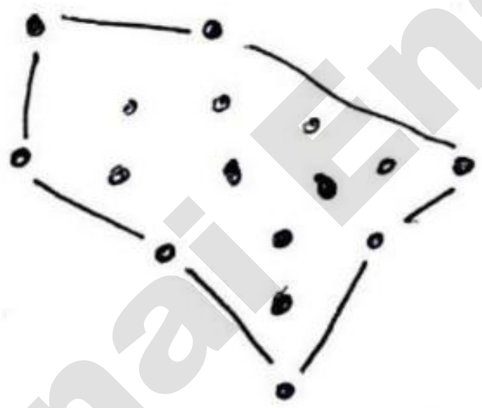
(a) Non - Convex



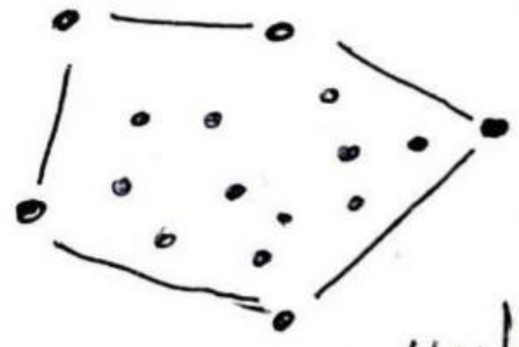
(b) Convex

→ With this definition now we can explain what Convex hull means, imagine in a two dimensional space there exists a finite set of points $k = \{k_1, k_2 \ldots k_n\}$ on a cartesian plane.

→ The set k Contains a subset of points $L = \{l_1, l_2, \ldots l_n\}$ where $2 \leq L \leq K$, then L is the smallest set of points that embrace all the points in k.



(a) A hull, but not convex



(b) Convex Hull

→ The image in the left is not a convex hull because although all the points touching the edges embrace all the points in the entire set.

Scanned by TapScanner

⇒ This set of points cannot be a convex hull because there are line segments between points that lie outside the outer boundary.

⇒ The image on the right, on the other hand, is a convex hull because all line segments between points lie entirely in the boundary.

Convex-hull problem

* The convex-hull problem is the problem of constructing the convex hull for a given set S of n points.

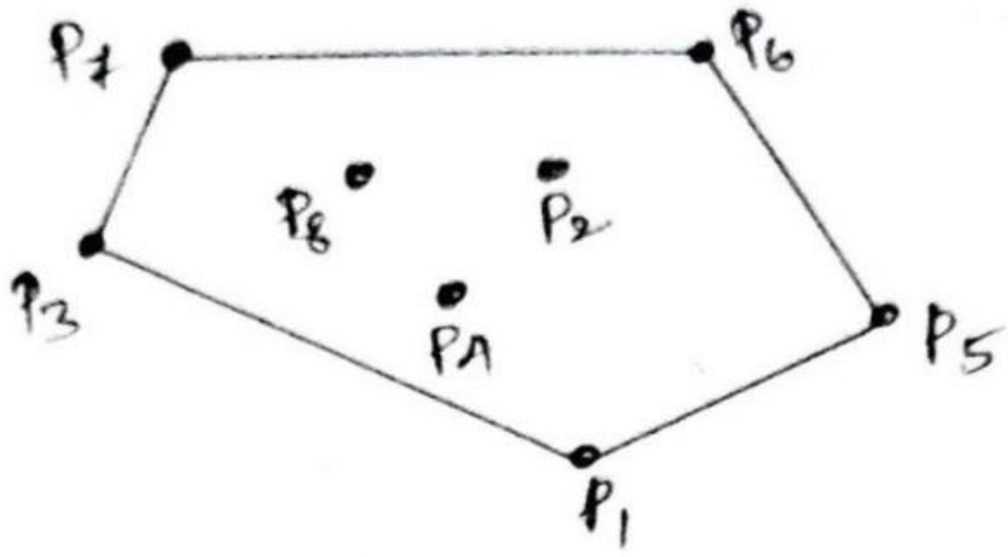
* To solve it, we need to find the points that will serve as the vertices of the polygon in question.

* Vertices of such a polygon as 'extreme points'.

for example

The extreme points of a triangle are its three vertices, the extreme points of a circle are all the points of its circumference and the extreme points of the convex hull of the set of eight points in below fig P_1, P_5, P_6, P_4 and P_3.

Scanned by TapScanner



Properties:

1. The hull is a cycle graph whose vertices are composed of a subset of the point in set S.
2. No points in S lie outside the graph.
3. All interior angles in the graph are less than 180 degrees.

Brute force Algorithm for convex hull.

PLAIN TEXT CODE

1. for all points p in S
2. for all point q in S
3. if p!=q
4. .draw a line from p to q
5. if all points in S except p and q lie to the left of the line.
6. add the directed vector pq to the solution set.

Scanned by TapScanner

(2) Explain Merge Sort Algorithm with an example (Apr|May'18) (Nov|Dec'17) (May|June'16)

Merge Sort

The strategy:

* Mergesort divides the array into two equal halves and sorts the halves separately (using recursion), then it merges the sorted halves.
* The parameters used for merge are, the array name E and the first, mid and last indexes of the subranges it is to merge.
  (ie) 2 sorted subranges are E[first]....E[mid], E[mid+1]...E[last] (after recursion end the final sorted array is E[first]...E[last].

Example

Given

The array of 5 elements E[1:5] = (310, 285, 179, 652, 351)

① Mergesort Splits E[1:5] into two sub arrays each of size (ie) E[1:3] and E[4:5]

(310, 285, 179 | 652, 351)

② Then the items in E[1:3] are split into two subarrays of size E[1:2] & E[3:3]

(310, 285 | 179 | 652, 351)

Scanned by TapScanner

3. The two values E[1:2] are split at a final time into one-element subarrays and now merging begins.

(310 | 285 | 179 | 652, 351)

A. Elements E[1] and E[2] are merged

(285, 310 | 179 | 652, 351)

5. Elements E[3] is merged with E[1:2]

(179, 285, 310 | 652, 351) – Returns the first recursive call and is about to process 2nd recursive call.

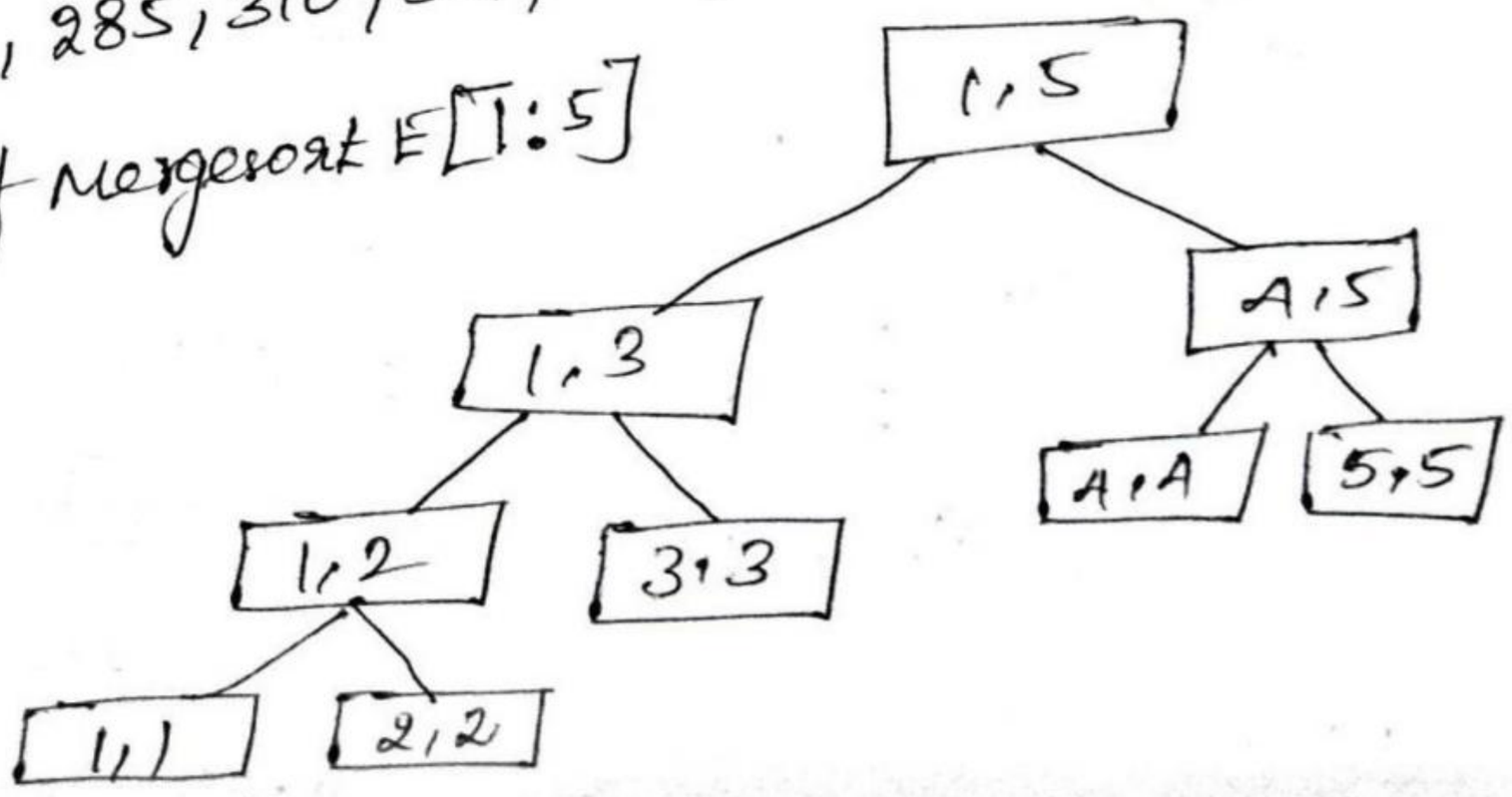
6. Elements E[4] and E[5] are divided and merged

(179, 285, 310 | 351, 652]

7. Merge E[1:3] with E[4:5], resulting a sorted array of n numbers.

(179, 285, 310, 351, 652)

Tree of calls of Mergesort E[1:5]



Scanned by TapScanner

Algorithm : Mergesort

Input : Array E and Indexes first and last, such that the elements of E[i..j] are defined for first ≤ i ≤ last.

output : E[first] ... E[last] is a sorted rearrangement of the same elements.

```
Void mergeSort (Element [] E, int first, int last)
if (first < last)
int mid = (first + last)/2;
mergeSort (E, first, mid);
mergeSort (E, mid+1, last);
merge (E, first, mid, last);
return;
```

Algorithm: Merge

```
{ I/p : Array E & indexes first, mid & last
  o/p : Array E, ranges from index first to last
Void merge (Element [] E, int first, int mid, int last)
int i, j, k, b[];
  i = first;
  j = mid+1;
  k = first,
while ((i ≤ mid) && (j ≤ last))
if (E[i].key < E[j].key)
  b[k] = E[i];
  k++;
  i++;
else
  b[k] = E[j];
  k++;
  j++;
//Continue loop
if (i > mid)
  Copy E[i ... mid] to b[k .... last]
  Copy b[first .... last] to E[first ... last]
return.
```


Scanned by TapScanner

Analysis

⇒ Assume n is a power of 2, the recurrence relation for the number of key comparisons $C(n)$ is;

$$C(n) = C(n/2) + C(n/2) + C_{merge}(n) \quad \text{for } n>1$$

In general $C(1)=0$

⇒ $C_{merge}(n)$ — No. of. key comparisons done in merging

⇒ In the worst case, each step, exactly one comparison is done (ie) $(n-1)$

$$C_{worst}(n) = 2C(n/2) + n-1 \quad \text{for } n>1, \; C_{worst}(1)=0$$

Using Master's Theorem:

$$\boxed{T(n) = aT(n/b) + f(n)}$$

Here $a=2, b=2$ and to find the value of d;

$$f(n) \in \Theta(n) \in \Theta(n^d)$$

$\therefore d=1$

Here $a = b^d$ (ie) $2 = 2^1$

$\therefore C_{worst}(n) \in \Theta(n^d \log n) \in \Theta(n^1 \log n)$

Space Usage:

* MergeSort requires auxiliary workspace for merging operation, which is $\Theta(n)$ mergesort is not an in place sort.

Scanned by TapScanner

(3) Write the quicksort algorithm and explain it with an example. Derive the worst case & avg case time complexity. {(Nov/Dec'16) (Nov/Dec'18), (Apr/may'19)}

Quick Sort

Quick Sort is one of the earlier divide and conquer algorithm. → In this ~~step~~ method division is carried out dynamically.

→ 3 steps of quick sort are,

① Divide : Split the array into 2 subarrays that each elements in the left sub array is less than or equal to the middle element and each element in the right sub array is greater than the middle element. splitting is based on the pivot element.

② Conquer : Recursively sort the 2 subarrays

③ Combine : Combine all the sorted elements in a group of to form a list of sorted elements.

MergSort : Division of array is based on the positions of array elements.

QuickSort : Division is based on the actual value of the element.

Scanned by TapScanner

Given:

E - Array of elements n - Number of elements

first & last - Indexes of the first and last entries.

(ii) first = 0

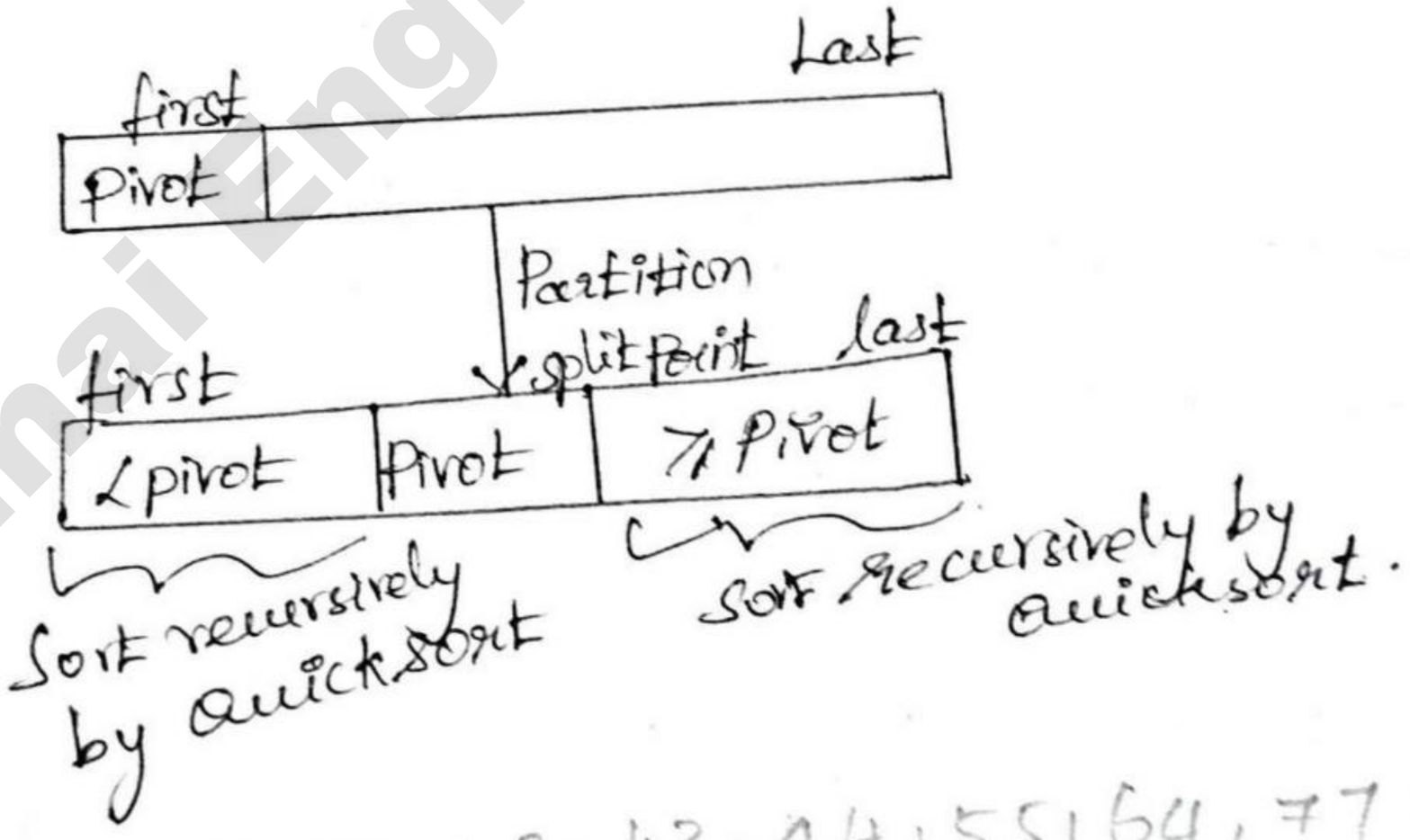
last = n-1

⇒ QuickSort algorithm chooses an key element, called the pivot element. (ii) the leftmost element in the subrange is moved to a local variable leaving a vacancy in the array.

⇒ QuickSort passes the pivot to the partition subroutine which rearranges the other elements, finding an index splitPoint such that

i) first ≤ i < splitPoint, E[i].key < pivot

ii) splitPoint < i ≤ last, E[i].key ≥ pivot.



12, 33, 23, 43, 44, 55, 64, 77 and 76

Scanned by TapScanner

Example:

45 1A 62 51 75 96 33 8A 20

Pivot

* Pivot element = 45, Vacant is created in that position

— 1A 62 51 75 96 33 8A 20 ←

* Search from backward to find < 45

20 1A 62 51 75 96 33 8A — →

* Search from the forward to find >= 45

20 1A — 51 75 96 33 8A 62 ←

* Process continues to find correct position for the pivot element = 4.

20 1A 33 51 75 96 — 8A 62.

→

20 1A 33 — 75 96 51 8A 62 ←

20 1A 33 (45) 75 96 51 8A 62.

⇒ pivot element is stored in the location of E[SplitPoint], which divides the array into two sub ranges.

Partition of first section: (before pivot is <45)

Pivot → (20) 1A 33

←

Scanned by TapScanner

Bnd. Position

— 1A 33 Pivot : 20 (∵ Pivot > element)

←

1A — 33

→

1A 20 33

Partition of second section : (After pivot (20) ≯45)

pivot → (75) 96 51 8A 62

←

62 96 51 8A —

→

62 — 51 8A 96

←

62 51 — 8A 96

62 51 (75) 8A 96

Partition of left subdivision

(before pivot (62) <75)

(62) 51

← pivot

51 62

Partion of right subsection

8A 96.

Sorted array sequence.

1A 20 33 — 1st section. 45 — Split point. 51 62 — left subsection. 75 — Split point. 8A 96 — Right subsection

Algorithm: QuickSort I/P : Array E and indexes first and last such that elements E[i] are defined for first ≤ i ≤ last.

O/P : E[first] ... E[last] is a sorted rearrangement of the same elements.

Scanned by TapScanner

```
Void quickSort ( Element [] E , int first , int last)
 if (first < last)
   Element pivotElement = E[first];
   key pivot = pivotElement.key;
   int splitPoint = partition (E, pivot , first , last);
   E[splitPoint] = pivotElement ;
   quickSort (E, first , splitPoint -1);
   quickSort (E, splitPoint +1, last);
```

Algorithm : Partition :

```
int partition (Element [] E, key ~~Point~~ Pivot , int first , int last)
 int low, high ;
 low = first ; high = last;
 while (low < high)
  int highvac = extendLargeregion (E, pivot , low, high);
  int lowvac = extendSmallregion (E, pivot , low+1, highvac);
  low = lowvac;
  high = highvac -1 ;
  return low; // This is like splitPoint.
// * PostCondition for extendLargeregion :
 int extendLargeregion (Element[] E, key pivot ;
                        int lowvac , int high)
 int highvac , curr;
  highvac = lowvac ;
   curr = high;
```


Scanned by TapScanner

```
while (curr > lowvac)
        if (E[curr].key < pivot)
                E[lowvac] = E[curr];
                highvac = curr;
                break;
                curr--;
        return highvac;
```

```
/* postcondition for extend small region.
int extendsmallregion (Element[] E, key pivot, int low,
                                        int highvac)
int lowvac, curr;
lowvac = highvac;
curr = low;
while (curr < highvac)
if (E[curr].key >= pivot)
E[highvac] = E[curr];
        lowvac = curr;
        break;
        curr++;
return lowvac;
```

Analysis : Worst Case

1. If there are k positions in the range of the array, partition does k-1 key comparison.
2. If E[first] has the smallest key then splitPoint = first, dividing the range into an empty subrange and subrange with k-1 elements.

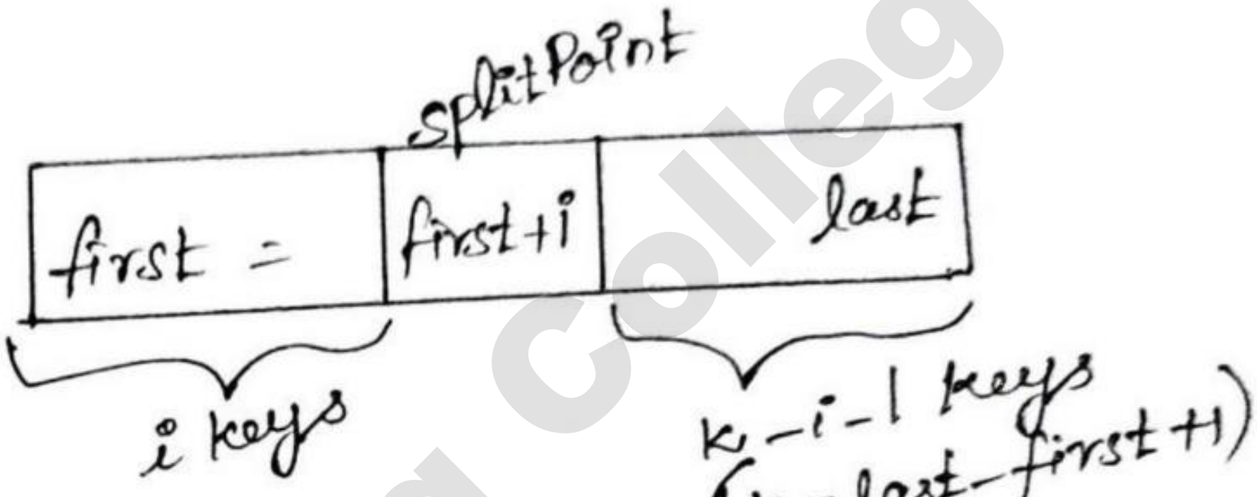
Scanned by TapScanner

3. If pivot is the smallest key each time partition is called then total no. of key comparison alone is

$$\sum_{k=2}^{n} (k-1) = \frac{n(n-1)}{2} \quad \left\{ 1+2+3 \ldots n = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n \right\} = n^2/2$$

4. A conflict type of coorst case is when the keys are already sorted in ascending order.

Average case



⇒ from the structure we can define 2 subranges

(a) first ... splitPoint -1

(b) splitPoint +1 ... last

⇒ Recurrence equation, from k elements is = 1/k

Let k=n

$$A(n) = n-1 + \sum_{i=0}^{n-1} \frac{1}{n} (A(i) + A(n-1-i)) \text{ for } n \geq 2$$

$$A(1) = A(0) = 0$$

⇒ A(n-1-i) runs from A(n-1) down to A(0). So, the sum of A(n-1-i) is equal to A(i) terms,

$$A(n) = n-1 + \frac{2}{n} \sum_{i=1}^{n-1} A(i) \text{ for } n \geq 1$$

⇒ A(n) for quicksort ⇒ $\Theta(n) = n + 2\Theta(n/2)$

$C_{best}(n)$	$= n \log_2 n$
$C_{avg}(n)$	$= 1.38 n \log_2 n$
$C_{worst}(n)$	$= \frac{n(n-1)}{2} \simeq \frac{n^2}{2}$

Scanned by TapScanner

(4) Explain the working of strassen's Matrix Multiplication with the help of divide and Conquer Method ? ((Apr/May '18) (Nov/Dec '15))

General Matrix Multiplication :

⇒ The time complixity of its number of multiplications for matrix multiplication is by $T(n) = n^3$, where n is the no. of. rows and columns in the matrices. We can analyse the no. of. additions.

⇒ The time complexity excluding no. of. additions is given by $T(n) = n^3 - n^2$. Hence the time complexity of matrix multiplication is in $O(n^3)$.

1. The divide and conquer approach can reduce the no. of. multiplications.

2. The product of two 2×2 matrices, A and B.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & C_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

using divide & conquer.

$$\begin{array}{c} \leftrightarrow n/2 \\ n/2 \updownarrow \end{array} \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \times \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right]$$

Scanned by TapScanner

Where,

$$A_{11} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,n/2} \\ a_{21} & a_{22} & \cdots & a_{2,n/2} \\ \vdots & & & \\ a_{n/2,1} & \cdots & \cdots & a_{n/2,n/2} \end{bmatrix}$$ is a $n/2 * n/2$ matrix

⇒ The Complexity analysis is as follows

1. I/p size : Matrix n*n
2. Basic operation : Multiplication / Addition
3. Number of multiplication is

$$C_{11} = (A_{11} * B_{11}) + (A_{12} + B_{21})$$
$$C_{12} = (A_{11} * B_{12}) + (A_{12} * B_{22})$$
$$C_{21} = (A_{21} * B_{11}) + (A_{22} * B_{21})$$
$$C_{22} = (A_{21} * B_{12}) + (A_{22} * B_{22})$$

So totally 8 times of $n/2 * n/2$ multiplications and 4 times of $n^2/4$ additions are performed.

$$T(n) = 8T(n/2) + 4T\left(\frac{n^2}{4}\right)$$

Applying Master's theorem, the complexity is $O(n^3)$

Strassen's Matrix Multiplication :

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} * \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Scanned by TapScanner

Strassen determinant :

$$m_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$

$$m_2 = (a_{21} + a_{22})\, b_{11}$$

$$m_3 = a_{11}(b_{12} - b_{22})$$

$$m_4 = a_{22}(b_{21} - b_{11})$$

$$m_5 = (a_{11} + a_{12})\, b_{22}$$

$$m_6 = (a_{21} - a_{11})(b_{11} + b_{12})$$

$$m_7 = (a_{12} - a_{22})(b_{21} + b_{22})$$

The product C is given by,

$$C = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

⇒ To multiply two 2×2 matrices, Strassen's method requires seven multiplications and 18 additions/subtractions, whereas the straight-forward divide and conquer method mentioned requires eight multiplications and four additions/subtractions.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

(n/2 × n/2 blocks)

AEC

Complexity analysis:

Ip size : n, the no. of rows and columns in the matrices.

Basic operation : One elementary multiplication.

$$T(n) = 7T\left[\frac{n}{2}\right] + 18\left[\frac{n}{2}\right]^2 \quad \text{for } n = 2^k, \text{ n a power of 2}$$

$$T(1) = 0$$

Solving the recurrance eqn,

$$M(n) = 7M(n/2) \text{ for } n \geq 1, \; M(1) = 1$$

since $n = 2^k$,

$$M(2^k) = 7M(2^{k-1}) = 7[7M(2^{k-2})] = 7^2 M(2^{k-2}) = \cdots$$

$$= 7^i M(2^{k-i}) \cdots = 7^k M(2^{k-k}) = 7^k$$

$$M(n) = 7^{\log_2 n} = n^{\log_2 7} \simeq n^{2.807}.$$

eg. PBM: Multiply the following Matrix.

$$\begin{bmatrix} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{bmatrix}$$

Using Divide and Conquer Strassen's technique:

$$C = \begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{bmatrix}$$

Scanned by TapScanner

Where,

$$A_{00} = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}, A_{01} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}, A_{10} = \begin{bmatrix} 0 & 1 \\ 5 & 0 \end{bmatrix}, A_{11} = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

$$B_{00} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}, B_{01} = \begin{bmatrix} 0 & 1 \\ 0 & 4 \end{bmatrix}, B_{10} = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}, B_{11} = \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix}$$

$$M_1 = (A_{10} + A_{11})\ (B_{00} + B_{11}) = \begin{bmatrix} 4 & 0 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix},$$

$$M_2 = (A_{10} + A_{11})\ B_{00} = \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix},$$

$$M_3 = A_{00}\ (B_{01} - B_{11}) = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -5 & 4 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix},$$

$$M_4 = A_{11}\ (B_{10} - B_{00}) = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix},$$

$$M_5 = (A_{00} + A_{01})\ B_{11} = \begin{bmatrix} 3 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 5 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix},$$

$$M_6 = (A_{10} - A_{00})\ (B_{00} + B_{01}) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}$$

$$M_7 = (A_{01} - A_{11})\ (B_{10} + B_{11}) = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$C_{00} = M_1 + M_4 - M_5 + M_7$$

$$= \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

Scanned by TapScanner

$C_{01} = M_3 + M_5$

$= \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} + \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix} = \begin{bmatrix} 7 & 3 \\ 1 & 9 \end{bmatrix}$,

$C_{10} = M_2 + M_4$

$= \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ 5 & 8 \end{bmatrix}$,

$C_{11} = M_1 + M_3 - M_2 + M_6$

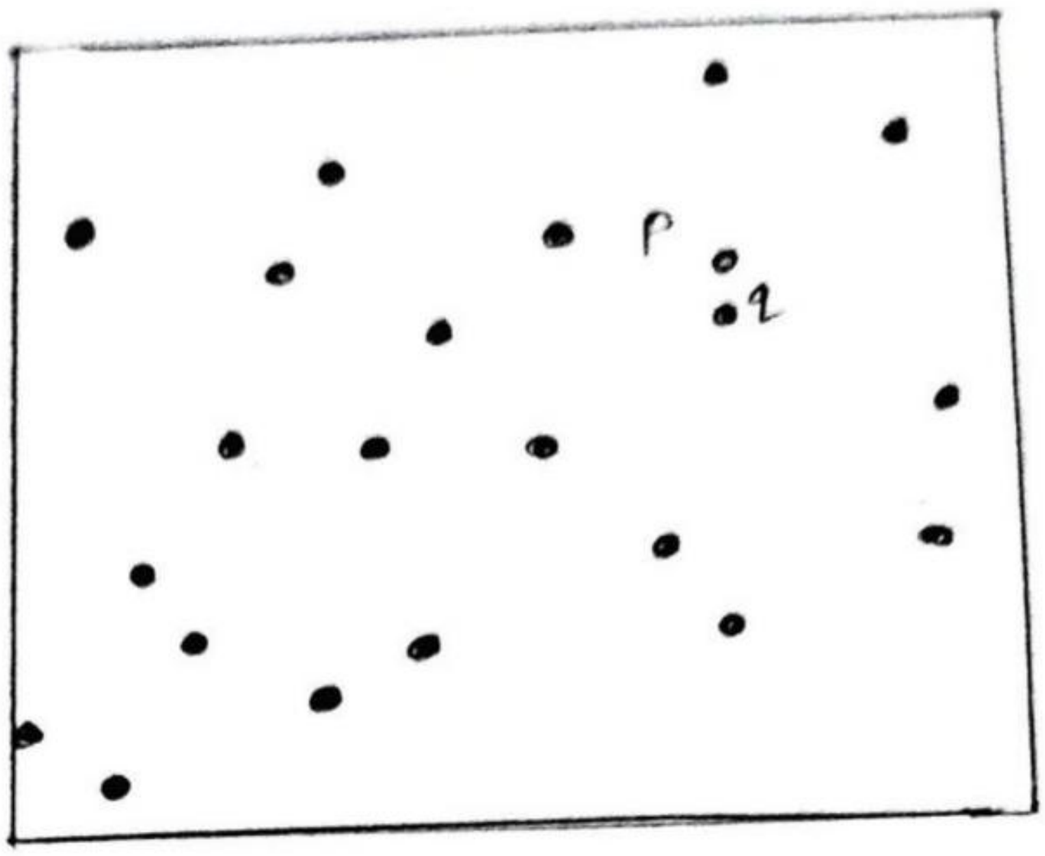
$= \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix} - \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ 7 & 7 \end{bmatrix}$

$$C = \begin{bmatrix} 5 & 4 & 7 & 3 \\ 4 & 5 & 1 & 9 \\ 8 & 1 & 3 & 7 \\ 5 & 8 & 7 & 7 \end{bmatrix}$$

(5) Write the algorithm to find the closest pair of point using divide and conquer and explain it with an example. Derive the worst case and average case time complexity. (Nov/Dec'19) (Nov/Dec'15)

* The closest pair problem is to find the two closest points in a set of n points.

Scanned by TapScanner



⇒ It is used in computational geometry that deals with proximity of points in the plane or higher dimensional spaces.

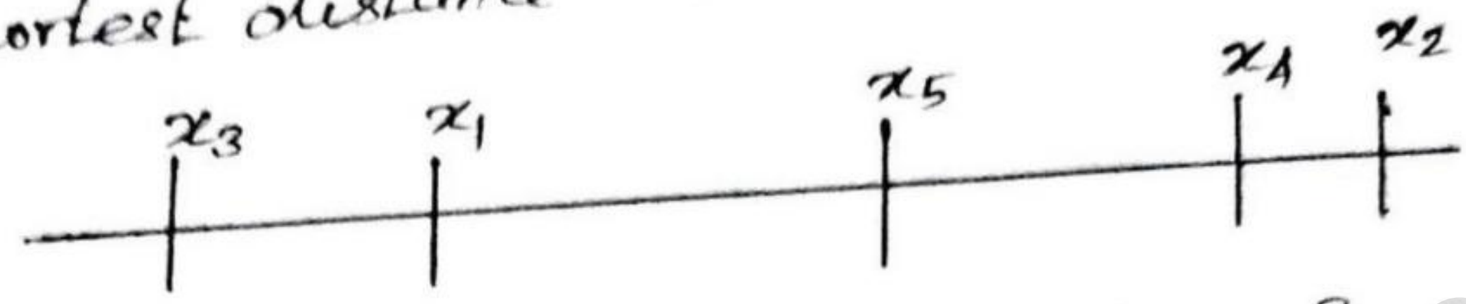
⇒ The points in the cartesian plane. the point are ordered using efficient sorting algorithm in nondecreasing order of their x co-ordinate.

⇒ It will also be convenient to have the points sorted in a seperate list in nondecreasing order of the y coordinate.

Scanned by TapScanner

One dimensional case of closest pair problem.

⇒ One Dimensional problem can be solved in O(n log n) via sorting (x_1, x_2, x_3, x_4, x_5) and finding the shortest distance between two consecutive points.



Two dimensional case of the closest pair problem:

⇒ If $2 \le n \le 3$, the problem can be solved by the obvious brute-force alg.

⇒ If $n > 3$, we can divide the points into two subsets P_1 and P_2 of $n/2$ and $n/2$ points, resp.

⇒ By drawing a vertical line through the median m of their x coordinates so that $n/2$ points lie to the left of or on the line itself and $n/2$ points lie to the right of or on the line.

⇒ Then we can solve the closest pair problem recursively for subsets P_1 and P_2.

⇒ Let d_1 and d_r be the smallest distance between pairs of points in P_1 and P_r resp. & let $d = \min\{d_1, d_2\}$

Scanned by TapScanner

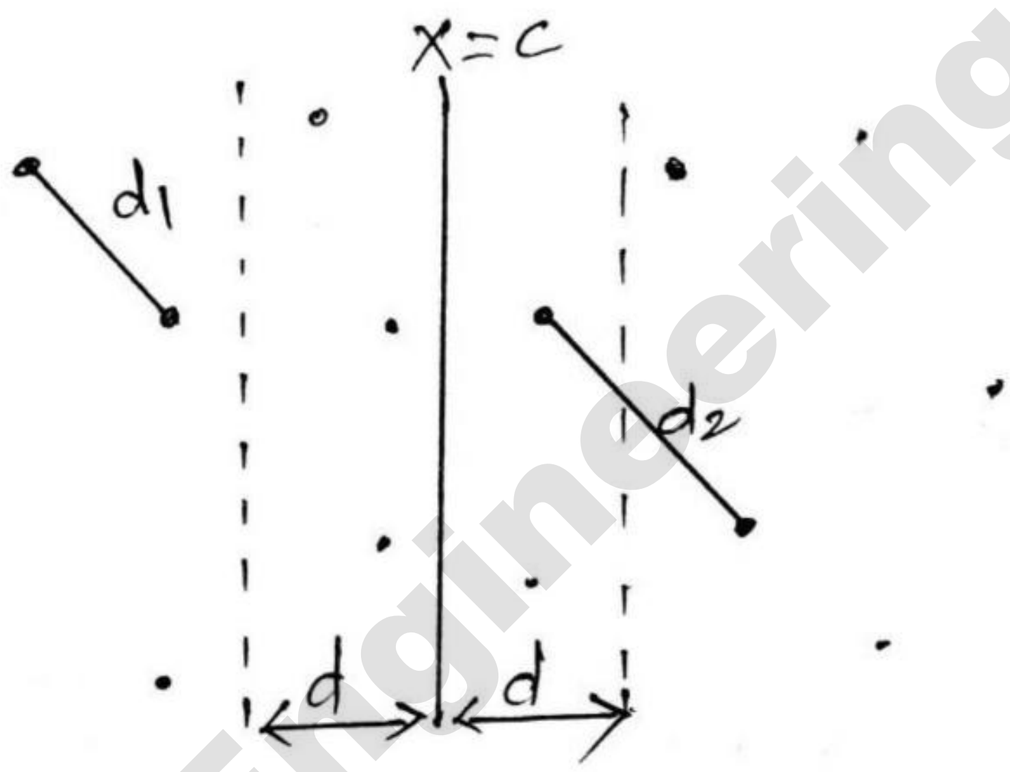
Algorithm

I/p : A set S of n planar points.

O/p : The distance between two closest points.

Step 1:

Divide the points given into two subsets S_1 and S_2 by a vertical line $x = c$ so that half the points lie to the left or on the line and half the points lie to the right or on the line.



Step 2: Find recursively the closest pairs for the left and right subsets.

Step 3: Set $d = \min\{d_1, d_2\}$

Step 4: For every point $P(x,y)$ in C_1, we inspect points in C_2 that may be closer to p than d. There can be no more than 6 such points.

Scanned by TapScanner

Algorithm:

```
Efficient ClosestPair (P,Q)
// Solves the closest-pair problem by divide and conquer
// I/p: An array P of n >= 2 points in the Cartesian plane
        sorted in,
// non decreasing order of their x coordinates and
   an array Q of the same points sorted in non
   decreasing order of the y coordinates.
// o/p: Euclidean distance between the closest pair
        of points.
if n <= 3
   return the minimal distance found by the brute
force algorithm.
else
   Copy the first ⌈n/2⌉ points of P to array P1
   Copy the same ⌈n/2⌉ points from Q to array Q1
   Copy the remaining ⌊n/2⌋ points of P to array Pr
   Copy the same ⌊n/2⌋ points from Q to array Qr.
   d1 ← EfficientClosestPair (P1, Q1)
   dr ← EfficientClosestPair (Pr, Qr).
```


Scanned by TapScanner

$d \leftarrow \min\{d_1, d_r\}$

$m \leftarrow P[\lceil n/2 \rceil - 1].x$

copy all the points of Q for which $|x - m| < d$ into array $S[0..num-1]$

$dminsq \leftarrow d^2$

for $i \leftarrow 0$ to $num - 2$ do

$k \leftarrow i+1$

while $k \leq num - 1$ and $(S[k].y - S[i].y)^2 < dminsq$

$dminsq \leftarrow \min((S[k].x - S[k].x)^2 + (S[k].y - S[i].y)^2, dminsq)$

$k \leftarrow k+1$

return sqrt(dminsq)

Analysis

Running time of the algorithm is described by

$T(n) = 2T(n/2) + M(n)$, where $M(n) \in O(n)$

By the Master Theorem (with $a=2$, $b=2$, $d=1$)

$T(n) \in O(n \log n)$

Scanned by TapScanner

UNIT - III

1) what is Brute force method?

Brute force is the simplest technique of the design strategies. Brute force is a straight forward approach to solving a problem, usually directly based on the problem's statement and definitions of the concepts involved.

2) Define a binary search tree.

⇒ A binary Search tree is one of the important data structure and its application is to implement a dictionary.

⇒ To increase the search efficiency more frequently used words are arranged nearer to the root and less frequently used words away from the root, with the help of probability value of each word. This type of arrangement is called binary search tree.

3) State the principle of optimality.

⇒ It is the basic principle of dynamic programming.

⇒ The principle says, that an optimal path has the property that whatever. The initial conditions & control variables over some initial period, the control chosen over the remaining period must be optimal for the remaining pbm with the state resulting from the early decisions taken to the initial condition.

Scanned by TapScanner

4) what is the constraint for binary search tree insertion?

* A binary search tree is a tree, with one additional constraint — it keeps the element in the tree in a particular order.

* Formally each node in the BST has two children, a left child and Right child.

5) Define multistage Graph.

* A multistage graph is a directed graph in which the nodes can be divided into a set of stages such that all edges are from a stage to next stage only.

⇒ In other words there is no edge between vertices of same stage and from a vertex of current stage to previous stage.

6) How dynamic programming is used to solve knapsack problem?

* The knapsack problem states that given n items of known weights $w_1, w_2 \ldots w_n$ and values $v_1, v_2 \ldots v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.

* In dynamic programming we need to obtain the solution by solving the smaller subinstances.

Scanned by TapScanner

7) Define the minimum spanning tree problem.

* A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted (un) directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

8) What does Flogd's algorithm do?

* It is used for computing shortest path between every pair of vertices of graph. The graph may contains negative edges but should not contain negative cycles.

9) State the general principle of greedy algorithm.

* The general principle of greedy algorithm is Select an input at each stage which derives a feasible solution then it is added to the optimal solution until the problem terminates with a condition.

10) What do you mean by dynamic programming?

* It is a technique for solving problems with overlapping sub problems.

* The smaller sub problems are solved only once and recording the results in a table from which the solution to the original pbm is obtained. eg) Knapsack.

Scanned by TapScanner

11) How to calculate the efficiency of Dijkstra's Algorithm.

⇒ The efficiency of dijikstra's algorithm varies depending on $|V|=n$. Determines and $|E|$ updates for the priority Queues that were used.

⇒ It is expressed in terms of Big O notation as $O(E \log n)$

12) Define the single source shortest paths problem.

⇒ It is to find a shortest path from a given source r to every other vertex.

⇒ $V \in V-(r)$. The weight of a path.

⇒ $P=V_0, V_1 \ldots V_k$ is the sum of the weights of its constituent edges.

$$W(P) = \sum_{i=1}^{k} w(V_{i-1}, V_i)$$

$$S(u,v) = \min(w(p)) : P \text{ is a path from } u \text{ to } V.$$

13) List the advantage of greedy algorithm.

* Greedy algorithm produces a feasible solution
* Greedy method is very simple to solve a pbm.
* Greedy method provides an optimal solution directly.

14) Mention the applications of Minimum Spanning Tree?

⇒ Spanning Tree are used to obtain an independent set of circuit equations for an electric network.

Scanned by TapScanner

⇒ Another application of spanning tree arises from the property that a spanning tree is a minimal subgraph G of G such that $V(G') = V(G)$ and G' is connected.

15) Illustrate any two characteristics of Greedy Algorithm.

a) To solve a problem in an optimal way construct the solution from given set of candidates

b) As the algorithm proceeds, two other sets get accumulated among this one set contains the candidate that have been already considered and chosen while the other set contains the candidates that have been considered but rejected.

16) Show the general procedure of dynamic programming.

1. Characterize the structure of an optimal solution
2. Recursively define the value of the optimal sol.
3. Compute the value of an optimal solution in the bottom-up fashion.

17) Define Warshall's algorithm.

*Warshall's algorithm is an application of dynamic programming technique, which is used to find the transitive closure of a directed graph.

Scanned by TapScanner

18) Define Kruskal's Algorithm

⇒ kruskal's algorithm is another greedy alg for the minimum spanning tree problem.

⇒ kruskal's algorithm constructed a minimum spanning tree by selecting edges in increasing order of their weights provided that the inclusion does not create a cycle.

⇒ kreskal algorithm provides a optimal solution.

19) List the advantage of Huffman's encoding.

A) Huffman's encoding is one of the most important file compression method.

b) It is simple

c) It is versatility

d) It provides optimal and minimum length encoding.

20) State the time efficiency of floyd's algorithm.

O(n3). It is cubic.

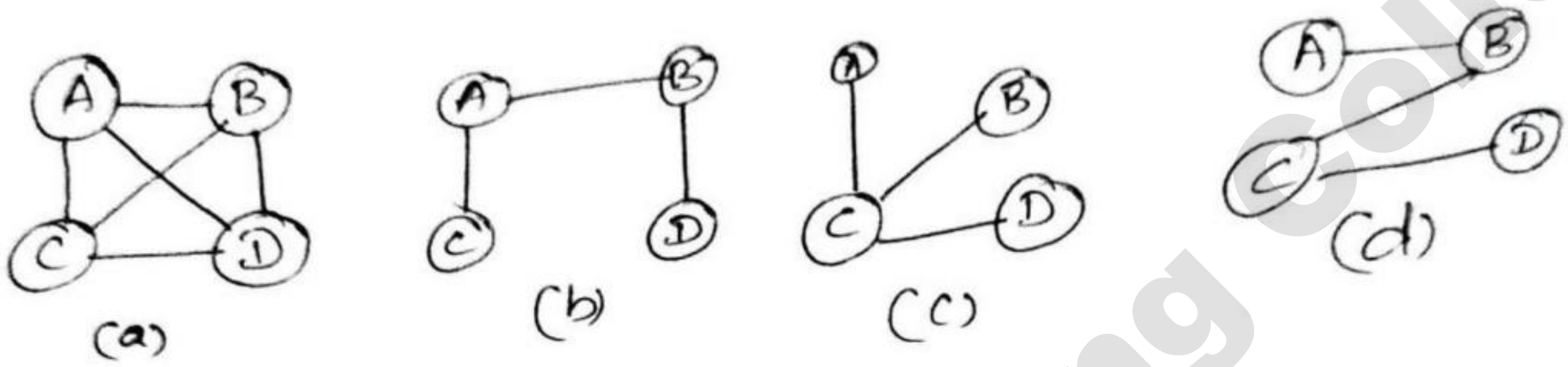
Scanned by TapScanner

① Explain Prim's alg (or) Give the pseudo code for prim's alg & apply the same to find the MST (NOV/DEC'19) {(APR/MAY'18)(NOV/DEC'17)}

PRIM'S ALGORITHM

Definition: Minimum Spanning Tree (MST) {should not allow the circuit}

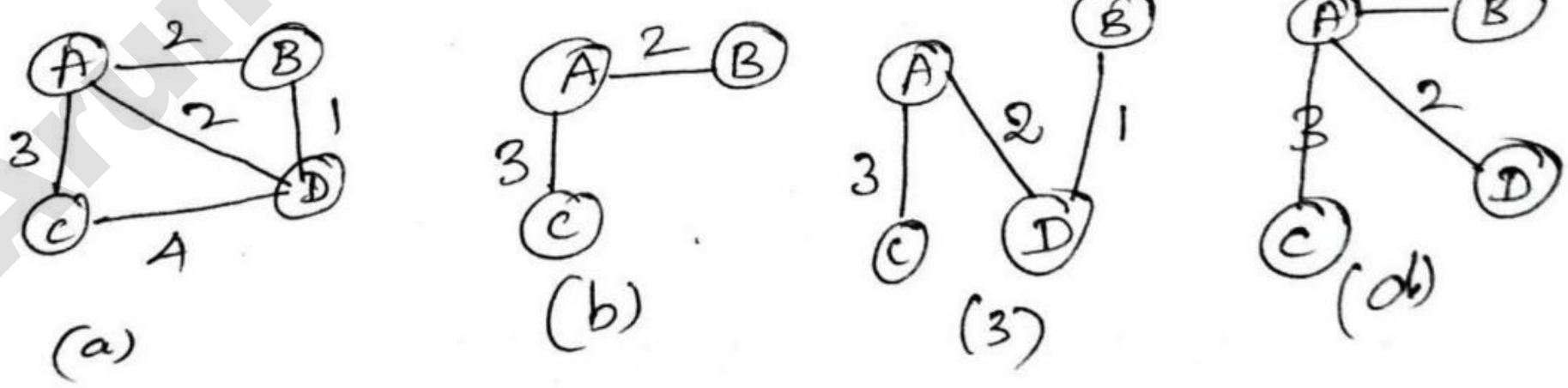
⇒ A spanning tree for a connected undirected graph $G = (V, E)$ is a subgraph $t = (V, E')$ if t is a tree.



(a) $G = (V, E)$ (b), (c) & (d) — Some example for spanning tree.

⇒ In a weighted Graph $G = (V, E, W)$, the weight of a subgraph is the sum of the weights of the edges in the subgraph.

⇒ A MST for a weighted Graph is a spanning tree with minimum weight.



(a) $G = (V, E, W)$

(b) and (c) are minimum spanning trees ($w=6$) when comparing to (d) ($w=7$).

Scanned by TapScanner

The strategy

1. Prim's algorithm selects a starting vertex from the given graph and classifies the start vertex under "tree vertices".
2. The nodes adjacent to tree vertices are identified and classified under "fringe vertices", & the remaining vertices are classified as "unseen vertices".
3. The key step of the alg is selection of new vertex from the "fringe" and an incident edge, which is now added to "tree vertices".

* After this new inclusion again the "fringe vertices" and unseen vertices are reclassified.
* The selection of new vertex from the "fringe" is depends on the weight of the edge. This process continues until the "fringe" is empty.

Algorithm

PrimMST (G, in) // outline

Initialise all vertices as unseen.

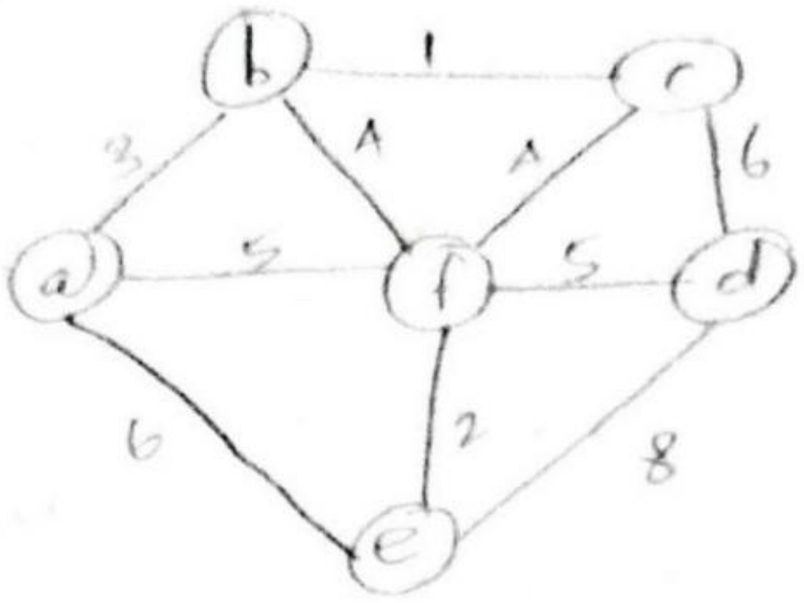
Select an arbitrary vertex s to start the tree; reclassify it as tree. Reclassify all vertices adjacent to s as fringe.

while there are fringe vertices.

Select an edge of minimum weight between a tree vertex t and a fringe vertex v;

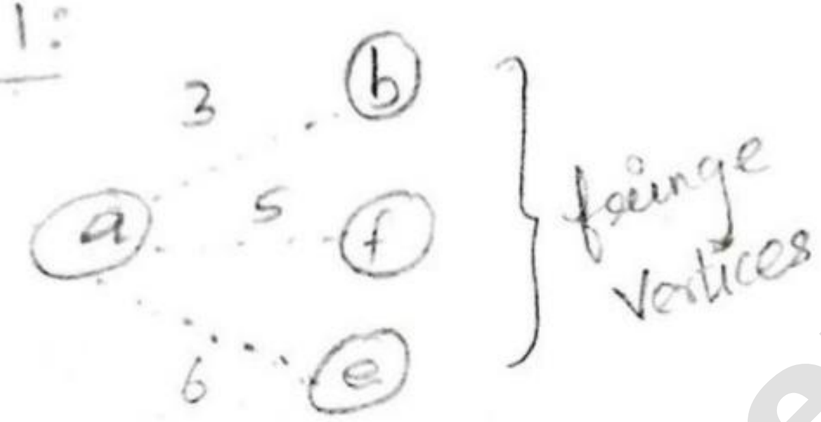
Scanned by TapScanner

Reclassify v as tree ; add edge to v to the tree ;
Reclassify all unseen vertices adjacent to v as fringe.

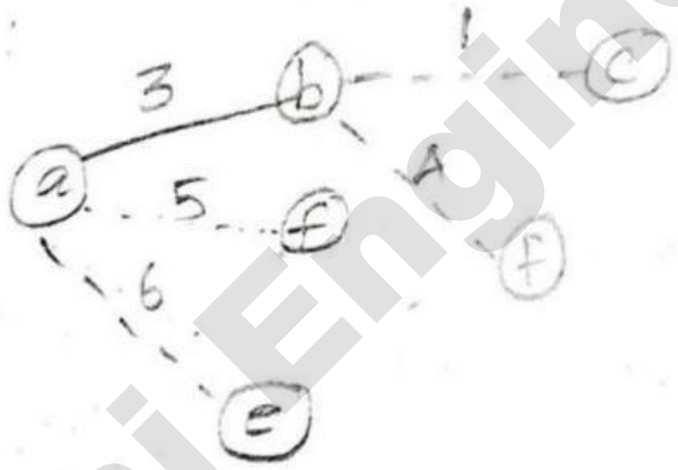


Node
Solid lines – Tree edges
Dotted lines – Edges of fringe vertices

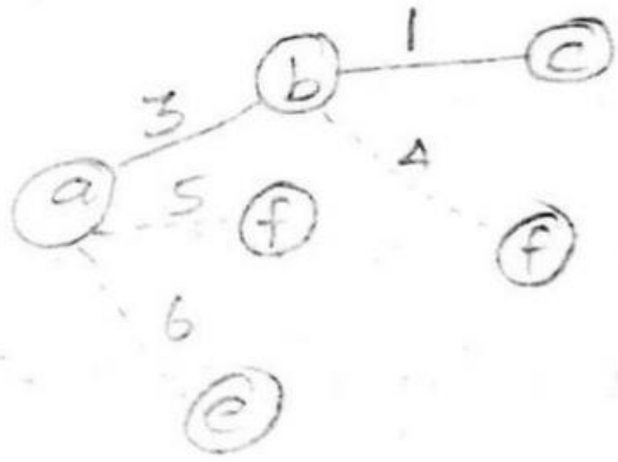
Step 1:



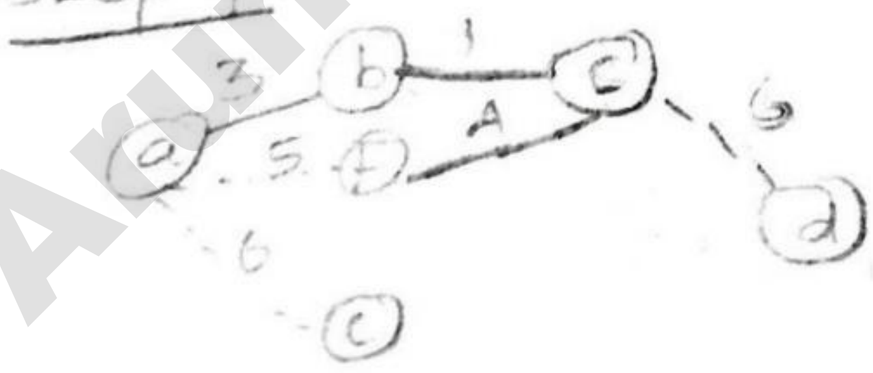
Step 2:



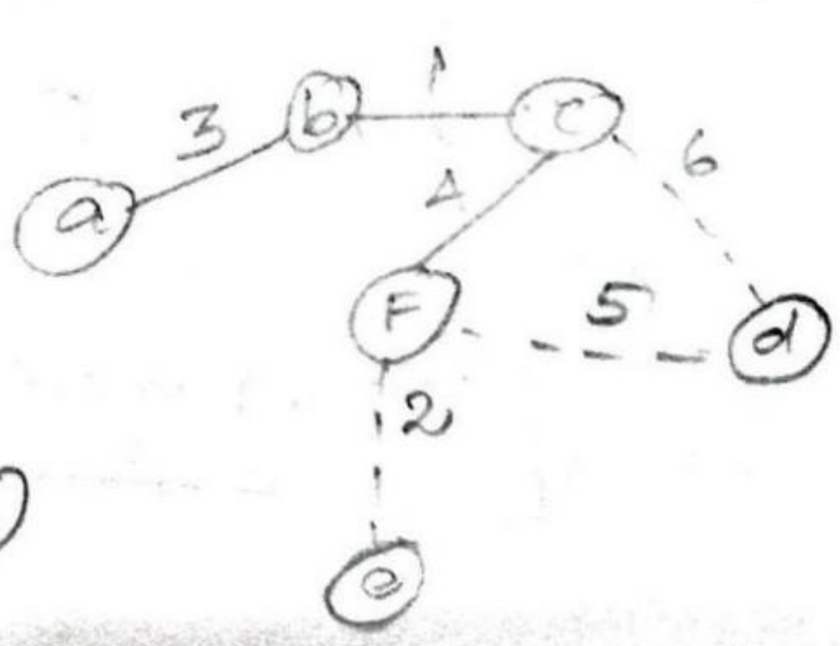
Step 3



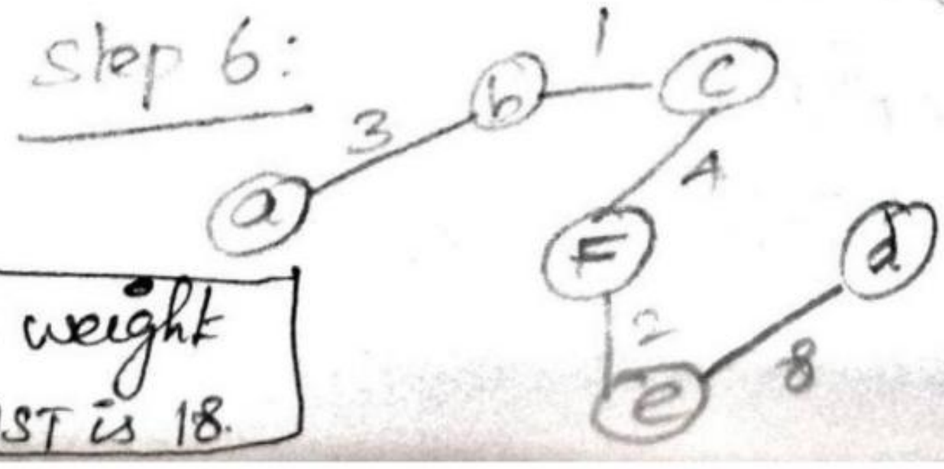
Step 4



Step 5



Step 6:



So the weight of MST is 18.

Scanned by TapScanner

Analysis

Time Complexity:

(i) Assume the given graph has n vertices and m edges

ii) The prim's alg does insert, getMin and deleteMin about n times (while loop) and the decrease key operations for almost m times.

iii) Assume insert is less expensive than deleteMin

iv) Time Complexity of n vertices & m edges using prim's algorithm.

$$T(n,m) = O(nT(\text{getMin}) + nT(\text{deleteMin}) + mT(\text{decreasekey}))$$

⇒ The runtime of decreasekey is faster than $O(\log n)$ and in the while loop getMin and deleteMin are performed sequentially.

So, the above eqn is rewritten as,

$$T(n,m) = O(n(n)) + m$$

$$= O(n^2 + m)$$

$$T(n,m) = O(n^2)$$

Scanned by TapScanner

2) Explain Minimum Spanning Tree with Kruskal's algorithm. { (APR/MAY '15) (NOV/DEC '16) }

Kruskal's Algorithm

→ It uses greedies strategy.

Strategy:

1. To construct MST, edges of the graph are selected in non decreasing order of cost (or) distance.
   * Discards an edge if it forms a cycle.
2. Set of t edges so far selected for the spanning tree will form a forest, but not necessarily one tree.
3. Terminates when all edges have been processed.

ALGORITHM Kruskal(MST(G, n) //outline

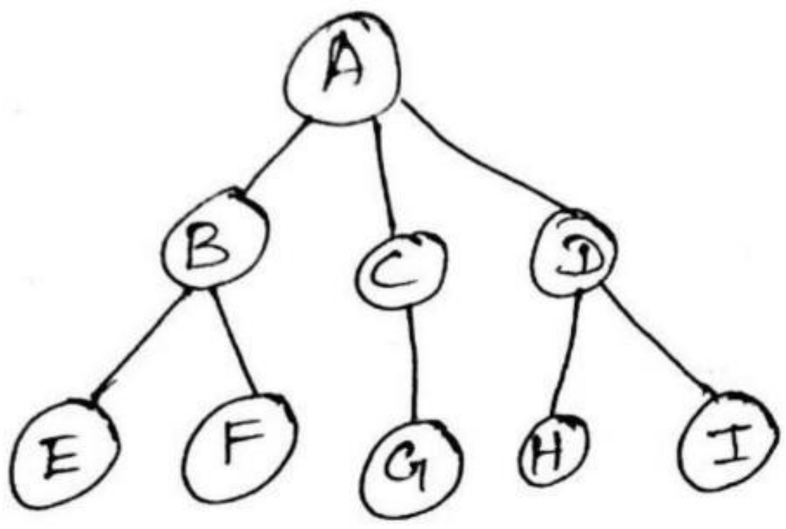
```
R = E; //R is remaining edges
F = φ; //F is forest edges.
while (R is not empty)
    Remove the lightest (shortest) edge vw, from R;
    if (vw does not make a cycle in f)
    Add vw to F;
return F;
```


Scanned by TapScanner

Forest:

→ A forest is a set of $n \geq 0$ disjoint trees. The notion of a forest is very close to that of a tree because if we remove the root of a tree, we get a forest.

⇒ For eg; if we remove A, we get a forest with 3 trees.



Tree representation

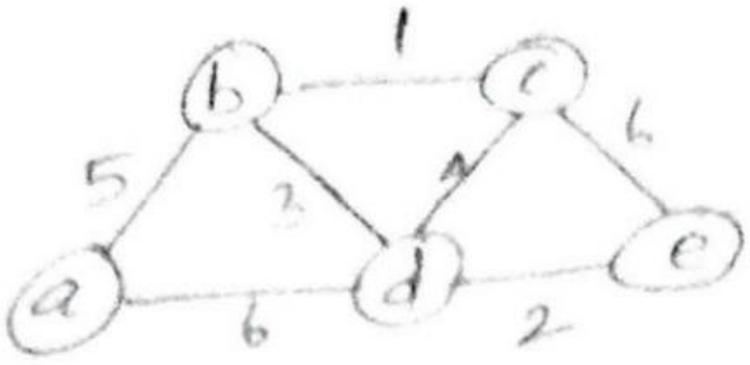
Tree representation after removing A - forest

Spanning Tree collection

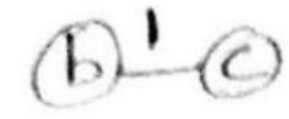
⇒ Let $G = (V, E, W)$ be a weighted undirected graph. A spanning tree collection for G is a set of trees, one for each connected component of G, such that each tree is a spanning tree for its connected component.

⇒ MST is collection of a spanning tree collection whose edges have minimum total weight, (ie) collection of MST.

Scanned by TapScanner

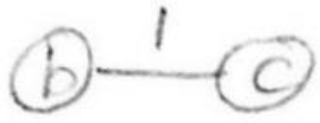


Step 1



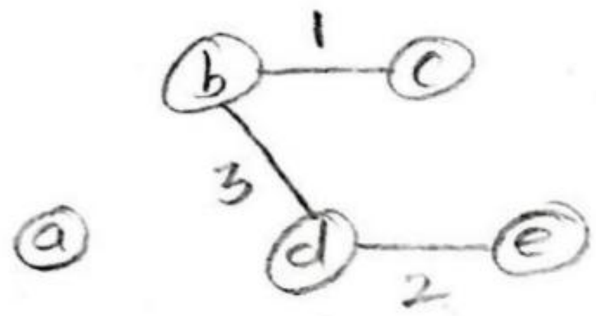
Edge with minimum weight bc = 1 is selected.

Step 2



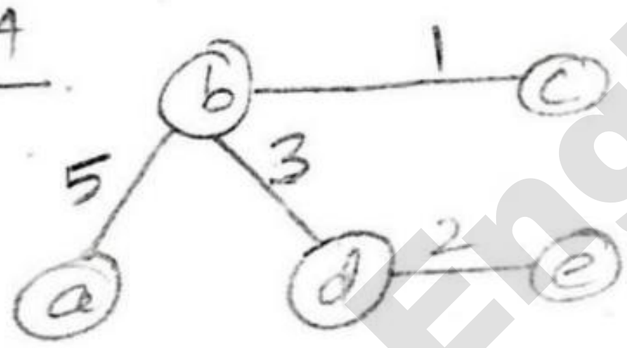
Second edge is added next minimum weightage edge (ie) de = 2

Step 3



Third edge is added (bd = 3)

Step 4



Fourth edge is added (ba = 5)

Analysis

(i) The priority queue of edges can be implemented as a min heap, since the decreasekey operation is not used in the alg. It can be built in time $O(m)$. Deleting all the edges require $O(m \log m)$ in the worst case.

(ii) The worst case running time of Kruskal's MST $= O(m \log m)$

Scanned by TapScanner

(3) Explain how greedy approach is used in Dijkstra's algorithm for finding the single-source shortest path. (NOV/DEC '17) (NOV/DEC '18)

⇒ This is a single source shortest path algorithm, used for two different types of applications.

(a) To find a minimum weight path between two specified vertices.

(b) To find minimum weight paths between s and every vertex reachable from s.

⇒ Dijkstra algorithm is used to find the minimum weight path from a specified source vertex to every other vertex in a weighted directed or undirected graph.

⇒ The weight (length or cost) of a path is the sum of the weights on the edges of in that path.

⇒ when weight is interpreted as distance, a min weight path is called as shortest path.

Principle (or) strategy

1). The Dijkstra's alg selects a starting vertex from the given graph and classifies the start vertex under "tree vertices".

Scanned by TapScanner

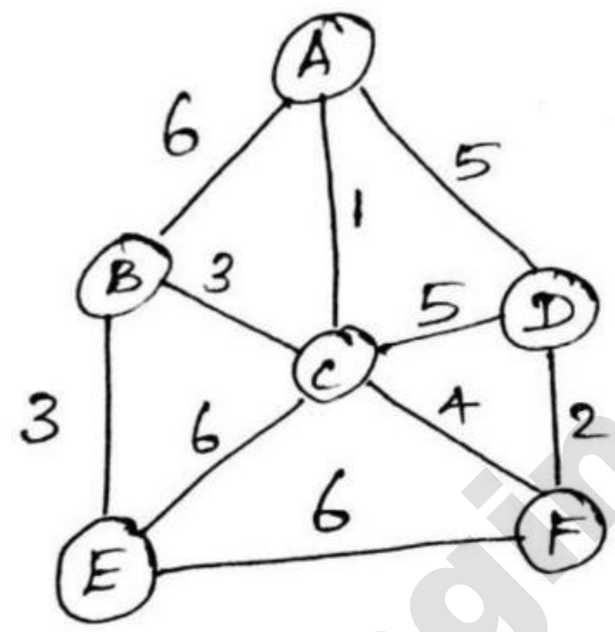
2) The nodes adjacent to tree vertices are identified and classified under "fringe vertices" and the remaining vertices are classified as "unseen vertices".

3) The key step of the algorithm is selection of new vertex and an incident edge from the "fringe" which is now added to "tree vertices".

* After this new inclusion again the "fringe vertices" and "unseen vertices" are reclassified.

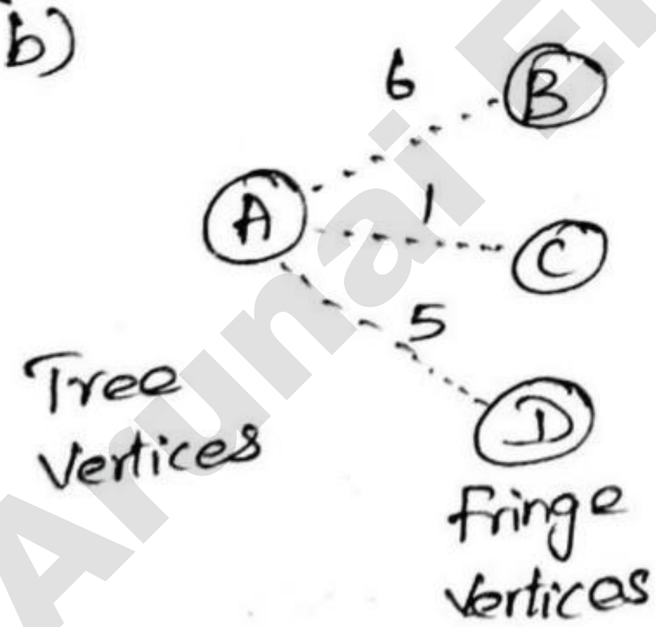
Example:

(a)



* A weighted graph
$G = (V, E, W)$

(b)



⇒ The tree and fringe after the starting vertex is selected.

$d(A, A) + W(AB) = 6$

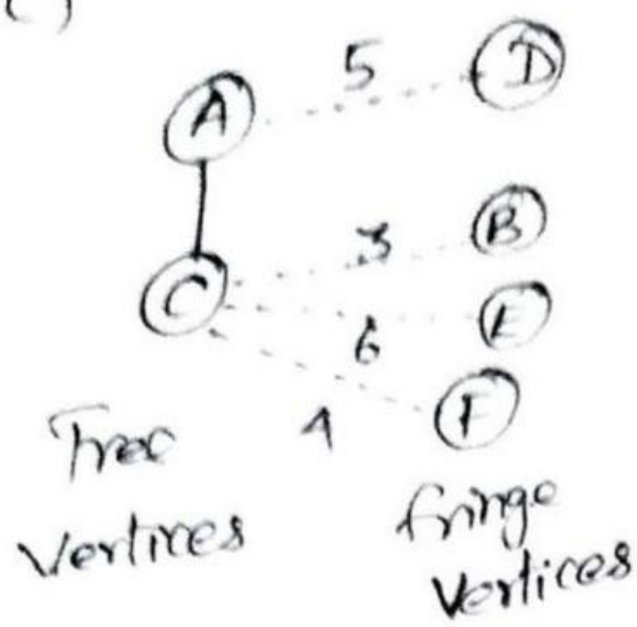
$d(A, A) + W(AC) = 1$

$d(A, A) + W(AD) = 5$

select AC next.

Scanned by TapScanner

(c)

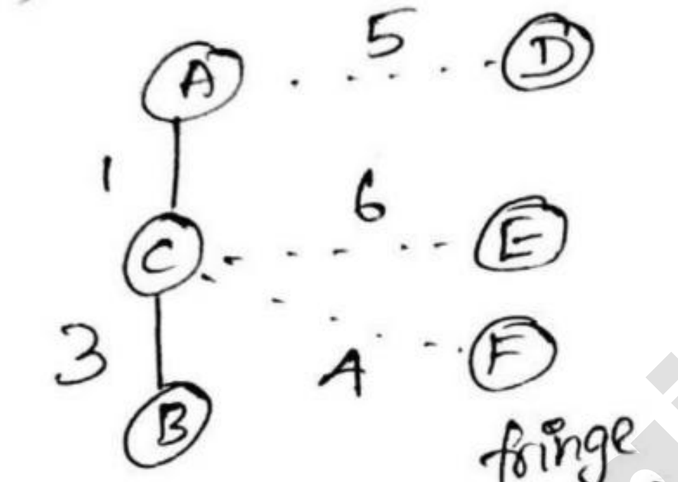


⇒ AB was replaced by CB as a candidate edge.

⇒ CD was considered, but not chosen, because AD is exist as a candidate edge.

⇒ d(A,A) + W(AD) = 5
d(A,C) + W(CB) = 4
d(A,C) + W(CE) = 7
d(A,C) + W(CF) = 5
Select CB next.

(d)

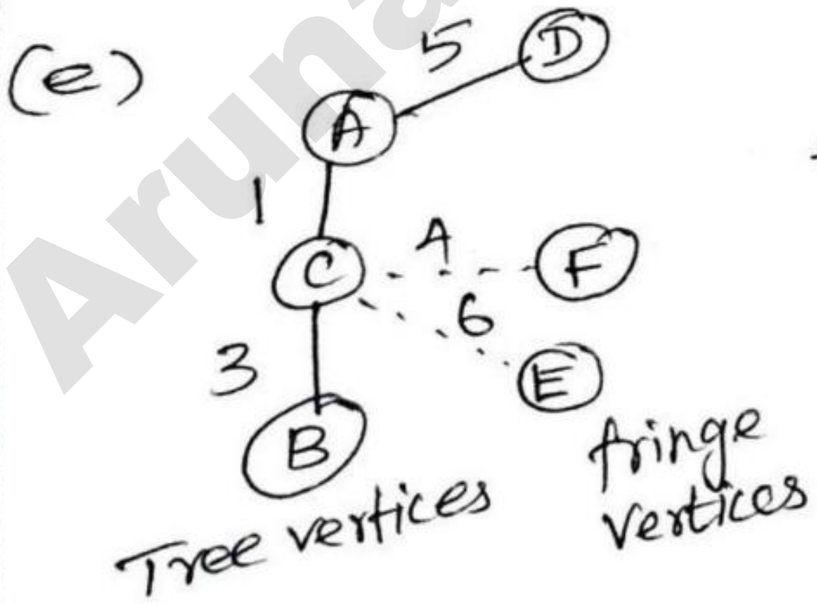


⇒ Either BE (or) CE may be to Selected, because the path value from A-E is same.

(i) A-C-B-E = 7
A-C-E = 7

* d(A,A) + W(AD) = 5
d(A,C) + W(CF) = 5
d(A,C) + W(CE) = 7
Select AD next.

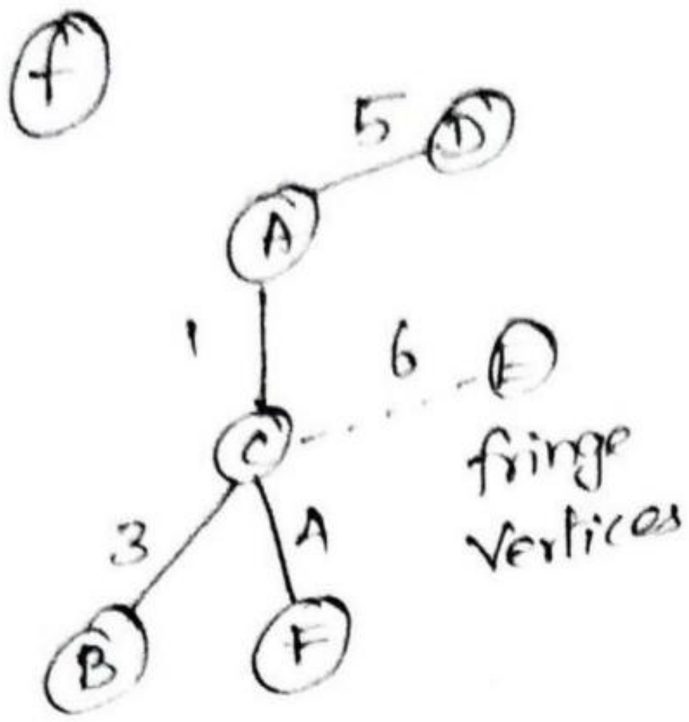
(e)



⇒ *DF was considered, but not choosen, bcos CF is exist as a candidate edge.

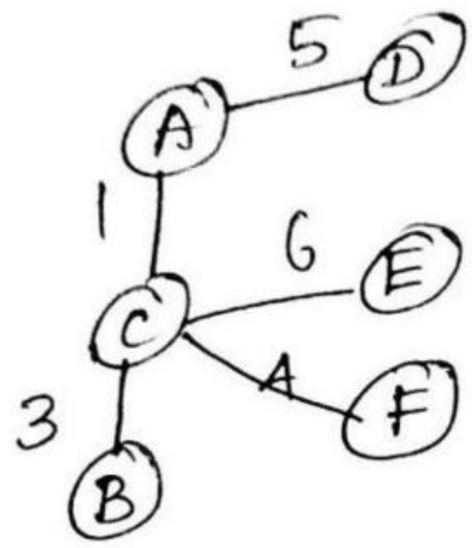
⇒ d(A,C) + W(CF) = 5
d(A,B) + W(CE) = 7
Select CF next.

Scanned by TapScanner



* The only one vertex remains is Vertex E.
* Select CE next.

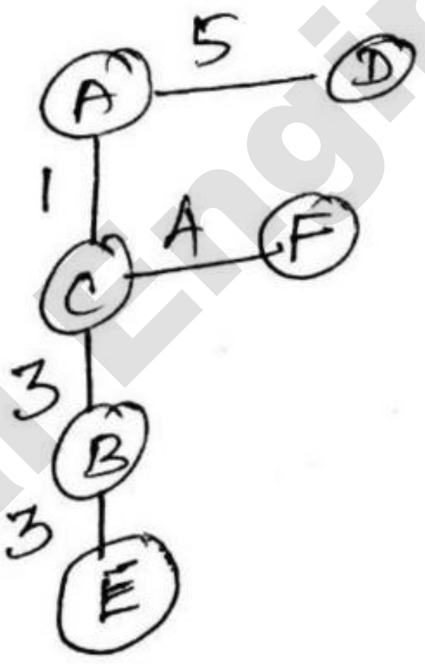
(g)



⇒ The Vertex E can be selected from B (or) C, bcos the path length is same.

⇒ Shortest distance from Vertex A to any other vertices. (B, C, D, E, F).

(OR)



ALGORITHM

dijkstraSP(G, n) //outline
Initialize all vertices as unseen
Start the tree with the specified source vertex
S; reclassify it as tree.

Scanned by TapScanner

define $d(s,s)=0$

Reclassify all vertices adjacent to s as fringe.

while there are fringe vertices.

Select an edge between a tree vertex t and a fringe vertex v such that $(d(s,t)+W(tv))$ is minimum;

Reclassify v as tree; add edge tv to the tree;

define $d(s,v) = (d(s,t)+W(tv))$

Reclassify all unseen vertices adjacent to v as fringe.

Analysis

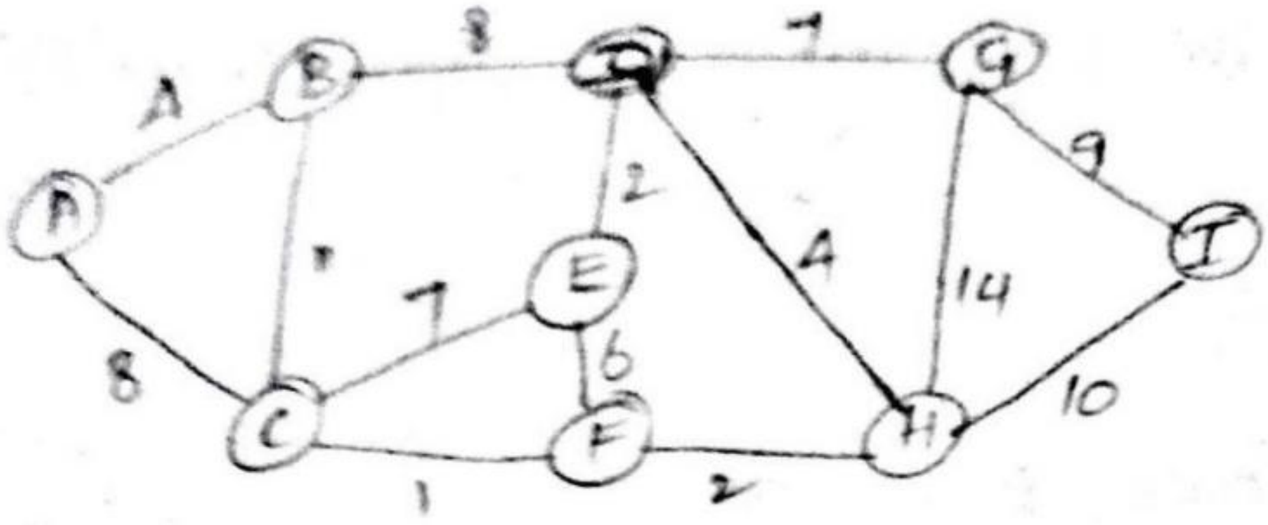
⇒ The worstcase complexity of Dijkstra's alg is $O(n)^2$.

⇒ If a significant no. of vertices are expected to be unreachable, it might be more efficient to test for reachability as a preprocessing step, eliminate unreachable vertices and renumber the remaining vertices as $1 \ldots n_r$.

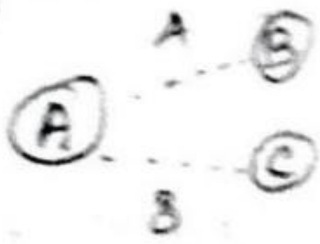
⇒ The total cost would be in $O(m+n_r^2)$, rather than $O(n^2)$.

Scanned by TapScanner

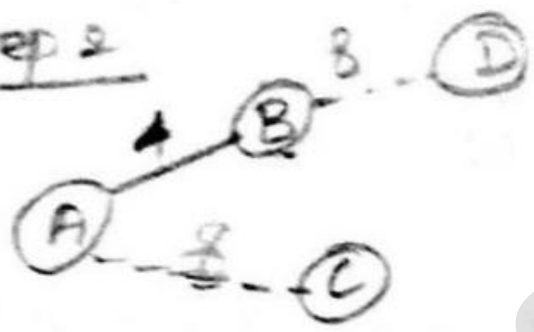
9 problem (University pbm) [Nov/Dec '18]



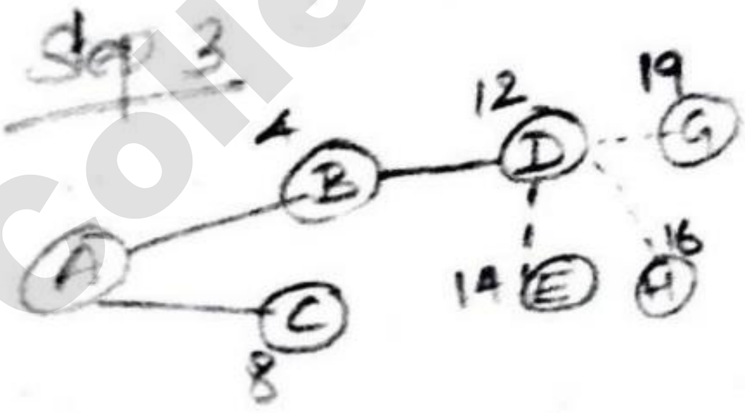
Step : 1



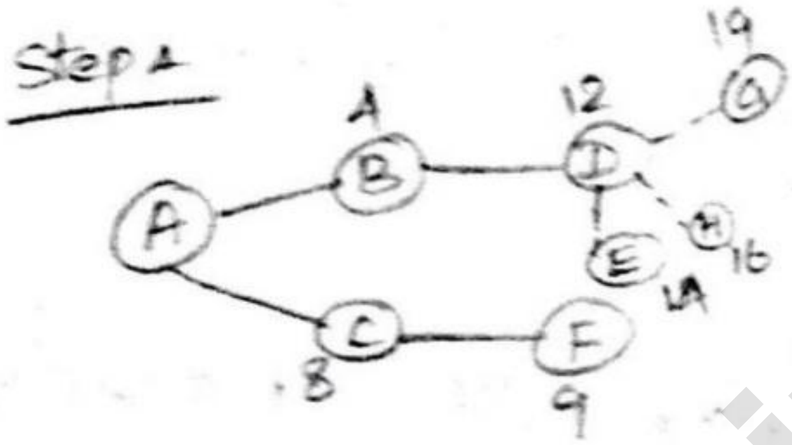
Step 2



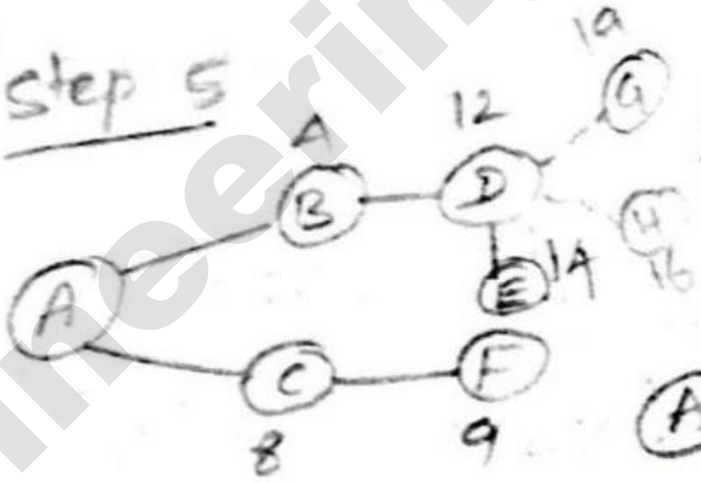
Step 3



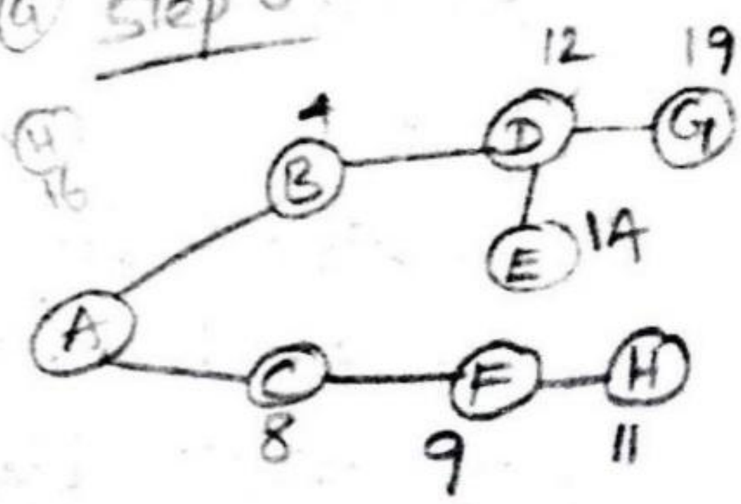
Step 4



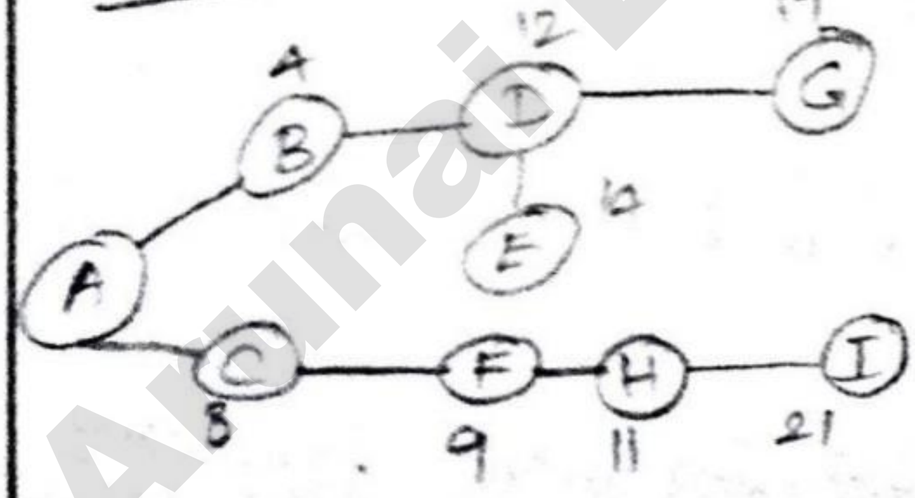
Step 5



Step 6



Step 7



Scanned by TapScanner

(4) Explain the steps in Building Huffman Tree. find the codes for the alphabets given as example. according to frequency. {(NOV/DEC'17) (APR/MAY'17) (NOV/DEC'15) (APR/MAY'15) (APR/MAY'19)}

Huffman Trees:

⇒ The huffman trees are constructed for encoding a given text of characters, each character is associated with some bit sequence. This bit sequence is named as code word.

⇒ The encoding is classified into two types, depends on the number of bits used for each character in the text.

(a) Fixed length encoding

⇒ Each character is associated with a bit string of some fixed length.

Example : ASCII – 7 bits for a character

(b) Variable length encoding

⇒ Each character is associated with a bit string of different length.

(i) Shorter length codeword for more frequent characters and longer length codeword for less frequent characters. (eg) Telegraph code.

Scanned by TapScanner

ii) Using the property of prefix code - no codeword is a prefix of another character's code word. This property is used to identify the no. of bits required to encode the i^{th} character of a text.

Huffman's Algorithm:

Step 1: Initialize n one-node trees and label them with the characters of the alphabet. Record the freq of each character in its tree's root to indicate the tree's weight.

⇒ The weight of a tree will be equal to the sum of the frequencies in the tree's leaves.

Step 2: Repeat the following operation until a single tree is obtained. Find two trees with the smallest weight. Make them the left and right subtree of a new tree and record the sum of their weights in the root of the new tree as its weight.

⇒ A tree constructed by this alg is called a Huffman tree - It defines the character in a form of strings (code), based on their frequencies in a given text is known as Huffman code.

Scanned by TapScanner

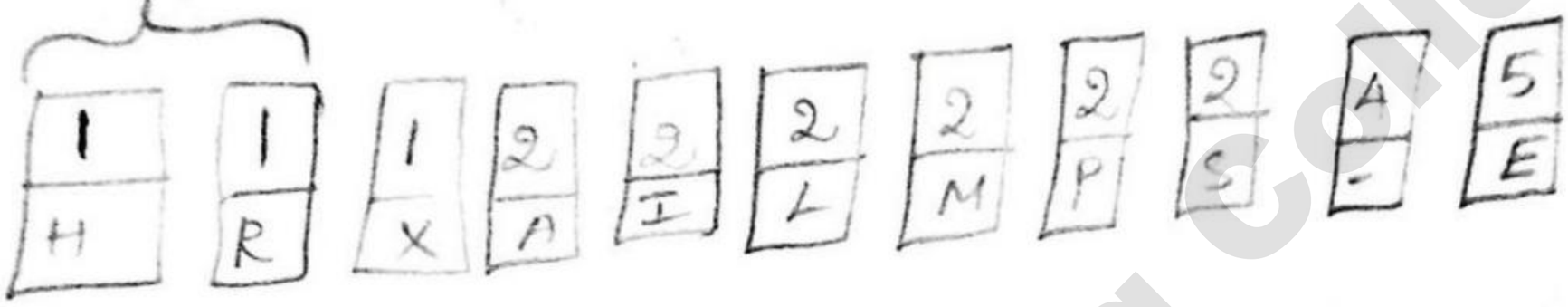
eg:

	A	R	5	I	2	2	2	2	I	2	I
	-	A	E	H	I	L	M	P	R	S	X

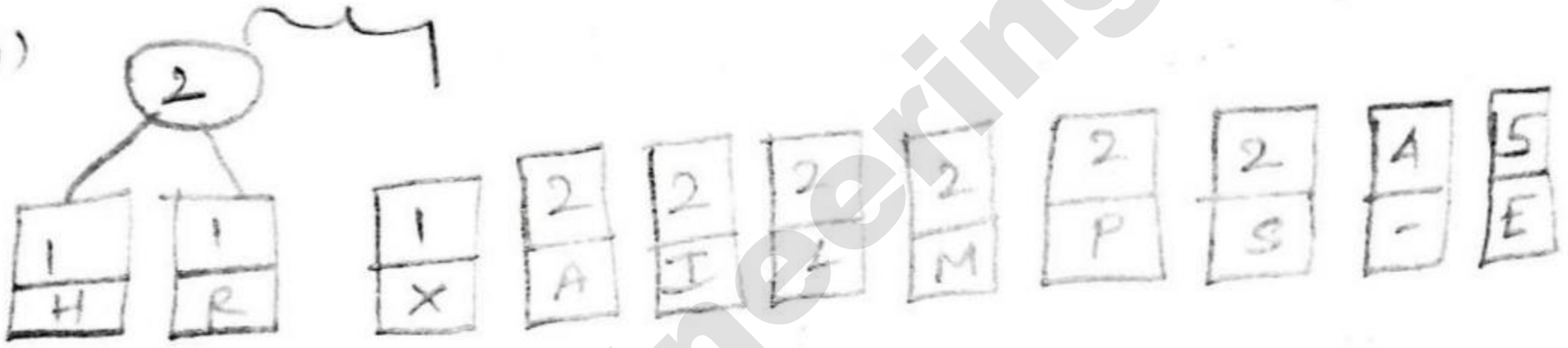
⇒ Arrang in increasing order

Frequency	1	1	1	2	2	2	2	2	2	4	5
Alphabets	H	R	X	A	I	L	M	P	S	-	E

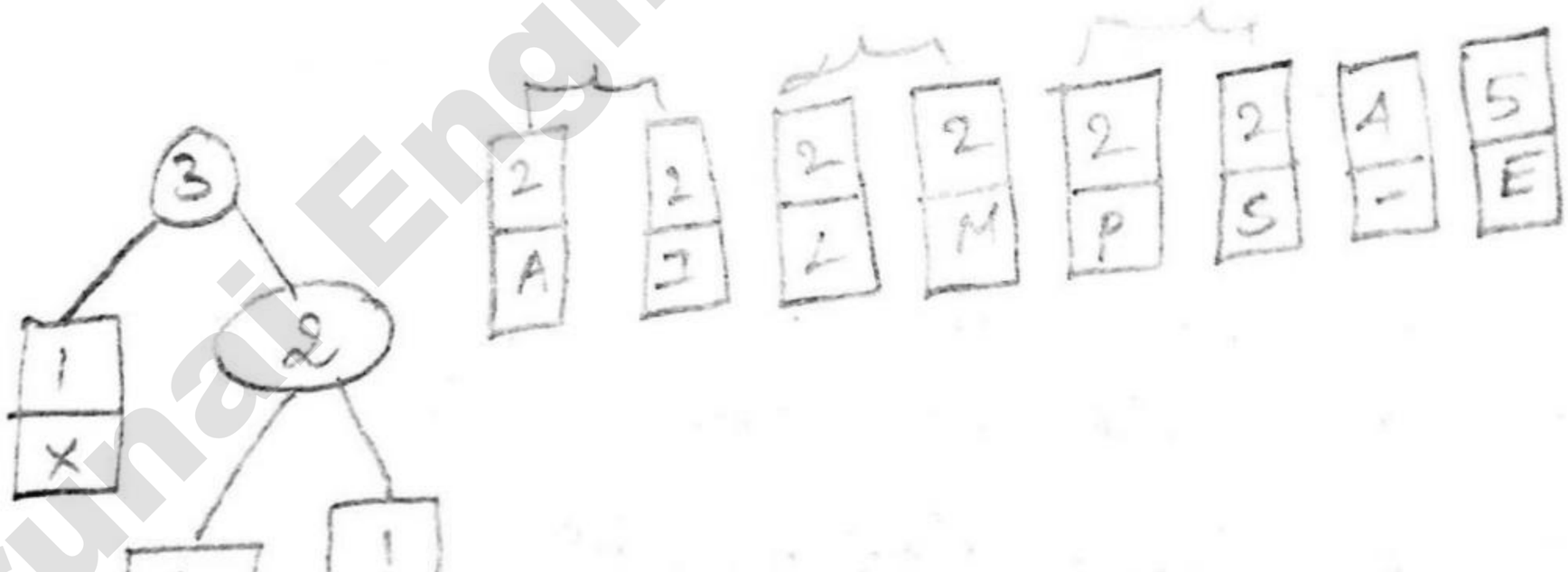
step i)



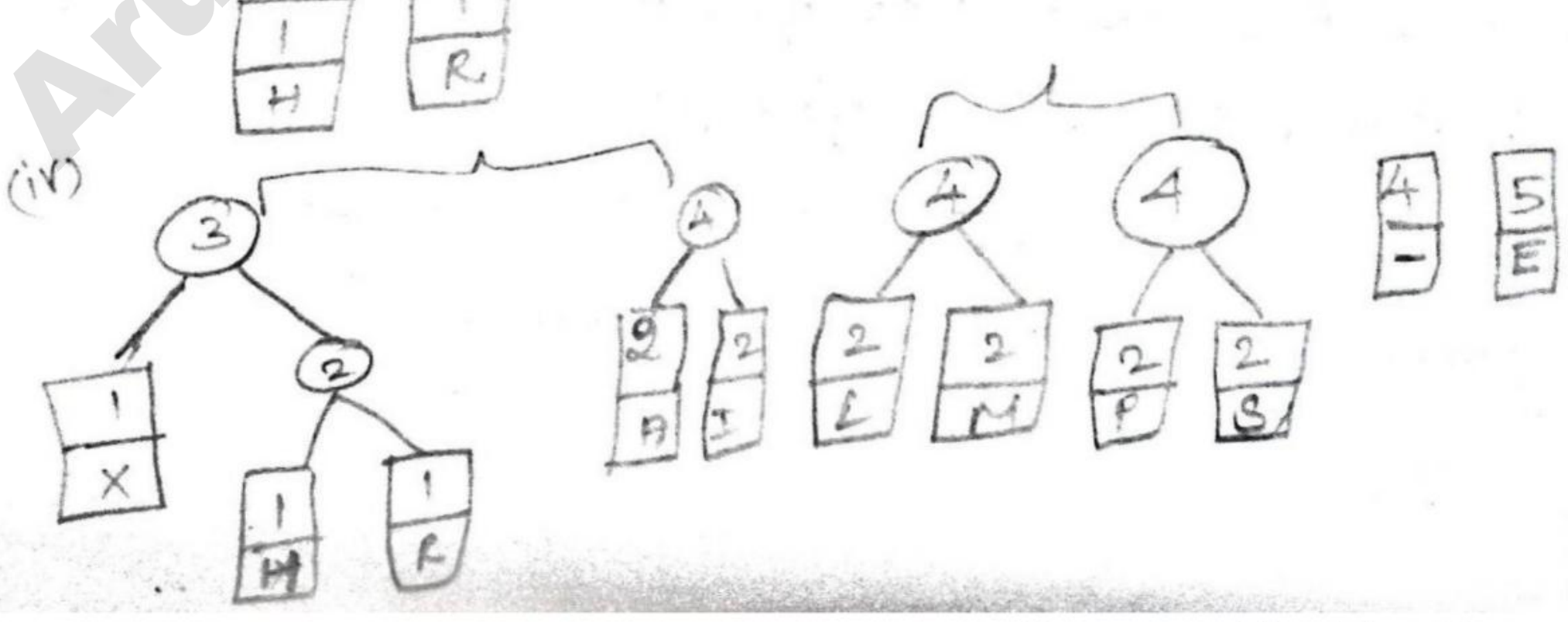
(ii)



iii)

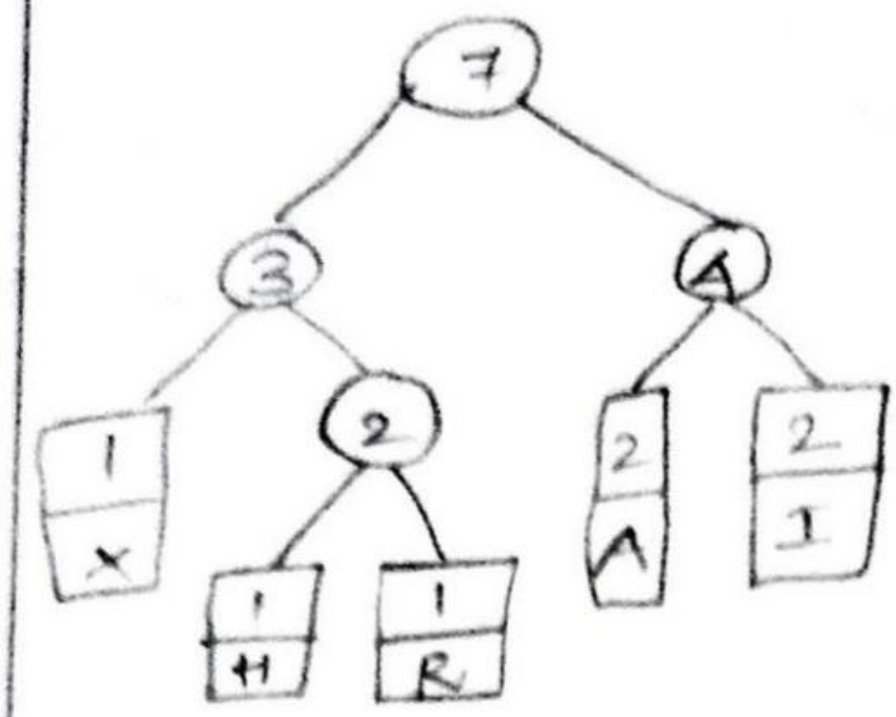


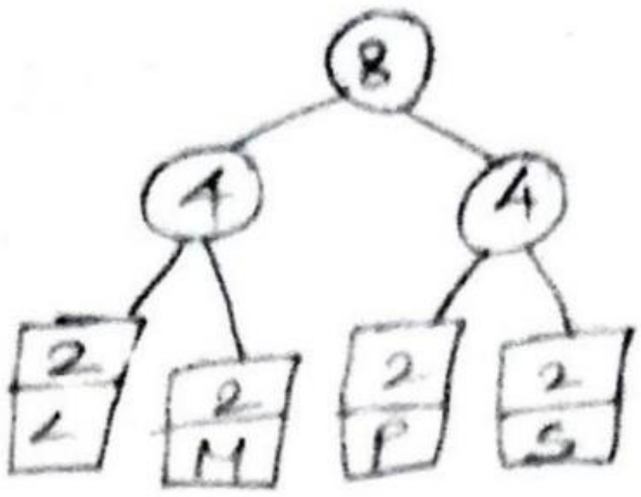
(iv)

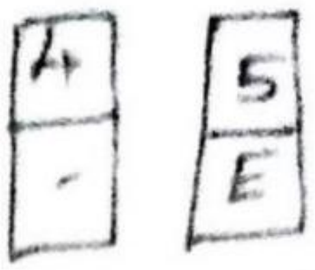


Scanned by TapScanner

V

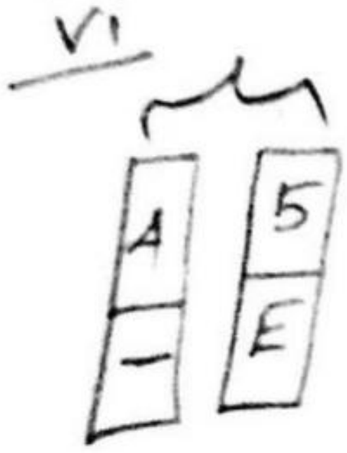


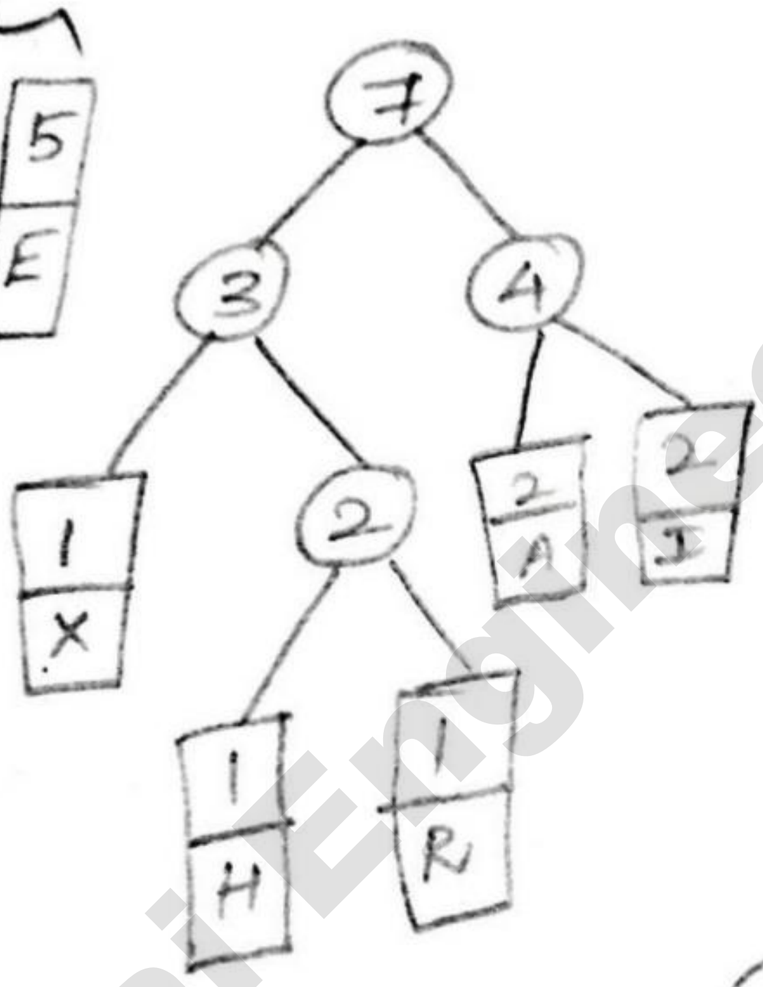


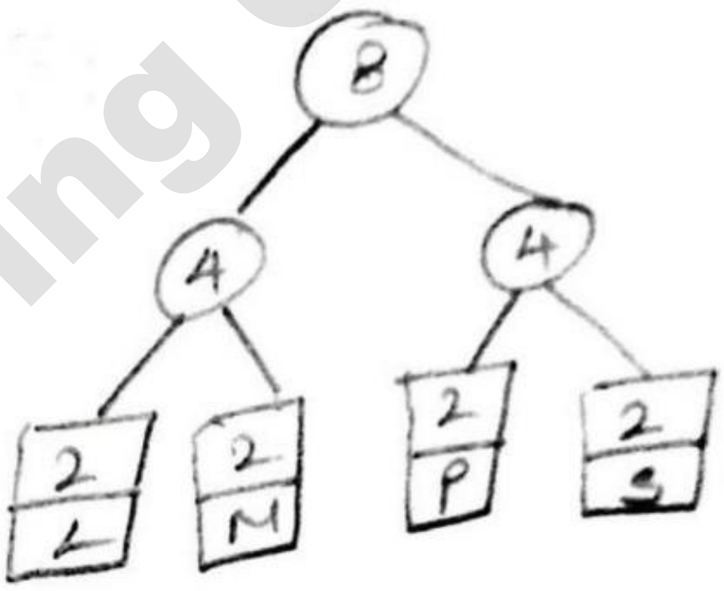


⇒ Arrange in order (increasing)

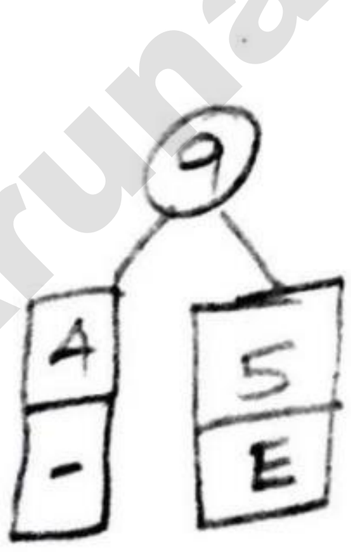
VI

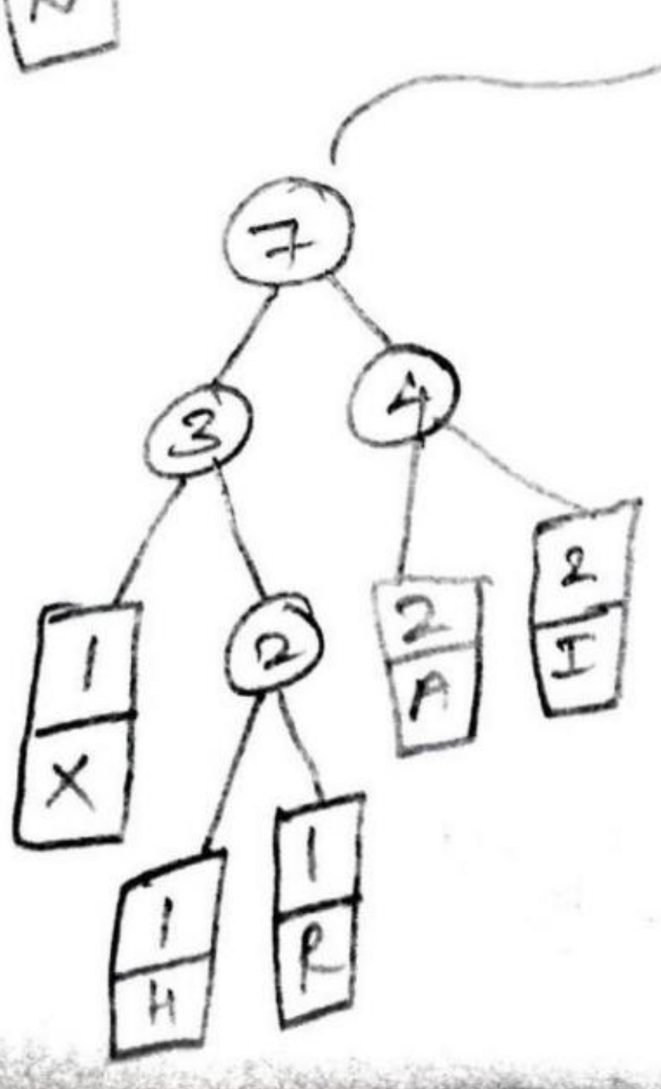


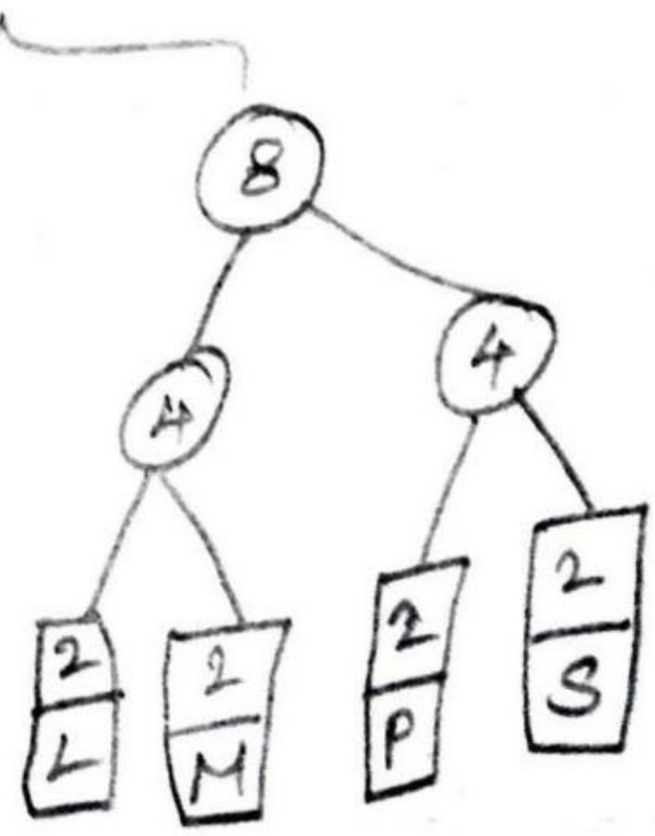




VII

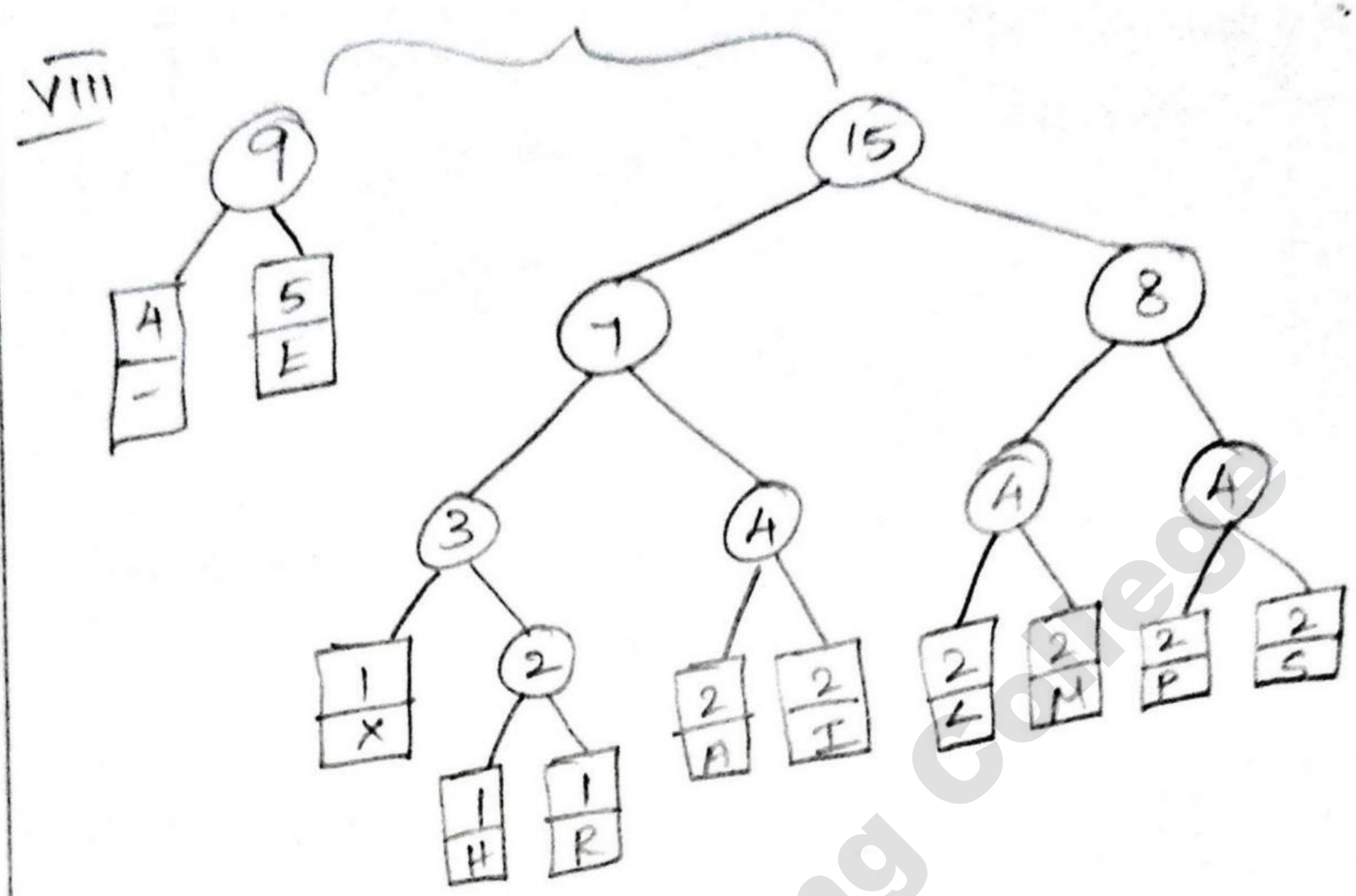




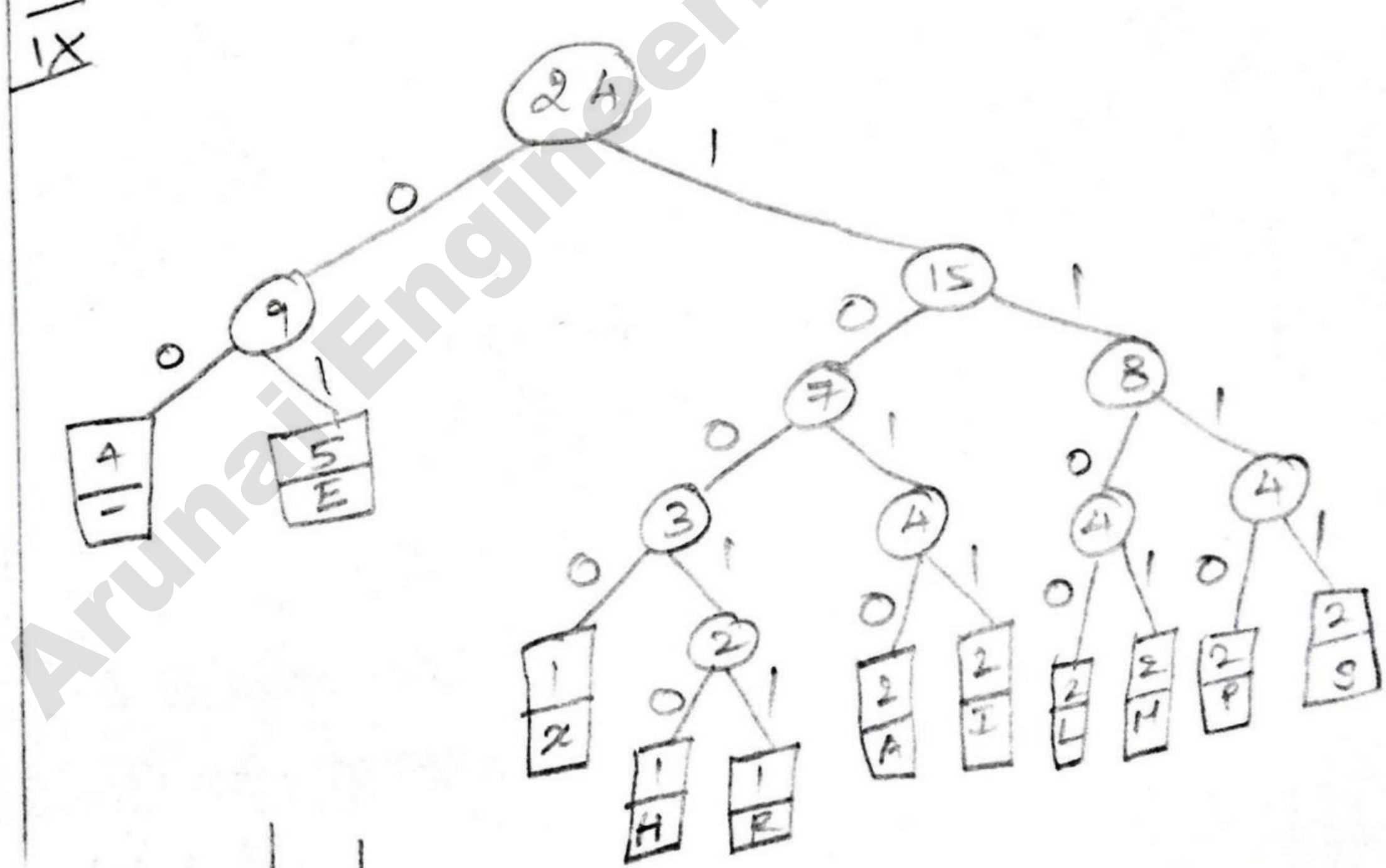


Scanned by TapScanner

VIII



IX



Alphabet	—	A	E	H	I	L	M	P	R	S	X
Frequency.	4	2	5	1	2	2	2	2	1	2	1
Codeword	00	1010	01	10010	1011	1100	1101	1110	10011	1111	1000

Scanned by TapScanner

(5) Write a Greedy algorithm to solve the 0/1 knapsack problem. Analyse its time complexity. Show that this algorithm is not optimal with an example.

{(Nov/Dec'19)

1. N objects is the range of (1...n), of weights $(w_1, w_2, \ldots w_n)$ and profits $(P_1 \ldots P_n)$ resp.

2. A knapsack of capacity m (ie) the total weights placed in the bag should not exceed $m (\leq m)$

Selection criteria:

⇒ If a fraction x_i of an objects i is placed in the bag then a profit of $P_i x_i$ is earned, and the weight of the bag contents is $\leq m$.

⇒ x_i is in range of 0 to 1 (ie) $0 \leq x_i \leq 1$.

⇒ i is in range of 1 to n (ie) $1 \leq i \leq n$.

objective function:

⇒ To maximise the total profit earned by filling the bag appropriately.

$$\text{maximise} \sum_{1 \leq i \leq n} P_i x_i \qquad 1 \leq i \leq n$$

$$\text{subject to} \sum_{1 \leq i \leq n} w_i x_i \leq m. \qquad 0 \leq x_i \leq 1$$

Scanned by TapScanner

Feasible Solution:

⇒ Any set $(x_1 \ldots x_n)$ satisfying the weight limit condition.

Optimal solution:

⇒ A feasible solution which gives maximum profit.

Problem Instance:

(eg) $N=3$, $M=20$, $(P_1, P_2, P_3) = (25, 24, 15)$

$(w_1, w_2, w_3) = (18, 15, 10)$

Solution:

⇒ In case the sum of weights of all the objects is less than M.

$\sum_{1 \le i \le n} w_i \le m$ then all $x_i = 1$ and that is an optimal sol.

⇒ But if the sum of all weights is greater than m, then all x_i cannot have the value 1 and x_i is a fraction (ie) $0 \le i \le 1$.

Scanned by TapScanner

ALGORITHM:

```
Void Greedyknapsack (float m, int n)
// P[1...n] and W[1...n] contain the profits and weights
   resp.
// m is the knapsack size and x[1...n] is the sol vector.
{
   for (int i=1; i<=n; i++) x[i]=0.0;
   float u=m;
   for (i=1; i<=n; i++)
   {
      if (w[i]>U) break;
      x[i] =1.0; U=U-w[i];
   }
   if (i<=n) x[i]=U/w[i];
}
```

Analysis

⇒ The time complexity of all three strategies is O(n) and the time required to sort the objects in the required order initially is not considered.

Scanned by TapScanner

Example.

Let us consider that the capacity of the knapsack N=60 and the list of provided items are shown in following table.

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio $(\frac{Pi}{wi})$	7	10	6	5

⇒ As the provided items are not sorted based on $\frac{Pi}{wi}$. After sorting, the items are as shown below.

Item	A	B	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio $(\frac{Pi}{wi})$	10	7	6	5

Sol 1st all of B is chosen as weight of B is less than the capacity of the knapsack. Next, item A is chosen, as the available capacity of the knapsack is greater than the weight of A. Now, C is chosen as the next item.

⇒ fraction of C (ie (60-50)/20) is chosen.

⇒ Now, the capacity of the knapsack is equal to the selected items. Hence no more items selected.

The total weight of the selected items is $10+40+20*(\frac{10}{20})=$

The total profit is $100+280+120*(\frac{10}{20})=380+60=440$

Scanned by TapScanner

(6) Write the floyd algorithm to find all pairs shortest path and derive its time complexity. (Nov/Dec '19)

⇒ This algorithm finds the shortest path from each vertex to all other vertices of a given weighted connected graph, named as All-pairs shortest path problem.

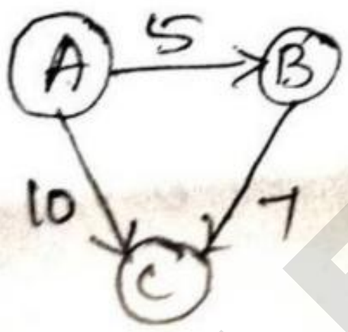
Weighted Graph.

⇒ A weighted graph is a triple (V, E, W), where (V, E) is a graph and W is a weight from vertex i to j.

W[i][j] = 0 if i = j

W[i][j] = 0 if there is no edge between i and j

W[i][j] = weight of edge.



⇒

	A	B	C
A	0	5	10
B	∞	0	7
C	∞	∞	0

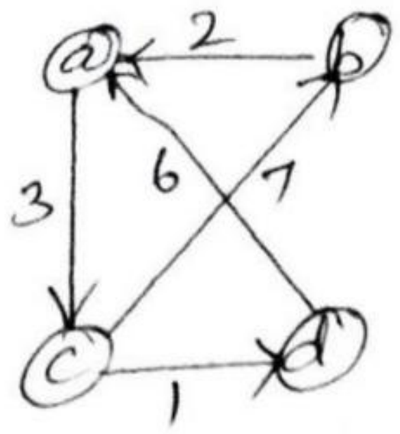
Distance Matrix

⇒ A n by n matrix D to record the lengths of shortest path is called distance Matrix

⇒ The element d_{ij} in the i^{th} row and j^{th} column of the matrix indicates the length of the shortest path from i^{th} vertex to the j^{th} vertex ($1 \le i, j \le n$).

Scanned by TapScanner

Example.



$$W = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array} \qquad D = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

Problem Definition:

Let $G = (V, E)$ be a directed graph with n vertices. Let cost be the cost adjancy matrix for G such that cost $(i, i) = 0$ for $1 \le i \le n$. The cost (i, j) is the length or cost of edge $\langle i, j \rangle$, if $\langle i, j \rangle \in E(G)$ and cost $(i, j) = \infty$ if $i \neq j$ and $\langle i, j \rangle \notin E(G)$.

Procedure to be followed

1) Computes the distance matrix of a weighted graph with n vertices through a series of n by n matrices.

$D^{(0)}, D^{(1)} \ldots D^{(k-1)}, D^{(k)}, \ldots D^{(n)}$.

2) $D^{(0)}$ is the given weighted matrix.

3) $D^{(1)}$ is computed from $D^{(0)}$ by including the intermediate vertex in the path from V_i to V_j as per the rules given below.

A) $D^{(k)}$ matrix is the shortest distance through k-1 intermediate vertices, with path length of k.

Scanned by TapScanner

57 Construct until $D^{(n)}$, (ie) the required distance matrix of shortest path blw each pair of vertices.

Formula:

To find the shortest path between any two vertices during the computation of $D^{(0)} \ldots D^{(n)}$.

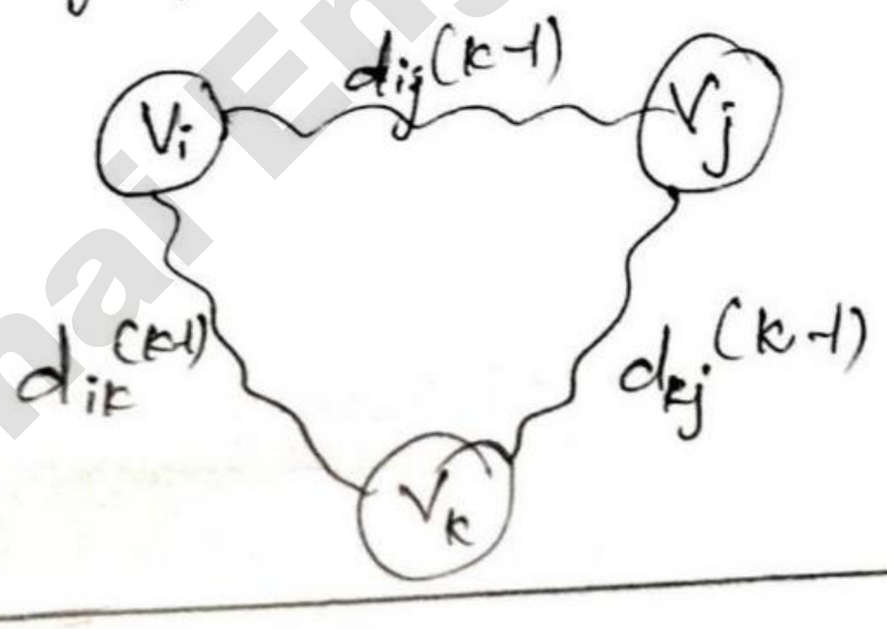
i) A shortest path from V_i to V_j with intermediate vertices $(V_1, V_2 \ldots V_{k-1})$ is,

$$D^{(k)} = d_{ij}(k) = d_{ij}^{(k-1)}$$

ii) A shortest path from V_i to V_j with intermediate vertices $(V_1, V_2, \ldots V_{k-1}, V_k)$ is,

$$D^{(k)} = d_{ij}(k) = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$$

⇒ The graphical representation of these 2 cases,



$$\therefore D^{(k)} = d_{ij}(k) = \min \left\{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right\}$$

for $k \geq 1$, $d_{ij}(0) = w_{ij}$.

Scanned by TapScanner

ALGORITHM Floyd (W[1...n, 1...n])

//Implements Floyd's alg for the all pairs shortest path's pbm

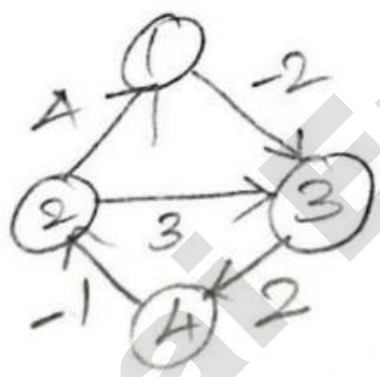
//I/P : The weight matrix W of a graph

//O/P : The distance matrix of the shortest path's lengths.

D ← W //is not necessary if W can be overwritten

```
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            D[i,j] ← min{D[i,j], D[i,k] + D[k,j]}
return D.
```

Example



$$D^{(0)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & -2 & \infty \\ 4 & 0 & 3 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & -1 & \infty & 0 \end{bmatrix} \end{array}$$

Step:1 $D^{(k)} = d_{ij}(k) = \min\{d_{ij}(k-1), d_{ik}(k-1) + d_{kj}^{(k-1)}\}$

$$D^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & -2 & \infty \\ 4 & 0 & 3 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & -1 & \infty & 0 \end{bmatrix} \end{array} \quad i=2, j=4$$

$i = 2, j = 3, k = 1$

$D^{(1)} = d_{23}^{(1)} = \min\{d_{23}^{(0)}, d_{31}^{(0)} + d_{13}^{(0)}\}$

$= \min\{3, \infty + (-2)\}$

$= 3$

$D(1) = d_{2A}(1) = \min\{d_{24}(0), d_{41}(0) + d_{1A}^{(0)}\}$

$= \min\{\infty, \infty + \infty\} = \infty$

Scanned by TapScanner

Step 2

	1	2	3	4
1	0	∞	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	∞	-1	∞	0

$\Rightarrow$

	1	2	3	4
1	0	∞	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	3	-1	-2	0

execution $i=3, j=1, k=2$

$$d_{31}(2) = \min\{d_{31}(1), d_{32}(1) + d_{21}(1)\}$$
$$= \min\{\infty, \infty + 4\} = \infty$$

$i=4, j=1, k=2$

$$d_{41}(2) = \min\{d_{41}(1), d_{42}(1) + d_{21}(1)\}$$
$$= \min\{\infty, -1+4\}$$
$$= 3$$

Step 3

	1	2	3	4
1	0	∞	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	3	-1	-2	0

$\Rightarrow$

	1	2	3	4
1	0	∞	-2	0
2	4	0	3	5
3	∞	∞	0	2
4	3	-1	-2	0

Execution

$i=1, j=4, k=3$

$$= \min\{d_{14}^{(2)}, d_{13}^{(2)} + d_{34}^{(2)}\}$$
$$= \min\{\infty, -2+2\}$$
$$= 0$$

$i=2, j=4, k=3$

$$= \min\{d_{24}^{(2)}, d_{23}^{(2)} + d_{34}^{(2)}\}$$
$$= \min\{\infty, 3+2\}$$
$$= 5$$

Step 4

	1	2	3	4
1	0	∞	-2	0
2	4	0	3	5
3	∞	∞	0	2
4	3	-1	-2	0

$\Rightarrow$

	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	-2	0

Find out

(1,1) (2,1) (3,1)
(1,2) (2,2) (3,2)
(1,3) (2,3) (3,3)

Scanned by TapScanner

Execution for Step 4

(i) $i=1, j=2, k=4$
$min = -1$

(ii) $i=1, j=3, k=4$
$min = -2$

iii) $i=2, j=1, k=4$
$min = A$

iv) $i=2, j=3, k=4$
$m = 3$

iv) $i=3, j=1, k=4$
$\Rightarrow min = 5$

(v) $i=3, j=2, k=4$
$min = 1$

UNIT

IV

Scanned by TapScanner

UNIT - IV

1) when is a residual network in the context of flow networks.

⇒ A residual network graph indicates how much more flow is allowed in each edge in the network graph.

⇒ If there are no augmenting paths possible from to, the the flow is maximum.

⇒ The result (ie) the max flow will be the total flow out of source node which is also equal to total flow in to the sink node.

2) when a linear program is said to be unbounded?

* An unbounded solution of a linear programming problem is a situation where objective function is infinite.

* A linear programming problem is said to have unbounded solution if its solution can be made infinitely large without violating any of its constraints in the problem.

3) State the principle of duality?

* The principle of duality in Boolean algebra states that if you have a true Boolean statement (equation) then the dual of the Boolean statement (equation) is true.

Scanned by TapScanner

* The dual of a boolean statement is found by replacing the statement's symbols with their counterparts.

4) Define the capacity constraint in the context of maximum flow problem.

Capacity Constraints

$0 \leq x_{ij} \leq u_{ij}$ for every edge $(i,j) \in E$

⇒ Since no material can be lost or added to by going through intermediate vertices of the network, the total amount of the material leaving the source must end up at the sink.

$$\sum x_{1j} = \sum x_{jn}$$

$j:(1,j) \in E \quad j:(j,n) \in E$

⇒ The value of the flow is defined as the total outflow from the source.

5) Describe iterative improvement technique.

* This is a computational technique in which with the help of initial feasible solution the optimal solution is obtained iteratively until no improvement is found.

Scanned by TapScanner

(6) How Iterative improvement solves problems.

* The problem is an optimization problem, to find the solution that minimizes or maximizes some value.
* An initial solution can be easily found.
* It can be improved by a sequence of small changes.
* It is returned when no more improvements can be made.

(7) What is meant by Bipartite Graph?

* A ~~Bigraf~~ Bipartite Graph $G_1 = (V, E)$ is a graph in which the vertex set V can be divided into two disjoint subsets x and y such that every edge $e \in E$ has one end point in X and the other end point in Y.
* A matching M is a subset of edges such that each node in V appears in at most one edge in M.

(8) What is maximum cardinality matching?

* A maximum matching (also known as maximum cardinality matching) is a matching that contains the largest possible number of edges.
* There may be many maximum matchings. The matching number of a graph is the size of a max matching.

Scanned by TapScanner

9) Illustrate the stable Marriage problem.

* The stable marriage problem (SMP) is the problem of finding a stable matching between two sets of elements given a set of preferences for each element.

* A matching is a mapping from the elements of one set to the elements of the other set.

10) Show the requirements of the standard form in simplex method.

* It must be a maximization problem.

* All the constraints must be in the form of linear equations with nonnegative right-hand sides.

11) what is a cuts in flow networks.

* Cut is a collection of arcs such that if they are removed there is no path from source to sink.

12) what is a state space graph?

* Graph organization of the solution space is state space tree.

13) Define extreme point theorem.

⇒ Any LP problem with a non empty bounded feasible region has an optimal solution.

⇒ An optimal solution can always be found at an

Scanned by TapScanner

points of the problems feasible region.

14) what do you mean by perfect Matching in Bipartite graph?

⇒* A perfect matching of a graph is a matching in which every vertex of the graph is incident to exactly one edge of the matching.

⇒ A perfect matching is a matching containing n/2 edges, meaning perfect matching are only possible on graphs with an even number of vertices.

15) what is an articulation point in a graph?

⇒ A vertex in an undireded connected graph is an articulation point (or cut vertex) iff removing it disconnects the graph. It can be through of as a single point of failure.

16) How is a transportation network represented?

⇒ Transportation networks generally refer to a set of links, nodes and lines that represent the infrastructure or supply side of the transportation.

⇒ The links have characteristics such as speed and capacity for roadways.

Scanned by TapScanner

frequency and travel time data are defined on transit links or lines for the transit system.

17) Define slack variable

→ Variables transforming inequality constraints into equality constraints are called slack variable.

18) List various applications of iterative improvement method

1. Simplex Method
2. Matching graph vertices
3. Stable marriage problem
4. finding maximum network flow.

19) what is two colorable graph?

* It is a graph that can be colored with only two colors in such a way that no edge connects the same color. The bipartite graph is two colorable graph.

20) Define Max-flow Min cut Theorem.

* The value of a maximum flow in a network is equal to the capacity of its minimum cuts -

AEC

1 Summarize the steps of the Simplex Method.

{(May/June '16) (Nov/Dec'16) (AR/May '17) (Nov/Dec '17) (AR/May '18) (Nov/Dec '18) (Nov/Dec'19)}

(or)

List the steps in simplex method & give the efficiency of the same.

⇒ Linear programming (LP) problem is to optimize a linear function of several variables subject to linear constraints.

⇒ Algorithm for solving the linear programming pbm is called Simplex method.

⇒ Linear programming problem is of the general form

maximize (or minimize) $c_1x_1 + \dots + c_nx_n$

subject to $a_{i1}x_1 + \dots + a_{in}x_n \leq (\text{or} \geq \text{or} =) b_i$,

$i = 1 \dots m$

$x_1 \geq 0 \dots x_n \geq 0$

⇒ The function $z = c_1x_1 + \dots + c_nx_n$ is called the object function;

⇒ Constraints $x_1 \geq 0 \dots, x_n \geq 0$ are called nonnegativity contraints.

Scanned by TapScanner

Feasible solution

⇒ Any point (x, y) that satisfies all the constraints of the problem is called feasible solution.

Slack variables

⇒ Variables u and v transforming inequality constraints into equality constraints are called slack variables.

Basic feasible solution

⇒ A basic solution for which all variables are non negative is called basic feasible solution.

Applications of Linear programming

* Airline Crew scheduling
* Transportation & communication network planning.
* Oil exploration & refining
* Industrial production optimization.

⇒ The function $z = c_1x_1 + \dots c_nx_n$ is called the objective function.

Scanned by TapScanner

Step 1: optimality test

⇒ If all the entries in the objective row are nonnegative - stop : The tableau represents an optimal solution whose basic variables values are in the rightmost column and the remaining nonbasic variable's values are zeros

Step 2: Finding the entering variable.

⇒ Select a negative entry from among the first n elements of the objective row. Mark its column to indicate the entering variable and the pivot column.

Step 3: Finding the departing variable

⇒ for each positive entry in the pivot column calculate the θ ratio by dividing that row's entry in the rightmost column by its entry in the pivot column.

Step 4: Forming the next tableau:

⇒ Divide all the entries in the pivot row by its entry in the pivot column. Subtract from each of the other rows, including the objective row, the new pivot row multiplied by the entry in the pivot column of the row in question.

Scanned by TapScanner

Extreme point Theorem

⇒ Any LP pbm with a non empty bounded feasible region has an optimal solution. An optimal solution can always be found at an extreme point of the pbm's feasible region.

3 possible outcomes in solving an LP problem

* A finite optimal solution, which may not be unique.
* Unbounded: The objective function of maximization (minimization LP pbm is unbounded from above (below) on its feasible region.
* Infeasible: There are no points satisfying all the constraints (ie) the constraints are contracting.

Summerize (or) outline of the simplex method

Step 0:

Initialization: Present a given linear programming problem in std form and set up an initial tableau with non negative entries in the rightmost column and m other columns composing the mxm identity matrix.

⇒ These m columns define the basic variables of the initial basic feasible solution, used as the labels of the tableau's rows.

Scanned by TapScanner

example.

Maximize $6x_1 + 5x_2$

Subject to $x_1 + x_2 \leq 5$

$3x_1 + 2x_2 \leq 12$

$x_1, x_2 \geq 0$ using tabular form.

Sol.

Convert inequalities to equalities by adding slack Variable. ∴ maximize. $6x_1 + 5x_2 + 0S_1 + 0S_2$.

$$\left.\begin{aligned} x_1 + x_2 + S_1 &= 5 \\ 3x_1 + 2x_2 + S_2 &= 12 \end{aligned}\right\} \quad x_1, x_2, S_1, S_2 \geq 0$$

Iteration 1

obj	Basis	6 x_1	5 x_2	0 S_1	0 S_2	RHS
0	S_1	1	1	1	0	5
0	S_2	3	2	0	1	12.
	$C_j - Z_j$	6	5	0	0	0

Compute $C_j - Z_j$

$C_1 - Z_1 = 6 - [(S_1 * x_1) + (S_2 * x_1)] = 6$

$C_2 - Z_2 = 5 - [(S_1 * x_2) + (S_2 * x_2)] = 5$

$C_3 - Z_3 = 0 - [(S_1 * S_1) + (S_2 * S_1)] = 0$

$C_4 - Z_4 = 0 - [(S_1 * S_2) + (S_2 * S_2)] = 0$

$C_5 - Z_5 = [(S_1 * 5) + (S_2 * 12)] = 0$

Identify the largest possible $C_j - Z_j$ value in iteration 1.

→ Here $C_j - Z_j$ for $x_1 = 6$

∴ x_1 enters the basis ↑

→ Find the leaving variable either S_1 or S_2. Create another column Θ.

$$\Theta = \frac{\text{RHS}}{\text{Corresponding element of entering column}}$$

$= \frac{5}{1} = 5$

$= \frac{12}{S_2} = \frac{12}{3} = 4$

$\min(5, 4) = 4$ ∴ S_2 leaves the basis →

Entering Variable $= x_1$, leaving variable $= S_2$

Scanned by TapScanner

→ Choose the pivot row & pivot element

Pivot row = S_2 = Row corresponding to leaving Variable

Pivot element = 3 = Intersection of entering column & leaving row.

→ Perform row operation - divide the pivot row by pivot element.

	x_1	x_2	S_1	S_2	RHS	
S_1	(1-1)	(1-2/3)	(1-0)	(0-1/3)	(3-1)=1	② →
x_1	1	2/3 ↑	0	1/3	1	6
$C_j - Z_j$	0	1	•	○	-2	21

After 1 iteration ②

pivot element = 1/3, pivot row = S_1 → Entering Variable = x_2

Leaving Variable = S_1

Perform Row operation = Pivot Row / Pivot Element.

	x_1	x_2	S_1	S_2	RHS
x_2	0/1/3	1/3/1/3	1/1/3	-1/3/1/3	1/1/3

x_1 ⇓

	x_1	x_2	S_1	S_2	
x_2	0	1	3	-1	3
x_1	1-2/3(0)	2/3-2/3(1)	0-2/3(3)	1/3-2/3(-1)	4-2/3(3)

⇓

	x_1	x_2	S_1	S_2	
x_2	0	1	3	-1	3
x_1	1	0	-2	1	2
$C_j - Z_j$	0	0	-3	-1	27

Since $C_j - Z_j$ has no positive values, alg terminates here.

∴ The best solution is $x_1 = 2$, $x_2 = 3$, $Z = 27$.

Scanned by TapScanner

example

Maximize $6x_1 + 5x_2$

subject to $x_1 + x_2 \le 5$

$3x_1 + 2x_2 \le 12$

$x_1, x_2 \ge 0$ using tabular form.

Sol: Convert inequalities to equalities by adding slack variables

$\therefore$ maximize $6x_1 + 5x_2 + 0S_1 + 0S_2$.

$$\left.\begin{aligned} x_1 + x_2 + S_1 &= 5 \\ 3x_1 + 2x_2 + S_2 &= 12 \end{aligned}\right\} \; x_1, x_2, S_1, S_2 \ge 0$$

Iteration 1

obj	Basis	x_1	x_2	S_1	S_2	RHS
		6	5	0	0	
0	S_1	1	1	1	0	5
0	S_2	3	2	0	1	12
$C_j - Z_j$		6	5	0	0	0

Compute $C_j - Z_j$

$C_1 - Z_1 = 6 - [(S_1 * x_1) + (S_2 * x_1)] = 6$

$C_2 - Z_2 = 5 - [(S_1 * x_2) + (S_2 * x_2)] = 5$

$C_3 - Z_3 = 0 - [(S_1 * S_1) + (S_2 * S_1)] = 0$

$C_4 - Z_4 = 0 - [(S_1 * S_2) + (S_2 * S_2)] = 0$

$C_5 - Z_5 = [(S_1 * 5) + (S_2 * 12)] = 0$

Identify the largest possible $C_j - Z_j$ value in iteration 1.

→ Here $C_j - Z_j$ for $x_1 = 6$

$\therefore x_1$ enters the basis ↑

→ Find the leaving variable either S_1 or S_2. Create another column Θ.

$$\Theta = \frac{\text{RHS}}{\text{corresponding element of entering column}}$$

$= \frac{5}{1} = 5$

$= \frac{12}{3} = 4$

$\min(5, 4) = 4$ $\therefore S_2$ leaves the basis →

Entering variable $= x_1$, Leaving variable $= S_2$

Scanned by TapScanner

→ choose the pivot row & pivot element.

Pivot row - S_2 = Row Corresponding to leaving Variable

Pivot element = 3 = Intersection of entering column & leaving row.

→ Perform row operation - divide the pivot row by pivot element.

	x_1	x_2	S_1	S_2	
S_1	(1-1)	(1-2/3)	(1-0)	(0-1/3)	(3-1)=1 →
x_1	1	2/3 ↑	0	1/3	1 6
$C_j - Z_j$	0	1	0	-2	21

After Iteration 2

pivot element = 1/3, pivot row = S_1 Entering Variable = x_2
Leaving Variable = S_1

Perform Row operation = Pivot Row / Pivot Element.

	x_1	x_2	S_1	S_2	RHS
x_2	0/1/3	1/3/1/3	1/1/3	-1/3/1/3	1/1/3

x_1 ⇓

	x_1	x_2	S_1	S_2	
x_2	0	1	3	-1	3
x_1	1	0	-2	1	2
$C_j - Z_j$	0	0	-3	-1	27

Since $C_j - Z_j$ has no positive values, alg terminates here.

∴ The best solution is $x_1 = 2$, $x_2 = 3$, $Z = 27$.

Scanned by TapScanner

(2) Explain in detail about maximum flow problem with example?

Problem Statement:

⇒Problem of maximizing the flow of a material through a transportation network.

eg) pipeline system, Communications or transportation nlws.

⇒Formally represented by a connected weighted digraph with n vertices numbered from 1 to n with the following properties:

*Contains exactly one vertex with no entering edges, called the source (numbered 1).

*Contains exactly one vertex with no leaving edges, called the sink (numbered n).

*Has positive integer weight u_{ij} on each directed edge (i, j), called the edge capacity, indicating the upper bound on the amount of the material that can be sent from i to j through that edge.

Definition of a flow:

⇒Flow is an assignment of real numbers x_{ij} to edges (i, j) of a given network that satisfy the following.

Scanned by TapScanner

* Flow - Conservation requirements

$$\sum_{j:(j,i)\in E} x_{ji} = \sum_{j:(i,j)\in E} x_{ij} \quad \text{for } i = 2, 3 \ldots, n-1$$

* The total amount of material entering an intermediate vertex must be equal to the total amount of the material leaving the vertex.

* Capacity Constraints

$$0 \le x_{ij} \le u_{ij} \text{ for every edge } (i,j) \in E$$

Since no material can be lost or added to by going through intermediate vertices of the network, the total amount of the material leaving the source must end up at the sink.

$$\sum_{j:(1,j)\in E} x_{1j} = \sum_{j:(j,n)\in E} x_{jn}$$

⇒ The value of the flow is defined as the total outflow from the source

⇒ The maximum flow pbm is to find a flow of the largest value for a given network.

Maximum - Flow pbm as LP problem.

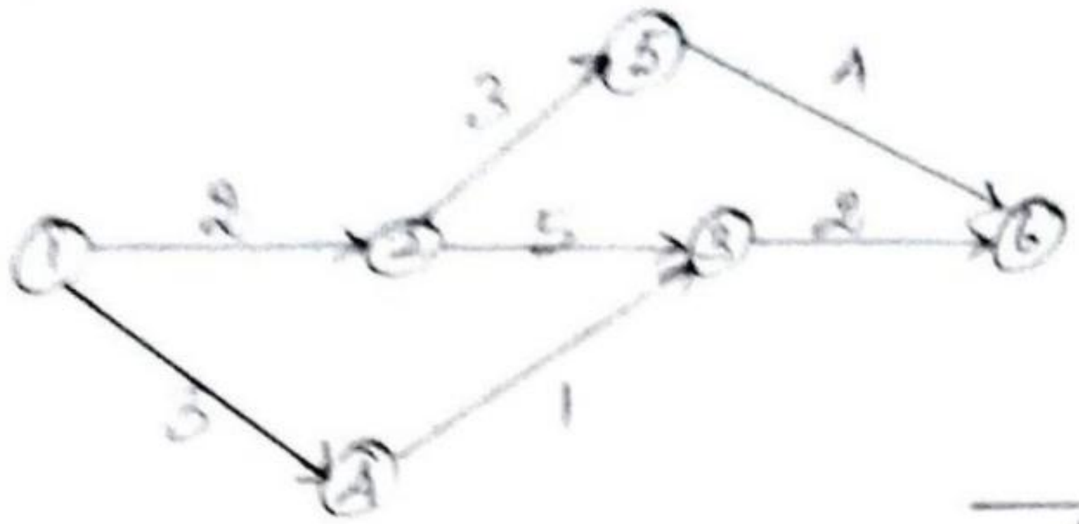
$$\text{Maximum } v = \sum_{j:(1,j)\in E} x_{1j}$$

$$\text{subject to } \sum_{j:(j,i)\in E} x_{ji} - \sum_{j:(i,j)\in E} x_{ij} = 0 \text{ for } i = 2, 3 \ldots, n-1$$

$$0 \le x_{ij} \le u_{ij} \text{ for every edge } (i,j) \in E.$$

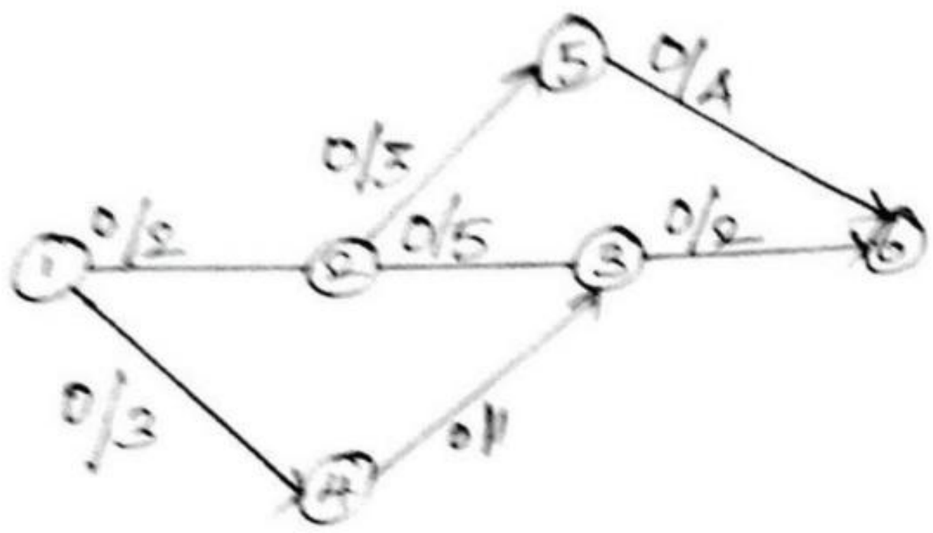
Scanned by TapScanner

Example:



Path	flow
1→2→3→6	2
1→4→3←2→5→6	1

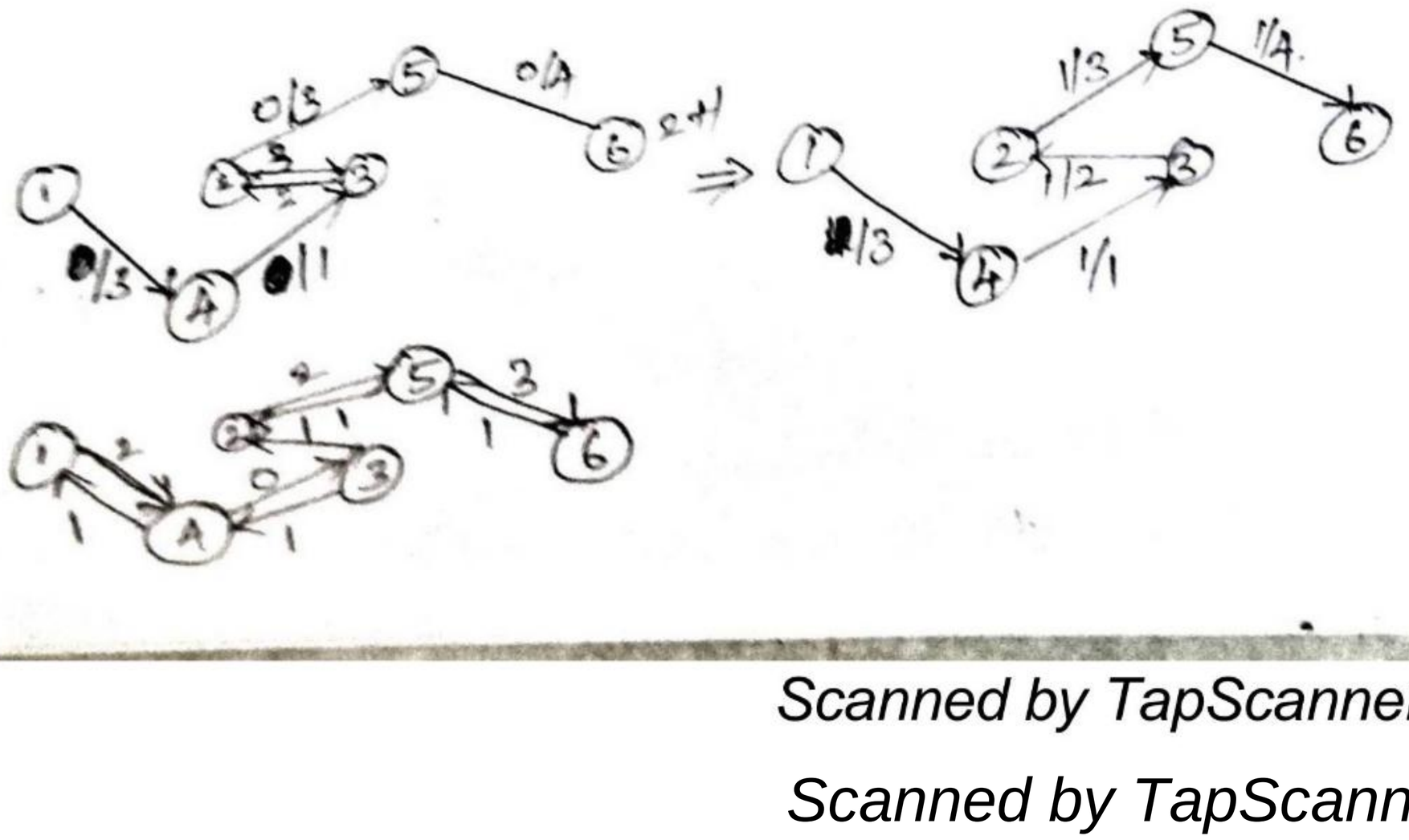
Sol:



Step 2: Augmenting path: 1 → 2 → 5 → 6

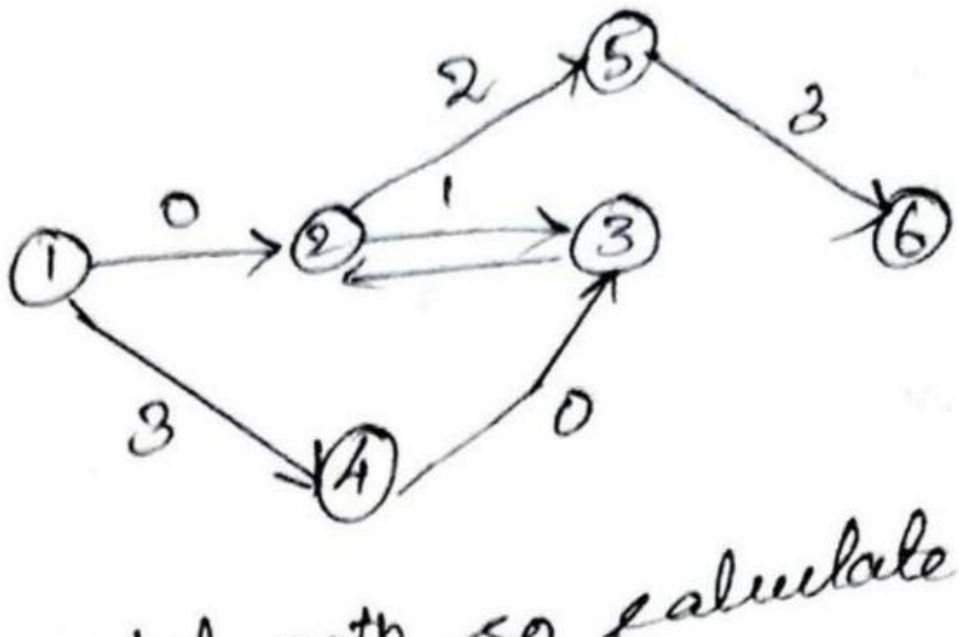
Note:
1 —2/2→ 2 (ie) 2-2=0
if the capacity is zero then the path will be blocked.

Step 3 Augmenting path: 1 → 4 → 3 → 6 → 5 → 6



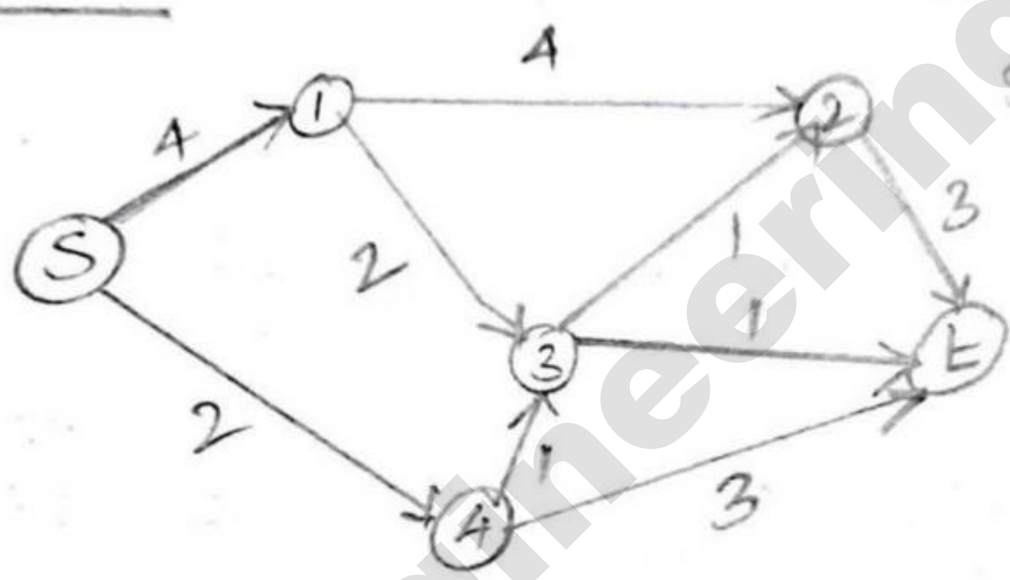
Scanned by TapScanner

Step 5



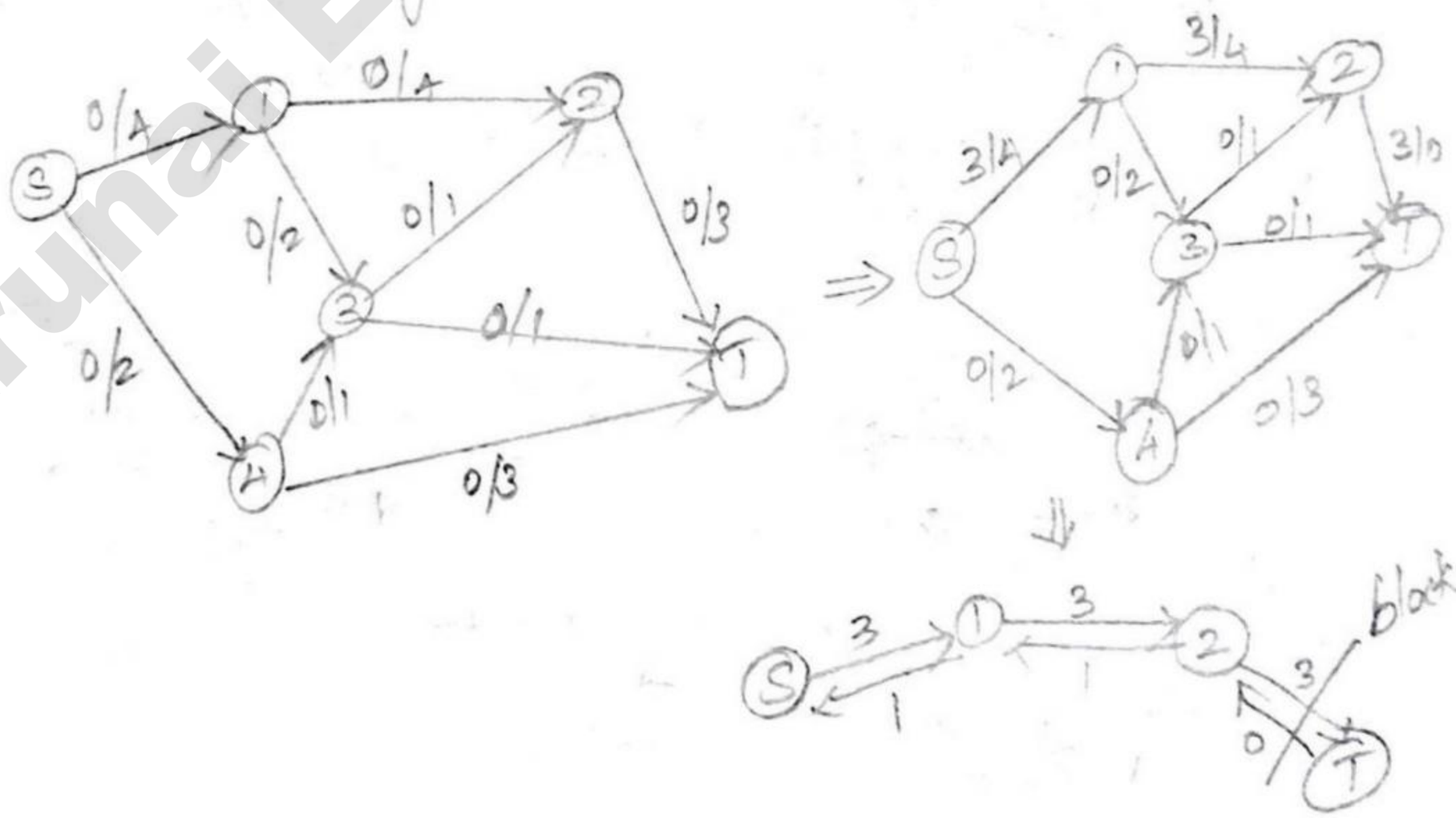
∴ No augmented path so calculate maximum flow so maximum flow = 2+1 = 3

Example : 2 (University pbm)



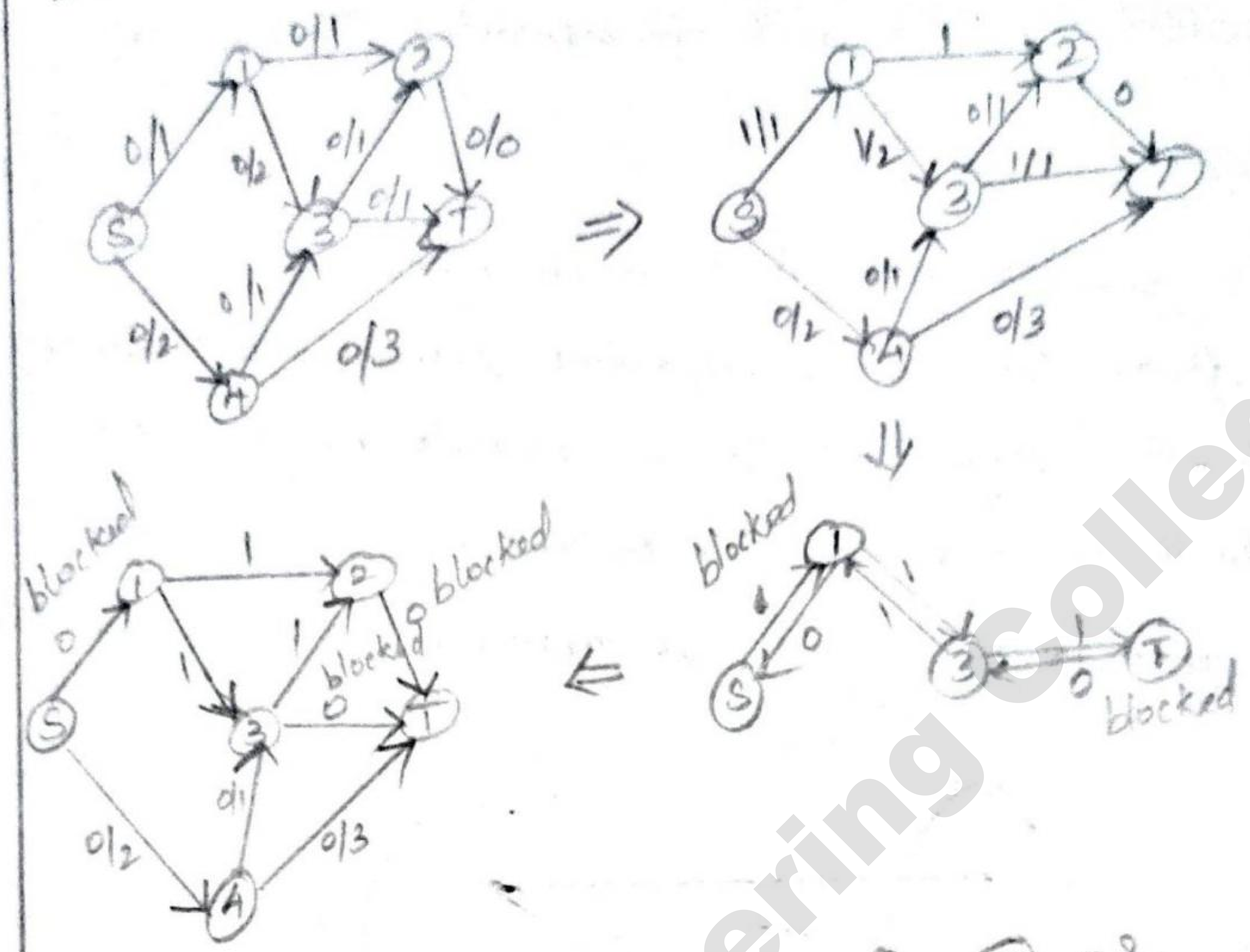
Path	flow
S→1→2→T	3
S→1→3→T	1
S→4→T	2
	6

Step:1

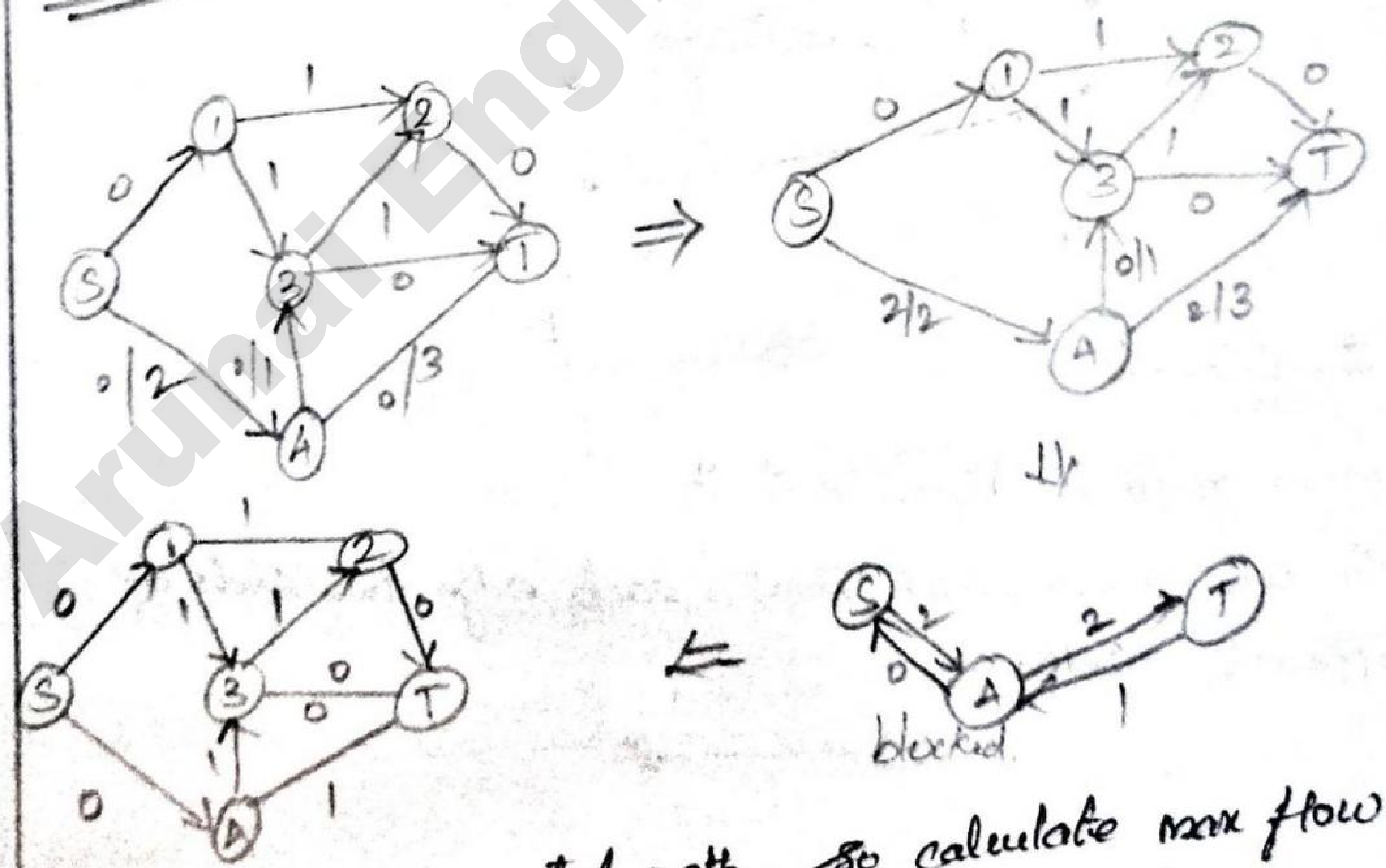


Scanned by TapScanner

Step 2 Augmenting path : S → 1 → 3 → T



Step 3 Augmented path = S → A → T = 2



∴ No augmented path. So calculate max flow.

3+1+2 ⇒ 6.

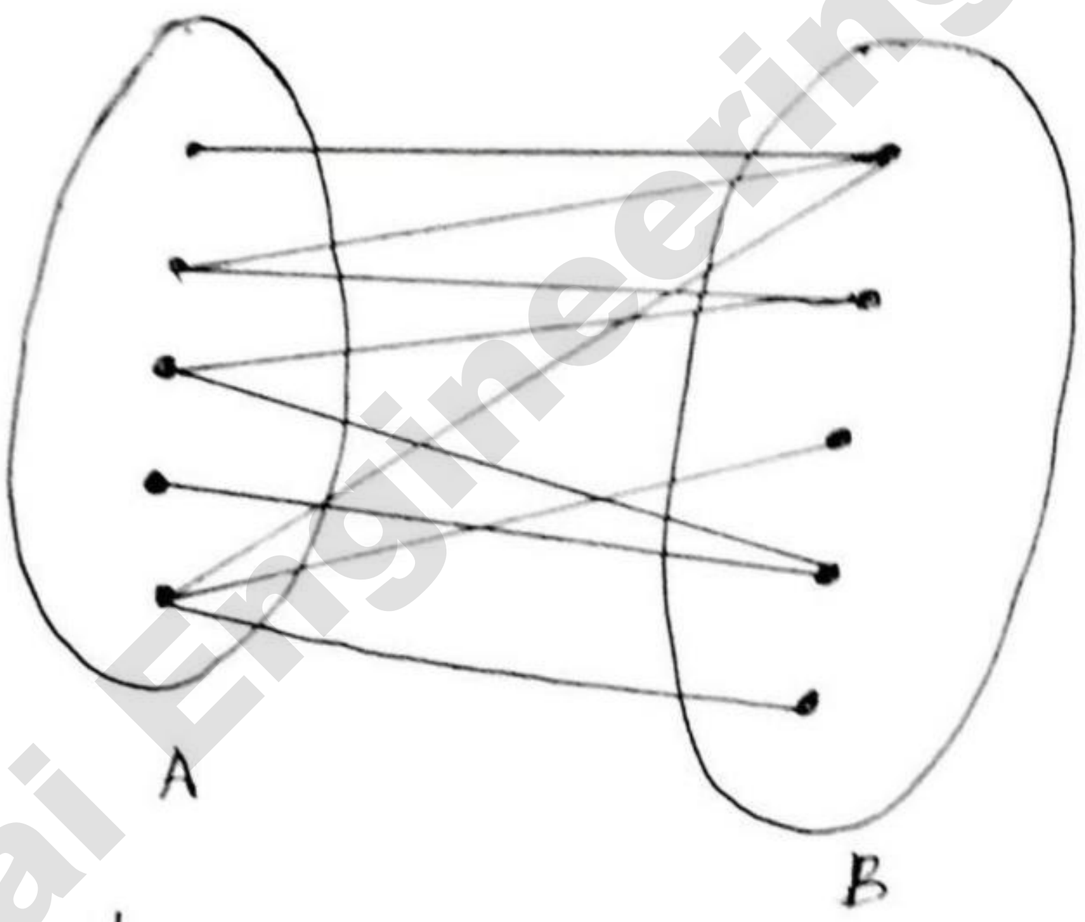
Scanned by TapScanner

(3) What is a g bipartite graph? Is the secheet of a bipartite ? outline with an example? [NOV/Dec '19]

Definition

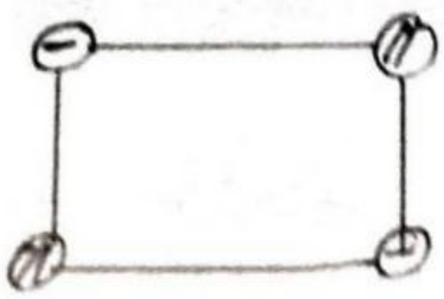
⇒ A Bipartite graph is a graph whose vertices can be divided into two independent sets A and B such that every edge (a,b) either connects a vertex from A to B or a vertex from B to A.

⇒ There is no edge that connects vertices of sameset.



2-Colorable

⇒ A graph is bipartite if its vertices can be colored with two colors such that each edge has ends of different colors.



Scanned by TapScanner

Matching:

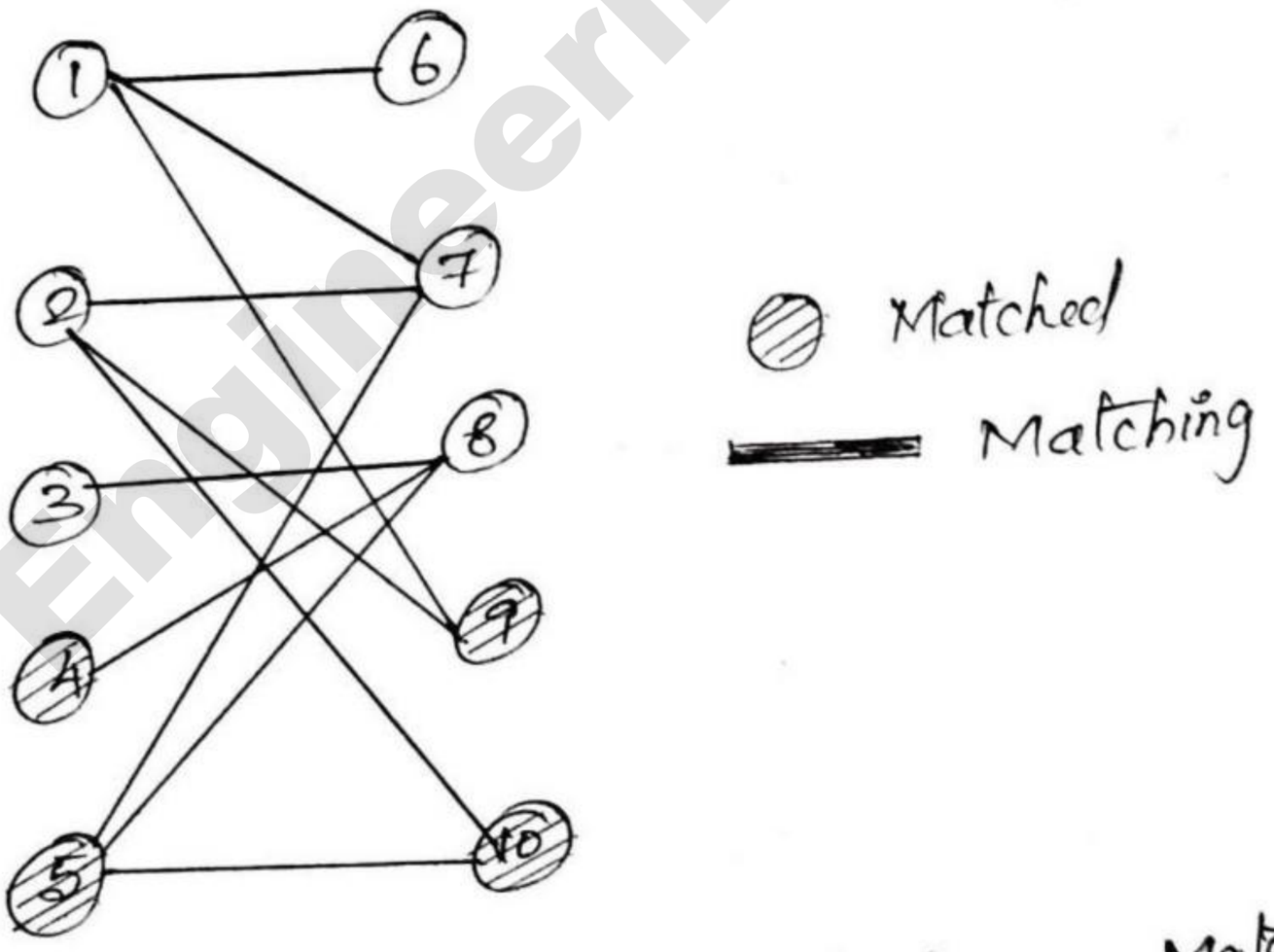
⇒ In a graph G = (V, E), a matching M ⊆ E is a collection of edges such that every vertex of V is incident to atmost one edge of M.

Unmatched / free vertices:

⇒ If a vertex V has no edge of M incident to it, then V is said to be unmatched.

⇒ A Matching is perfect if no vertex is unmatched

ii) if its cardinality is equal to |A| = |B|



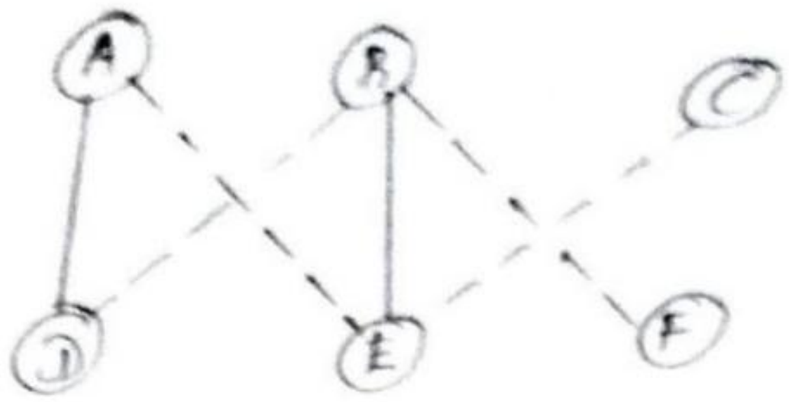
⇒ In fig edges (1, 6), (2, 7) and (3, 8) form a Matching

Vertices A 5, 9, 10 are unmatched.

Maximum cardinality matching problem:

⇒ Let us apply the iterative - improvement technique to this pbm, to find a matching M of max size.

Scanned by TapScanner



Bipartite Graph.

Solid edges are matching. (maximal but not maximum)

$M = \{\{A,D\},\{B,E\}\}$

Maximal Matching ⇒ In one to which no additional edge can be added.

⇒ for the same graph, another matching could also be performed to get maximum.

$M = \{\{A,D\}\{B,F\}\{C,E\}\}$

∴ The efficient matching algorithm is achieved through iterative improvement.

Edmod's Algorithm:

step 1: M = empty matching

step 2: While there is an augmenting path P for M, M = M ⊕ P.

step 3: o/p M.

Alternating Path: A path whose edges are alternately in and out of the matching.

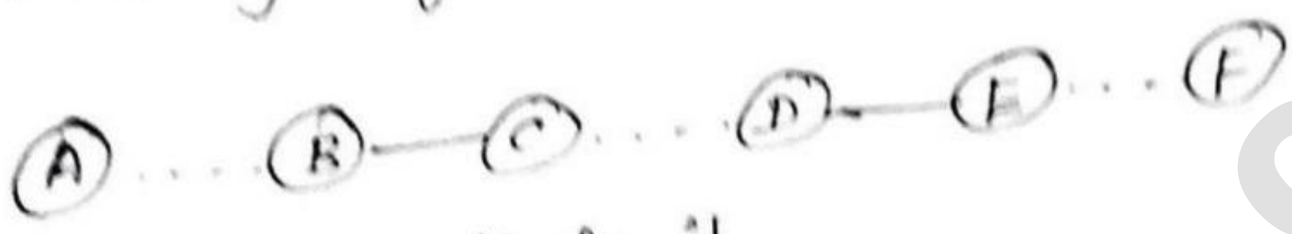
Scanned by TapScanner

Augmenting path:

⇒ An alternating path between two unmatched (free) Vertices.

Augmentation

⇒ Given an augmenting path, change its unmatched edges to matched and vice versa, thereby increasing size of the matching by one.



A, F are free

∴ It is augmenting path.

A, F are matched now. It is said to be augmentation

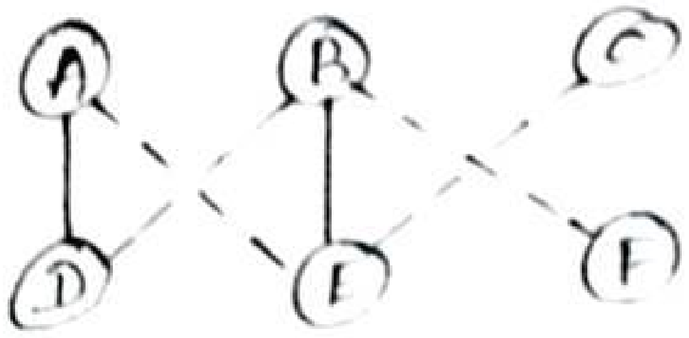
Augmenting path Algorithm

* Start with the empty matching.
* While there is an augmenting path, do an augmentation.

Theorem:

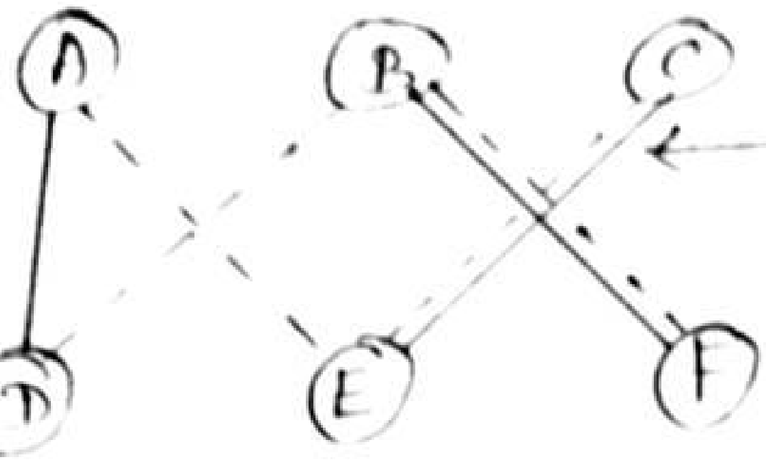
⇒ A matching M is maximal iff there is no augmenting path with respect to M.

Scanned by TapScanner



$M_a = \{(A,D), (B,E)\}$

Find augmenting path?

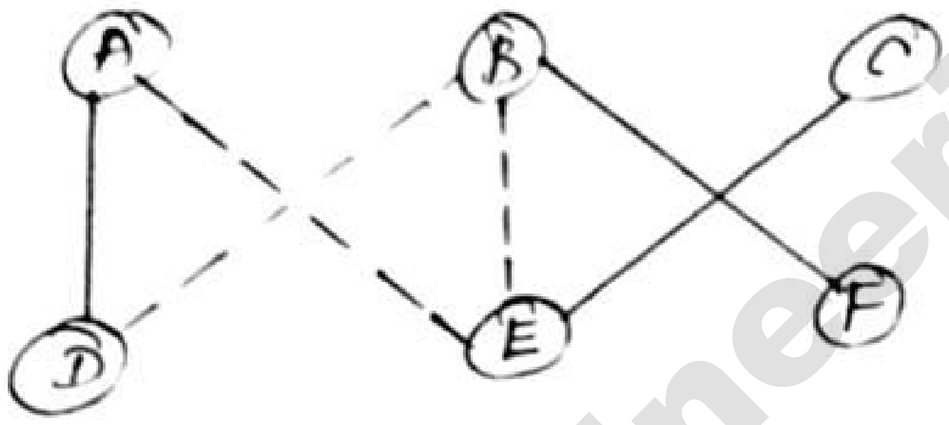


Augmenting path

C ---- E —— B ---- F

After Augmenting ⇓ Flip it

C —— E ---- B —— F



$M_b = \{(A,D), (C,E), (B,F)\}$

To prove:

⇒ If no augmenting path with respect to a matching M exists, then the matching is maximal.

AEC

④ What is stable marriage problem? Give the alg and analyse it. { [APR/May'15] [NOV/Dec'16] [NOV/Dec'17] [APR/May'18] [NOV/Dec'18] [NOV/Dec'19]

Definition

Problem Statement

⇒ There is an set $Y = \{m_1 \ldots m_n\}$ of n men and a set $X = \{w_1, \ldots, w_n\}$ of n women. Each man has a ranking list of the women and each woman has a ranking list of the men.

⇒ A marriage matching M is a set of n pairs (M_i, W_j).

⇒ A pair (m, w) is said to be a blocking pair for matching M if man m and woman w are not matched in M but prefer each other to their mates in M.

⇒ A marriage matching M is called stable if there is no blocking pair for it; otherwise, it's called unstable.

⇒ The stable marriage pbm is to find a stable marriage matching for men's and women's given preferences.

Scanned by TapScanner

Instance of the Stable Marriage Problem

⇒ An instance of the stable marriage pbm can be specified either by two sets of preference lists or by a ranking matrix, as in the example below.

men's preferences

	1st	2nd	3rd
Bob :	Lea	Ann	Sue
Jim :	Lea	Sue	Ann
Tom :	Sue	Lea	Ann

Women's preferences

	1st	2nd	3rd
Ann:	Jim	Tom	Bob
Lea :	Tom	Bob	Jim
Sue :	Jim	Tom	Bob

Ranking Matrix

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

{(Bob, Ann) (Jim, Lea) (Tom, Sue)} is unstable

{(Bob, Ann) (Jim, Sue) (Tom, Lea)} is stable

Stable Marriage Algorithm (Gale - Shapley)

1. Start with all the men and women being free
2. While there are free men, arbitrarily select one of them and do the following.

Scanned by TapScanner

Proposal:

⇒ The selected free man m proposes to w, the next woman on his preference list.

Response:

⇒ If w is free, she accepts the proposal to be matched with m. If she is not free, she compares m with her current mate. If she prefers m to him, she accepts m's proposal, making her former mate free; otherwise, she simply rejects m's proposal, leaving m free.

3. Return the set of n matched pairs.

Example:

Free men:
Bob, jim, Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Bob proposed to Lea
Lea accepted

Free men:
jim, Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

jim proposed to Lea
Lea rejected.

Scanned by TapScanner

Free men:
jim, Tom

	Ann	Lea	Sue
Bob	2,3	[1,2]	3,3
jim	3,1	1,3	[2,1]
Tom	3,2	2,1	1,2

jim proposed to sue.
sue accepted.

Free men
Tom

	Ann	Lea	Sue
Bob	2,3	[1,2]	3,3
Jim	3,1	1,3	[2,1]
Tom	3,2	2,1	1,2

Tom proposed to Sue but sue rejected.
(ii) Tom proposed to Lea. Lea also rejected.

free men
Tom

	Ann	Lea	Sue
Bob	2,3	[1,2]	3,3
Jim	3,1	1,3	[2,1]
Tom	[3,2]	2,1	1,2

Tom proposed to Ann.
Ann accepted.

Analysis of the Gale-Shapley Algorithm

⇒ The algorithm terminates after no more than n^2 iterations with a stable marriage output.

⇒ The stable matching produced by the alg is always man-optimal.

⇒ A man (woman) optimal matching is unique for a given set of preparticipant preferences.

⇒ The stable marriage pbm has practical applications such as assigning graduating medical students to their first hospital appointments.

Scanned by TapScanner

5) State and Prove Max-flow Min-Cut Theorem:

[may/June '16]

Definition of a Cut

⇒ Let X be a set of vertices in a network that includes its source but does not include its sink, and let $\bar{X}$, the complement of X, be the rest of the vertices including the sink.

⇒ The cut induced by this partition of the vertices is the set of all the edges with a tail in X and a head in $\bar{X}$.

⇒ Capacity of a cut is defined as the sum of capacities of the edges that compose the cut.

* The cut and its capacity is denoted by $c(X, \bar{X})$ and $c(\bar{X}, X)$.

* Note that if all the edges of a cut were deleted from the n/w, there would be no directed path from source to sink.

* Minimum cut is a cut of the smallest capacity in a given network.

Max-flow min cut theorem:

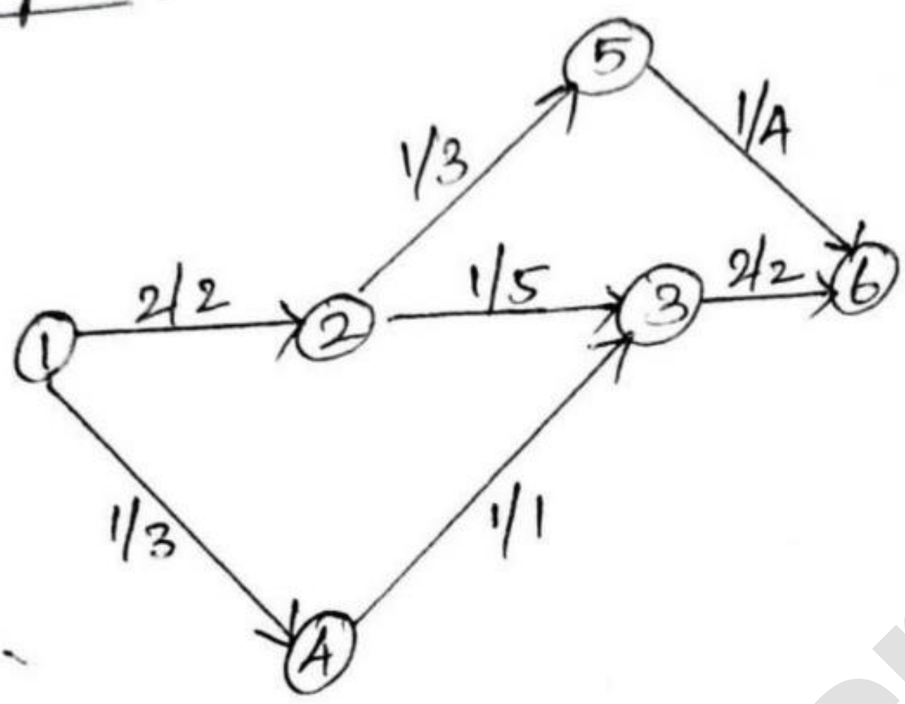
⇒ The value of maximum flow in a n/w is equal to the capacity of its minimum cut.

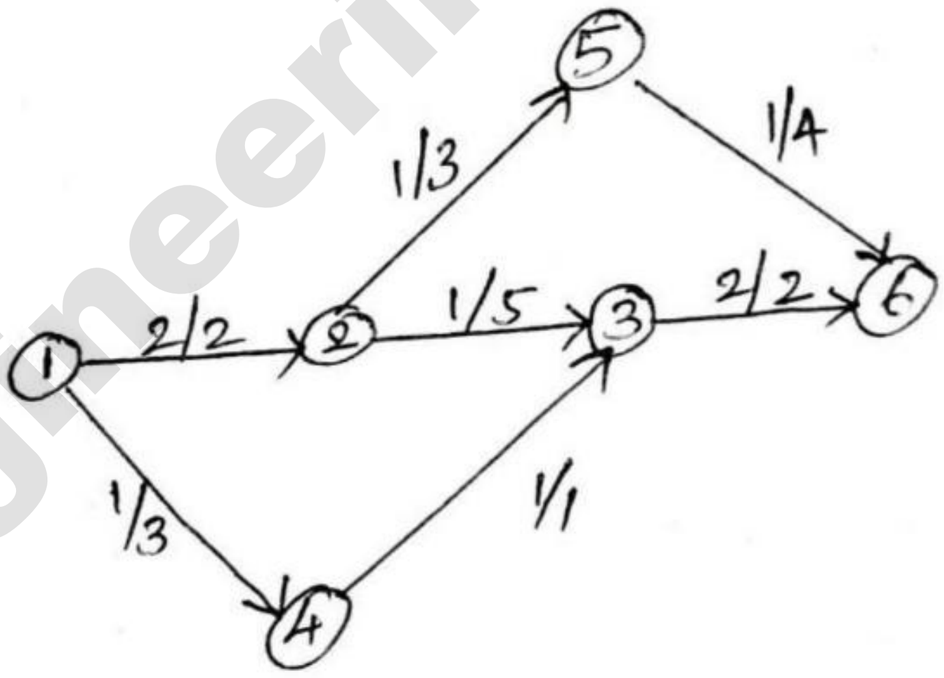
Scanned by TapScanner

⇒ The shortest augmenting path alg yields both a minimum cut:

* Max flow is the final flow produced by the alg
* Min cut is formed by all the edges from the labeled vertices to unlabeled vertices on the last iteration of the alg.

Examples





Queue = 1 4
↑ ↑

⇒ No augmenting path (the sink is unlabled)
the current flow is maximum.

Scanned by TapScanner

UNIT V

Scanned by TapScanner

UNIT-V

① When is a problem said to be NP hard?

⇒ A problem is said to be NP-hard if everything in NP can be transformed in polynomial time into it; and a problem is NP complete if it is both in NP and NP-hard.

② Show the purpose of lower bound.

⇒ Lower bound of a problem is an estimate on a minimum amount of work needed to solve a given problem.

③ State the Hamiltonian circuit problem.

⇒ A Hamiltonian path is a path in an undirected graph which visits each vertex exactly once.

⇒ A Hamiltonian circuit is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex.

4) Define NP Completeness and NP hard.

NP Completeness

* If a polynomial time alg exists for any of these problems all problems in NP would be polynomial time solvable. These problems are called NP-complete.

Scanned by TapScanner

Np. Hard.

* A problem is Np-hard if all problems in Np are polynomial time reducible to it, even though it may not be in Np itself.

5) Give an example for sum of subset problem.

Input : Set [ ] = {3, 34, 4, 12, 5, 2} sum=9.

output : True // There is a subset (4,5) with sum 9.

6) Define p and NP problems.

P : In computational complexity theory p also known as PTIME (or) DTIME (n). is a fundamental Complexity class.

⇒ It contains all decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time or polynomial time.

NP : The class of decision problems that are solvable in polynomial time on a nondeterministic machine

⇒ A nondeterministic computer is one that can guess the right answer or solution think of a nondetermini -stic computer as a parallel machine that can freely spawn an infinite number of process.

Scanned by TapScanner

7) How is lower bound found by problem reduction?

⇒ If problem P is at least as hard as problem Q, then a lower bound for Q is also a lower bound for P. Hence, find problem Q with a known lower bound that can be reduced to problem P in question. Then any algorithm that solves P will also solve Q.

6) What are tractable and non tractable problems?

⇒ Problems that are solvable by polynomial time algorithms as being tractable.

⇒ Problems that require super polynomial time as being intractable.

a) Define 0/1 knapsack problem.

⇒ The solution to the knapsack pbm can be viewed as a result of sequence of decisions.

⇒ To decide the value of x_i. x_i is restricted to have the value 0 or 1 and by using the function knap(1, j, y). we can represent the pbm as max $\sum p_i x_i$ subject to $\sum w_i x_i \leq y$ where 1 - iteration, j - no. of objects y - capacity.

Scanned by TapScanner

10) List Some applications of travelling salesperson problem

* Routing a postal van to pickup mail from boxes located at n different sites.
* Using a robot arm to tighten the nuts on some piece of machinery on an assembly line.
* Production environment in which several commodities are manufactured on the same set of machines.

11) Define backtracking?

* Depth first node generation with bounding function is called backtracking. The backtracking algorithm has its virtue the ability to yield the answer with far fewer than m trials.

12) what is Hamiltonian cycle in an undirected graph?

* A Hamiltonian cycle is round trip along n edges of G that visits every vertex once and returns to its starting position.

13) what is feasible solution?

* It is obtained from if given n inputs.
* Subsets that satisfies some constraints are called feasible solution.
* It is obtained on some constraints.

Scanned by TapScanner

14) What is optimal solution?

* It is obtained from feasible solution.
* feasible solution that maximizes or minimizes a given objective function.
* It is obtained based on objective function.

15) What is heuristic?

⇒ A heuristic is a common sense rule drawn from experience rather than from a mathematically proved assertion.

for eg, going to the nearest unvisited city in the travelling salesman pbm is good example for heuristic.

16) What is promising and non promising node?

* A node in a state space tree is said to be promising, if it corresponds to a partially constructed solution that may still lead to a complete solution. otherwise, a node is called non-promising.

17) Features of Heuristics algorithm.

1) Develop intuitive algorithms
2) Guaranteed to run in polynomial time.
3) No guarantees on quality of solution

Scanned by TapScanner

18) What is meant by approximation algorithm.

⇒ Given an optimization problem P. an algorithm A is said to be an approximation solution algorithm for P.

⇒ If for any given instance I, it returns an approximate solution, that is feasible solution

19) Features of Approximation Algorithm.

1. Guaranteed to run in polynomial solution.
2. Guaranteed to get a solution which is close to the optimal solution.

20) What are the searching techniques that are commonly used in Branch-and-Bround method.

① FIFO
② LIFO
③ Heuristic Search.

AEC

(1) Elaborate how backtracking technique can be used to solve the n-queens problem. Explain with an example.

Problem Statement:

⇒ The n-queens problem is a generalization of the 8 queen (or) 4-queen problem. Now n queens are to be placed on an n×n chessboard so that no two queens attack each other; that is no two queens are on the same row, column or diagonal.

The strategy

⇒ The chessboard square is considered as two dimensional array a[1:n, 1:n].

⇒ place (k, i) returns a boolean value that is true if the k^{th} queen can be placed in column i. It tests two conditions whether i is distinct from all previous values x[1] ..... x[k-1] and whether there is no other queen on the same diagonal.

Algorithm Nqueens (k, n) // solution to n-queen problem.
// using backtracking this procedure prints all possible placements of n queens on an n×n chessboard so that they are nonattacking.

```
{
  for i=1 to n do
    {
      if place (k, i) then
```


Scanned by TapScanner

```
{
    x[k] = i;
    if (k = n) then write (x[1:n]);
    else Nqueens (k+1, n);
  }
} Nqueens (k-1, n); i = x[k-1]+1;
}
```

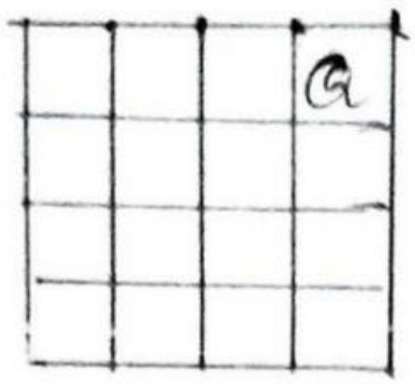
Algorithm place (k, i) // To place a new queen in the chessboard;

⇒ Returns true if a queen can be placed in k^{th} row and i^{th} column. Otherwise it returns false. x[ ] is a global array whose first (k-1) values have been set. Abs(r) returns the absolute value of r.

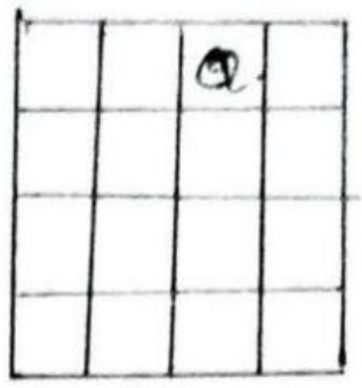
```
{
    for j = 1 to k-1 do
        if ((x[j] = i)
            or (Abs(x[j] - i) = Abs(j - k)))
        then return false;
    return true;
}
```

Example : A queens problem.

(i)



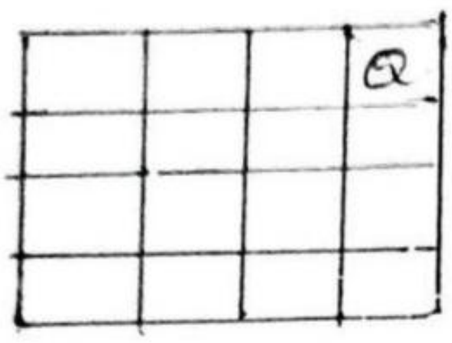
(ii)



Sol

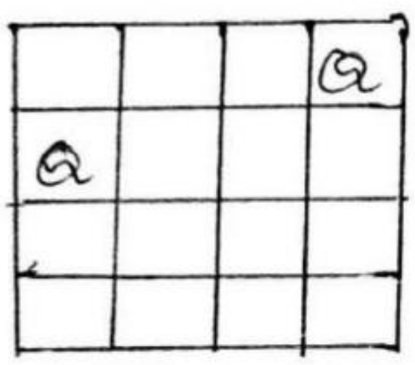
(i)

(a)



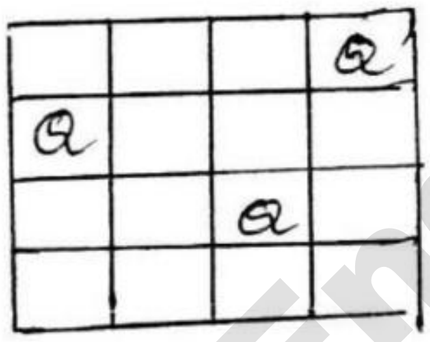
Given : Queen 1 is placed in (1,4)

(b)



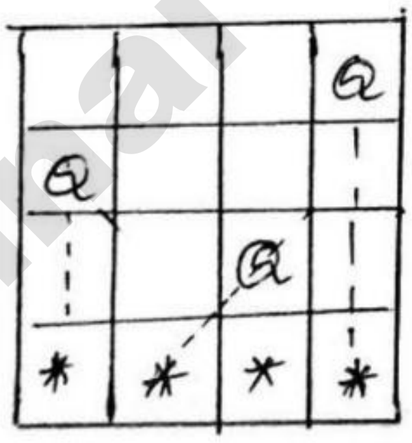
Queen 2 is placed in (2,1).

(c)



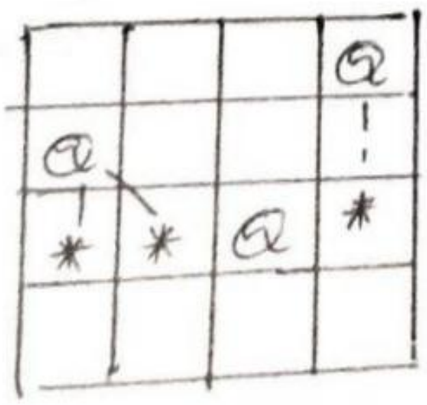
Queen 3 is placed in (3,3)

(d)



The dotted lines and * represents the failure occurrence of placing the queen 4 in the row 4. So backtracking is done to find another position for queen 3.

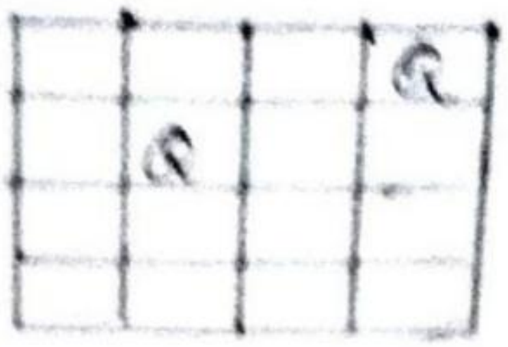
(e)



⇒ The only possible place where the queen 3 can be placed is (3,3) from which queen 4 can't be placed. So backtracking is done to find another position for queen 2.

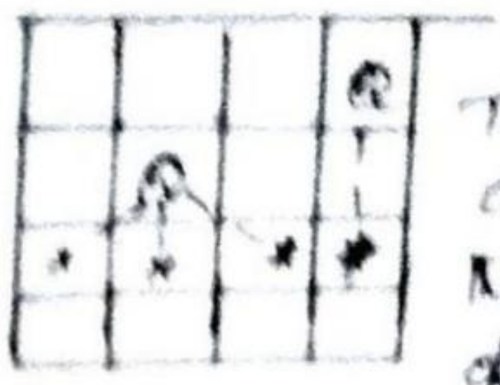
Scanned by TapScanner

(f)



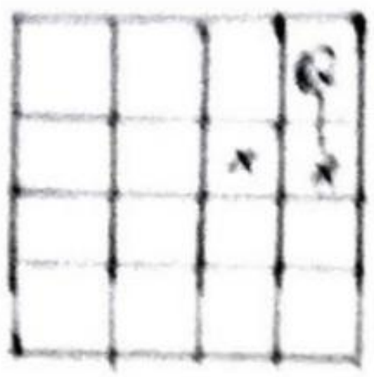
Queen 2 is placed in (2,2)

(g)



The dotted lines and * represents the failure of placing the queen 3.

(h)



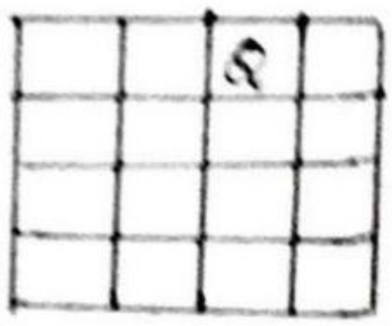
The other two positions (ie (2,3) & (3,4)) represents the failure of placing the queen 2.

Conclusion:

The solution is not exist for the given 4 queen problem

ii)

(a)

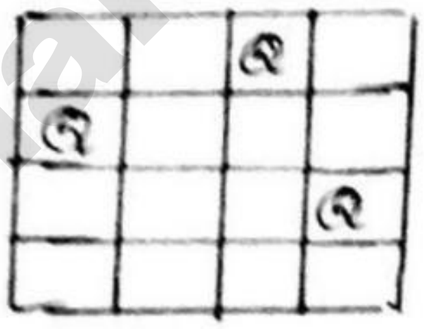


Given: queen 1 is placed in (1,3)

(b)

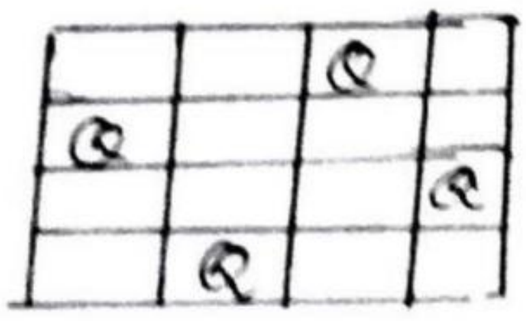
Queen 2 is placed in (2,1)

(c)



Queen 3 is placed in (3,4)

(d)



Queen A is placed in (1,4)

Scanned by TapScanner

(2) Outline the steps to find an approximate solution to NP-hard optimization problems using approximation algorithms with an example.

Approximation Algorithms for NP-hard problems:

⇒ Simplest approximation algorithms for the TSP are based on the greedy technique.

Nearest-neighbor algorithm:

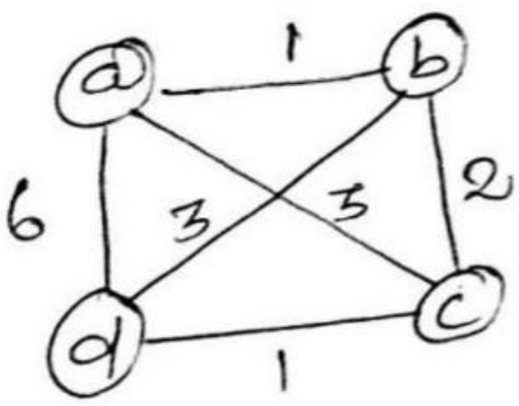
⇒ The following well-known greedy alg is based on the nearest-neighbor heuristic. always go next to the nearest unvisited city.

Step 1: Choose an arbitary city as the start

Step 2: Repeat the following operation until all the cities have been visited - go to the unvisited city nearest the one visited last.

Step 3: Return to the starting city.

eg:



⇒ with 'a' as the starting vertex, the nearest-neighbor algorithm yields the tour.

a-b-c-d-a of length 10.

Scanned by TapScanner

⇒ the optimal sol, as can be easily checked by exhaustive search, is the tour a-b-d-c-a of length 8.

⇒ the accuracy ratio of this approximation is

$$r(s_a) = \frac{f(s_a)}{f(s^*)} = \frac{10}{8} = 1.25$$

⇒ Unfortunately, except for the simplest, not many good things can be said about the nearest-neighbor algorithm.

Multifragment – heuristic algorithm.

⇒ Another greedy alg for the TSP considers it as the problem of finding a minimum-weight collection of edges in a given complete weighted graph so that all the vertices have deg 2.

⇒ An application of the greedy technique to this problem leads to the following alg.

Step 1: Sort the edges in increasing order of their weights. Initialize the set of tour edges to be constructed to the empty set.

Step 2: Repeat this step n times, where n is the no. of cities in the instance being solved: add the next edge on the sorted edge list to the set of tour edges, provided

Scanned by TapScanner

This addition does not create a vertex of degree 2 or a cycle of length less than n; otherwise skip the edge.

Step 3: Return the set of tree edges.

⇒ As an example, applying the alg to the graph in given fig yields {(a,b), (a,d), (b,c), (c,d)}. This set of edges forms the same tree as the one produced by the nearest-neighbor alg.

⇒ In general, the multifragment-heuristic alg tends to produce significantly better than the nearest-neighbor alg. But the performance ratio of the multifragment-heuristic alg is also unbounded.

⇒ There is very important subset of instances, called euclidean, for which we can make a non trivial assertion about the accuracy of both the nearest-neighbor and multifragment heuristic alg.

⇒ These are the instances in which intercity distances satisfy the following natural conditions. ie

⇒ Triangle inequality $d[i,j] \leq d[i,k] + d[k,j]$ for any triple of cities i, j, k.

Scanned by TapScanner

⇒ Symmetry $d[i,j] = d[j,i]$ for any pair of cities i & j.

⇒ A substantial majority of practical applns of TSP are its Euclidean instances.

⇒ They include in particular geometric ones, where cities correspond to points in the plane & distances are computed by the std Euclidean formula.

⇒ Although the performance ratio's of the nearest neighbor and multifragment heuristic algorithms remain unbounded for euclidean instances, their accuracy ratio satisfy the following inequality for any such instance with $n \geq 2$ cities.

$$\frac{f(s_a)}{f(s^*)} \leq \frac{1}{2}\left(\lceil \log_2 n \rceil + 1\right)$$

⇒ where $f(s_a)$ and $f(s^*)$ are the lengths of the heuristic tour & shortest tour resp.

Scanned by TapScanner

(3) Write an algorithm for subset sum and explain with an example.

SUBSET - SUM PROBLEM:

1. In Computer Science, the subset sum problem is an important problem in complexity theory and cryptography.

2. Given a set of integers, does the sum of some non-empty subset equal exactly zero.

eg) Given the set $\{-7, -3, -2, 5, 8\}$, the answer is YES because the subset $\{-3, -2, 5\}$ sums to zero.

Problem Statement:

Let, $S = \{S_1 \ldots S_n\}$ be a set of n positive integers, then we find a subset whose sum is equal to given positive integer d.

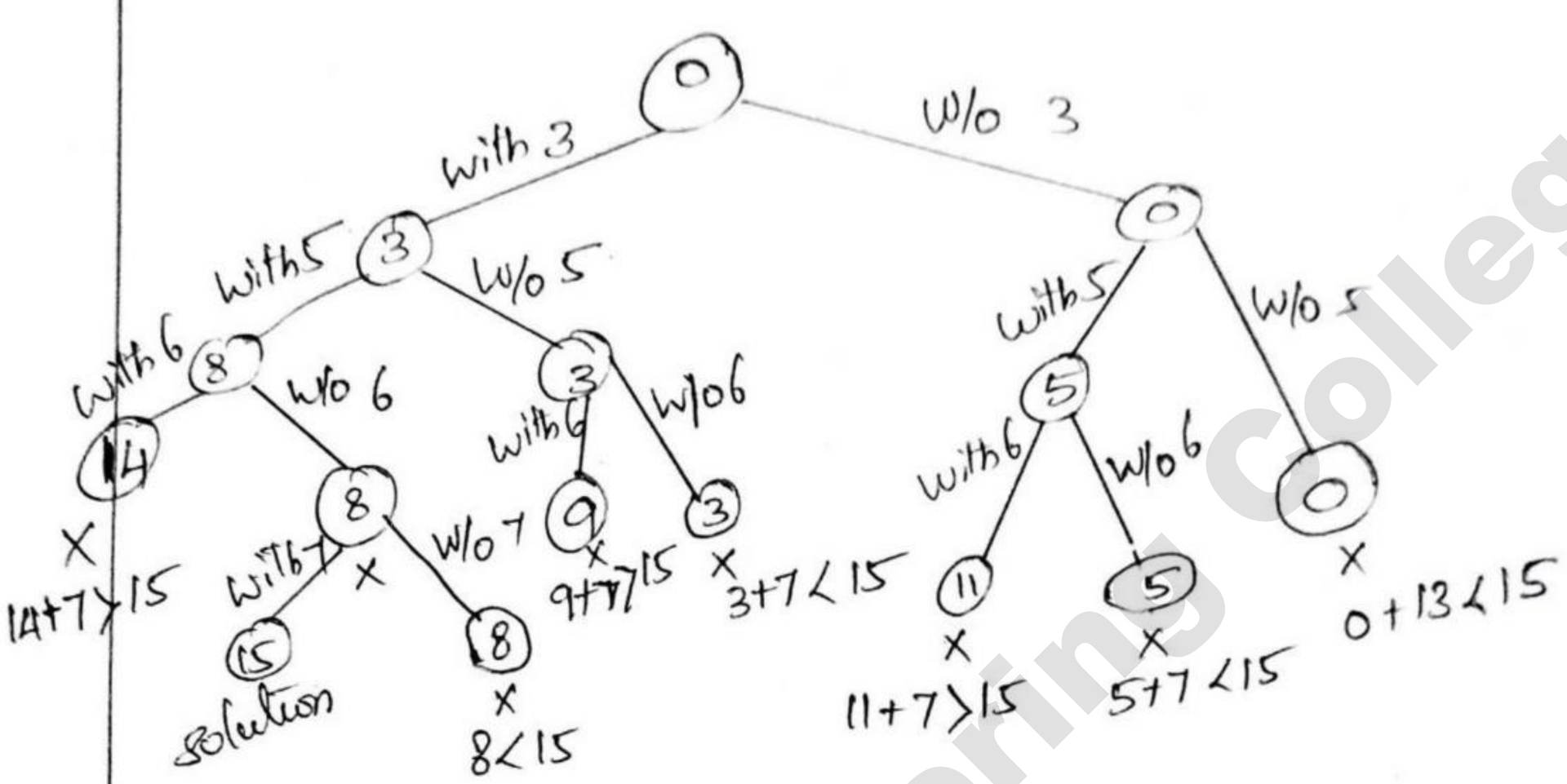
eg: $S = \{1, 2, 5, 6, 8\}$ and $d = 9$, there are two solutions: $\{1, 2, 6\}$ and $\{1, 8\}$.

→ It is always convenient to sort the sets elements in ascending order.

$S_1 \leq S_2 \leq S_3 \ldots\ldots \leq S_n$.

Scanned by TapScanner

⇒ The state space tree can be constructed as a binary tree for instance $S = \{3, 5, 6, 7\}$ and $d = 15$.



Procedure:

Step 1: The root of the tree represents the starting point, with no decisions about the given elements.

Step 2: Its left and Right children represent, inclusion and exclusion of S_1 in the set being sought.

Step 3: Going to the left from a node of the first level corresponds to inclusion of S_2 ~~in the set being sought~~ while going to right corresponds to exclusion.

Scanned by TapScanner

step 4: A path from the root to a node at the i^{th} level of the tree indicates which of the first i numbers have been included in the subsets represented by that node.

step 5: We record the value of s', the sum of these numbers in the node. If s' is equal to d, we have a solution to the pbm and stop. If all the solutions need to be found, continue by backtracking to the node's parent. If s' is not equal to d, we can terminate the node as nonpromising if either of the equalities holds.

$s' + s_{i+1} > d$ (the sum s' too large)

$s' + \sum_{j=i+1}^{n} s_j < d$ (the sum s' too small).

ALGORITHM

```
void SumOfSub (float s, int k, float r)
// Find all subsets of w[1:n] that sum to m. The
// values of x[j], 1 <= j < k, have already been determined.
// s = sum_{j=1}^{k-1} w[j] * x[j] and r = sum_{j=k}^{n} w[j].
```


Scanned by TapScanner

```
// The w[j]'s are in nondecreasing order.
// It is assumed that w[i] <= m and  sum_{i=1}^{n} w[i] >= m.
{
  //Generate left child. Note that s+w[k] <= m
  //because B_{k-1} is true.
  x[k] = 1;
  if (s+w[k]==m)
    { // subset found.
      for (int j=1; j<=k; j++) cout << x[j] << " ";
      cout << endl;
    }
  // There is no recursive call here
  // as w[j] > 0, 1 <= j <= n.
  else if (s+w[k]+w[k+1] <= m)
    SumOfSub[s+w[k], k+1, r-w[k]);
  // Generate right child and evaluate Bk.
  if ((s+r-w[k] >= m) && (s+w[k+1] <= m))
    {
      x[k]=0;
      SumOfSub (s, k+1, r-w[k]);
    }
}
```


Scanned by TapScanner

4) Find the optimal solution using Best Branch and Bound for the following assignment problem.

	Job 1	Job 2	Job 3	Job 4
Person A	9	2	7	8
Person B	6	4	3	7
Person C	5	8	1	8
Person D	7	6	9	4

Definition : Branch and Bound

* A branch and bound algorithm computes a number (bound) at a node to determine whether the node is promising.
* The no. is a bound on the value of the solution that could be obtained by expanding beyond the node.
* If the bound is no better than the value of the best solution found so far, the node is non-promising, otherwise it is promising.

ASSIGNMENT PROBLEM

Problem statement

⇒ There are n people to whom n jobs are assigned so that total cost of the assignment is as small as possible.

Scanned by TapScanner

⇒ The assignment pbm is specified by n by n matrix.
Select one element in each row of the cost matrix C
* no two selected elements are in the same column.
* Their sum is minimized as possible.

⇒ From the given problem, we select minimum value from each row then we get,

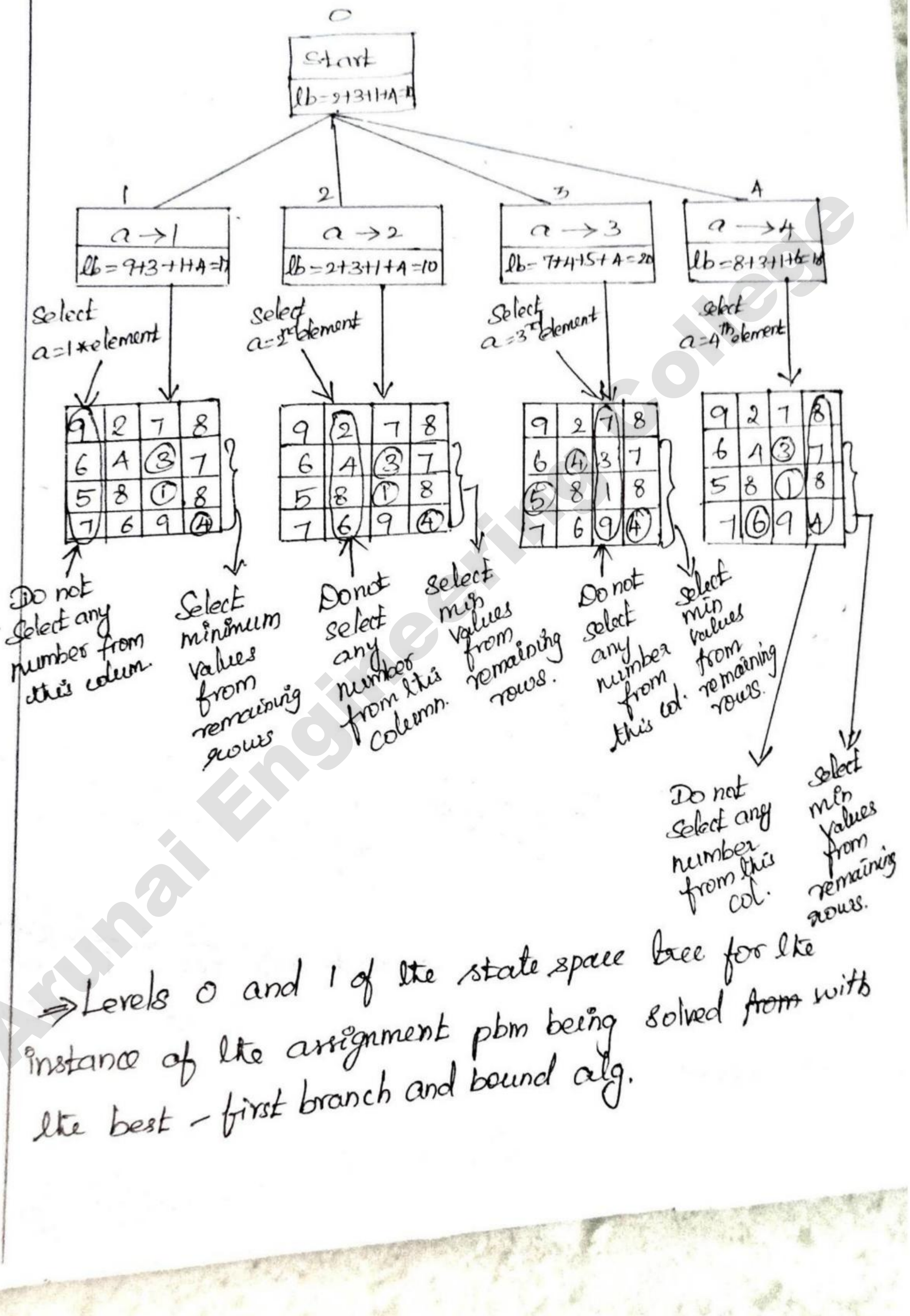
	Job1	Job2	Job3	Job4
Person A	9	(2)	7	8
Person B	6	4	(3)	7
Person C	5	8	(1)	8
Person D	7	6	9	(4)

Lower bound.

⇒ Any solution to this problem will have total cost at least: 2+3+1+4 (or (5+2+1+4) if we choose column.

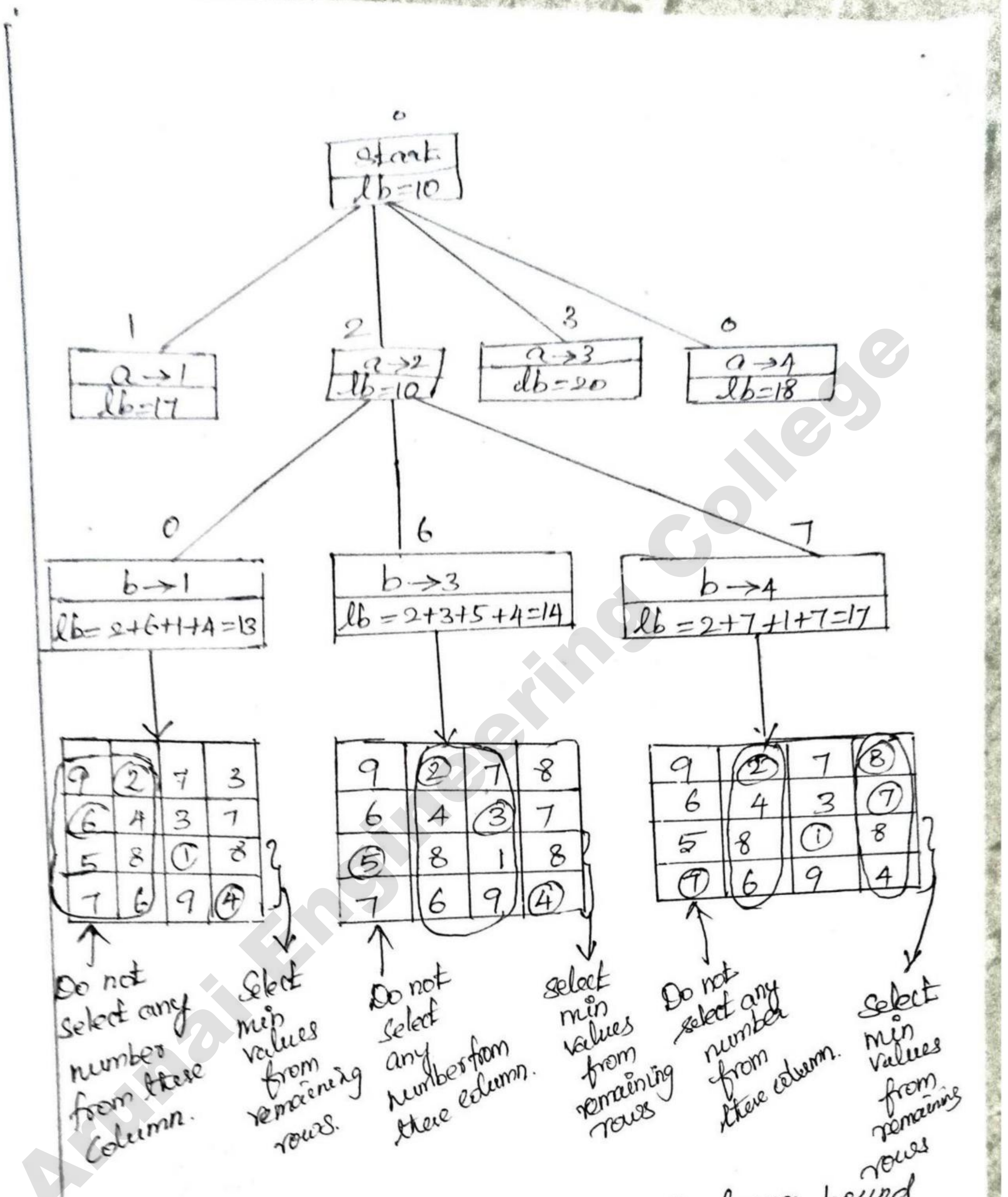
Scanned by TapScanner

State Space Tree:



⇒ Levels 0 and 1 of the state space tree for the instance of the assignment pbm being solved from with the best - first branch and bound alg.

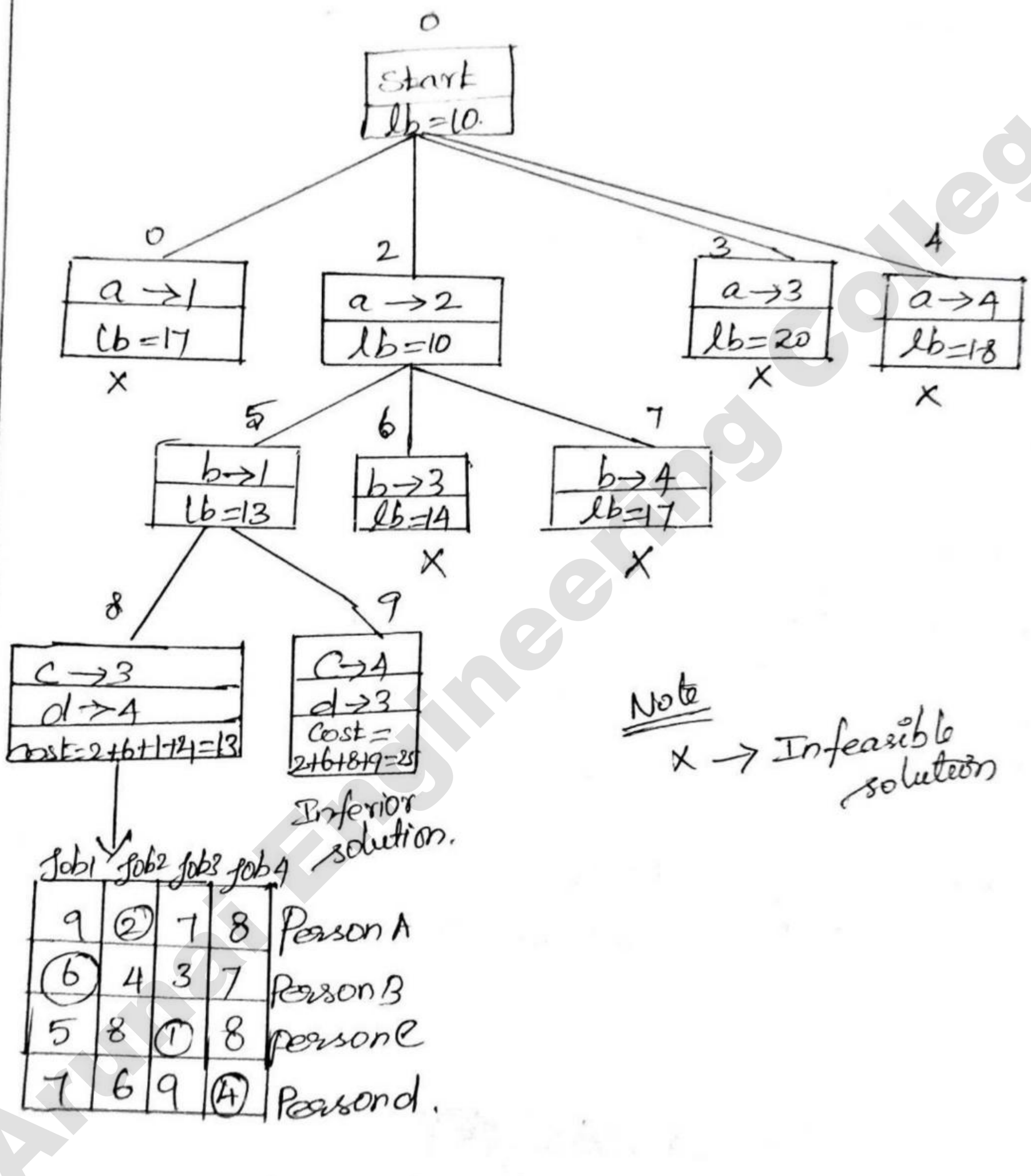
Scanned by TapScanner



⇒From level 1, the node with min lower bound value is selected (ie) node 2 with lb=10 is selected and expanded.

Scanned by TapScanner

⇒ Levels 0, 1 and 2 of the state-space tree for the instance of the assignment problem being solved with the best-first branch and bound alg.



Job1	Job2	Job3	Job4	
9	(2)	7	8	Person A
(6)	4	3	7	Person B
5	8	(1)	8	person C
7	6	9	(4)	Person d.

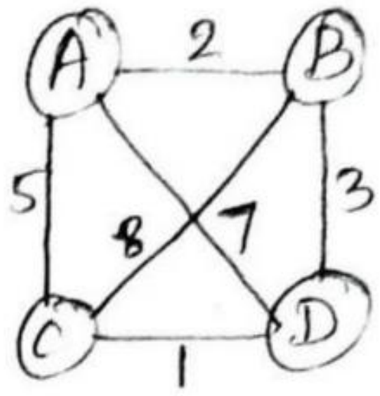
Note

X → Infeasible solution

⇒ The solution is a=2, b=6, c=1 and d=4 with a total cost of 13.

Scanned by TapScanner

(5) Apply the branch and bound algorithm to solve the travelling salesman problem for the following graph.



Problem Statement:

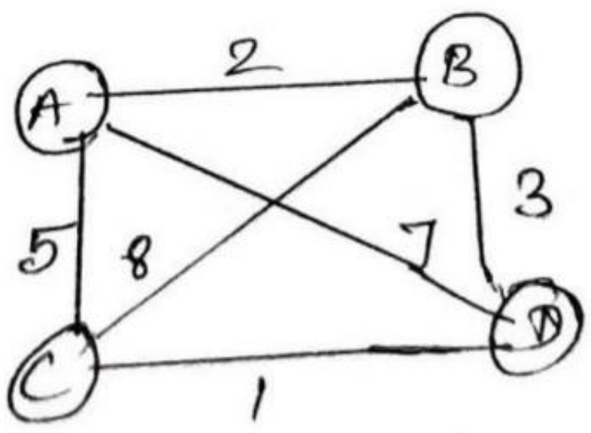
*If there are n cities and cost of travelling from any city to another city is given then we have to obtain the cheapest round-trip such that each city is exactly visited once and then returning to starting city, completes the tour.

⇒ Typically traveling salesman problem is represented by weighted graph.

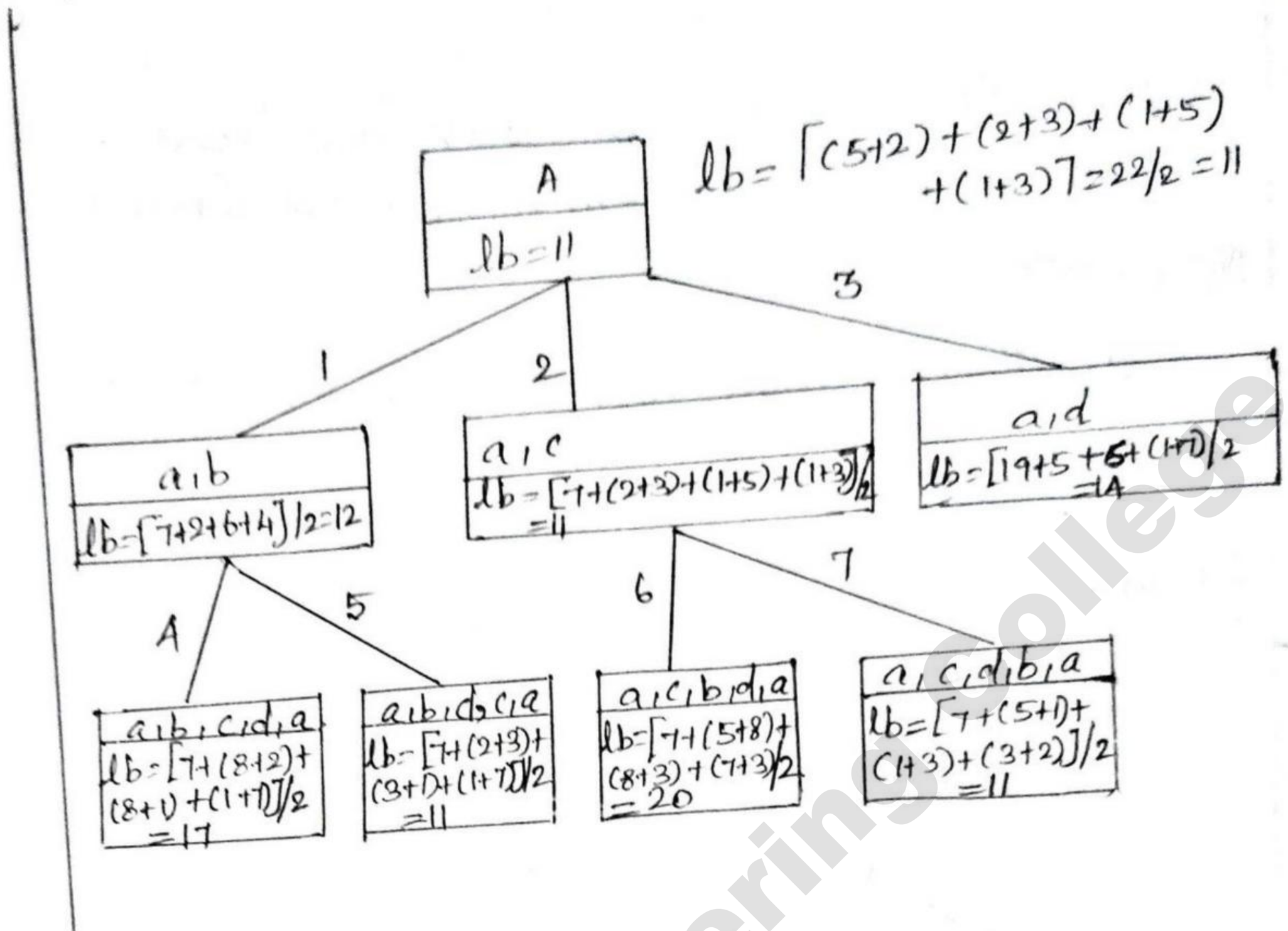
⇒ The lower bound is denoted by LB and can be obtained using the following formula.

$$LB = \frac{\sum_{v \in V} (\text{sum of costs of the two least cost edges adjacent to } v)}{2}.$$

Given Graph.



Scanned by TapScanner



Solutions

a b d c a

a c d b a

(6) Show that the Hamiltonian path problem reduces to the Hamiltonian circuit problem.

Problem Statement:

→ A Hamiltonian Path is a path in an undirected graph which visits each vertex exactly once.

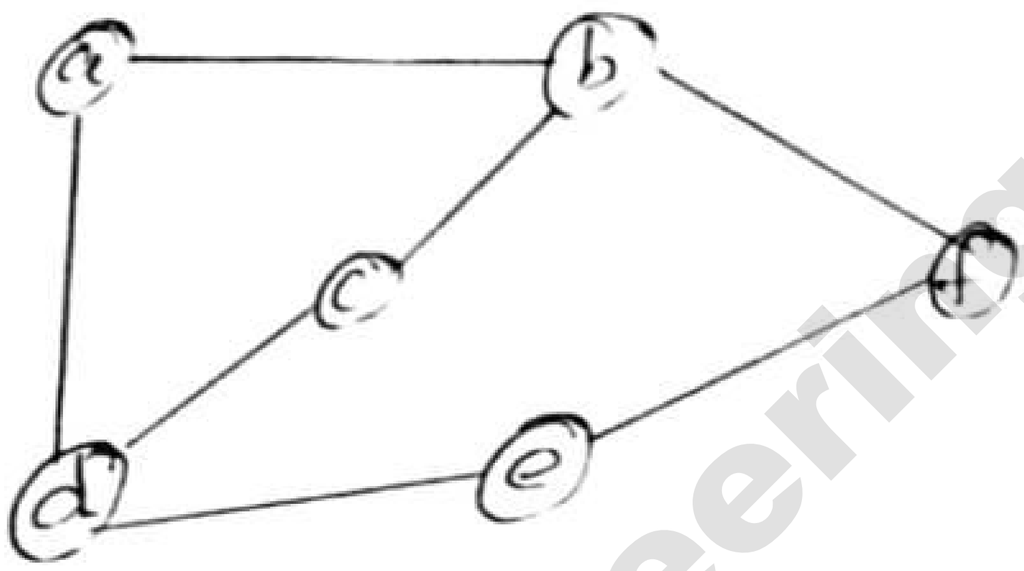
→ A Hamiltonian circuit is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex.

Scanned by TapScanner

* Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem which is NP complete.

⇒ Hamiltonian paths and cycles are named after william Rowan Hamilton.

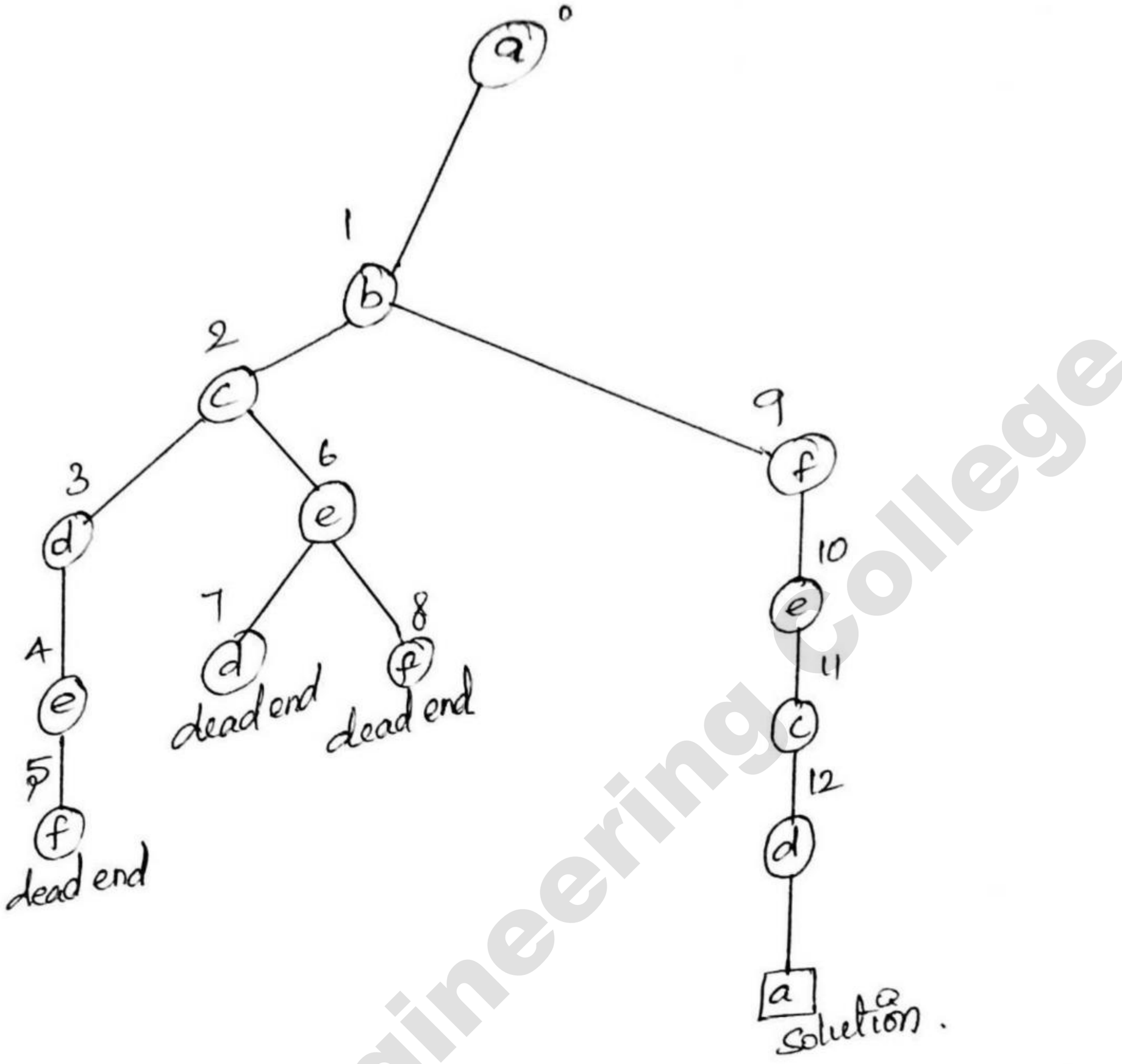
Example



⇒ The problem of finding Hamiltonian cycle is solved by backtracking approach.

⇒ We make vertex a the root space tree. from vertex a, we have three ways, so we resolve the tie using alphabet order, we select vertex b.

⇒ from b, the algorithm proceeds to c then to d, then to e and finally to f, which proves to be a dead end.



⇒ So the algorithm backtracks from f to e, then to d, and then to c, which provides alternative to pursue.

⇒ Going from c to e eventually proves useless, and the alg has to backtrack from e to c and then to d.

⇒ From there, it goes to the vertices f, e, c and d, from which it legitimately return to a, yielding the Hamiltonian circuit a, b, f, e, c, d, a.

Scanned by TapScanner

Scanned by TapScanner

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 77099

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2015.

Fourth Semester

Computer Science and Engineering

CS 6402 — DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Information Technology)

(Regulation 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Write an algorithm to find the number of binary digits in the binary representation of a positive decimal integer.

2. Write down the properties of asymptotic notations.

3. Design a brute-force algorithm for computing the value of a polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x \cdots a_0$ at a given point x_0 and determine its worst-case efficiency class.

4. Derive the complexity of Binary Search algorithm.

5. Write down the optimization technique used for Warshall's algorithm. State the rules and assumptions which are implied behind that.

6. List out the memory functions used under Dynamic Programming.

7. What do you mean by 'perfect matching' in bipartite graphs?

8. Define flow 'cut'.

9. How NP-Hard problems are different from NP-Complete?

10. Define Hamiltonian Circuit problem.

Scanned by TapScanner

PART B — (5 × 16 = 80 marks)

11. (a) If you have to solve the searching problem for a list of n numbers, how can you take advantage of the fact that the list is known to be sorted? Give separate answers for

(i) lists represented as arrays

(ii) lists represented as linked lists

Compare the time complexities involved in the analysis of both the algorithms. (16)

Or

(b) (i) Derive the worst case analysis of Merge Sort using suitable illustrations. (8)

(ii) Derive a loose bound on the following equation

$f(x) = 35x^5 + 22x^4 + 14x^3 - 2x^2 - 4x + x - 15$ (8)

12. (a) (i) Solve the following using Brute-Force algorithm: (10)

Find whether the given string follows the specified pattern and return 0 or 1 accordingly.

Examples:

(1) Pattern: "abba", input: "redblueredblue" should return 1

(2) Pattern: "aaaa", input: "asdasdasdasd" should return 1

(3) Pattern: "aabb", input: "xyzabcxzyabc" should return 0

(ii) Explain the convex hull problem and the solution involved behind it. (6)

Or

(b) A pair contains two numbers, and its second number is on the right side of the first one in an array. The difference of a pair is the minus result while subtracting the second number from the first one. Implement a function which gets the maximal difference of all pairs in an array (using Divide and Conquer method). (16)

13. (a) (i) Given the mobile numeric keypad. You can only press buttons that are up, left, right or down to the first number pressed to obtain the subsequent numbers. You are not allowed to press bottom row corner buttons (i.e., * and #). Given a number N, how many key strokes will be involved to press the given number. What is the length of it? Which dynamic programming technique could be used to find solution for this? Explain each step with the help of a pseudo code and derive its time complexity. (12)

(ii) How do you construct a minimum spanning tree using Kruskal's algorithm? Explain. (4)

Or

(b) (i) Let $A = \{a/119, m/90, c/247, g/283, h/72, f/77, b/92, j/19\}$ be the letters and its frequency of distribution in a text file. Compute a suitable Huffman coding to compress the data effectively. (8)

(ii) Write an algorithm to construct the optimal binary search tree given the roots $r(i, j)$, $0 \le i \le j \le n$. Also prove that this could be performed in time $O(n)$. (8)

14. (a) (i) Maximize $p = 2x + 3y + z$

subject to $x + y + z \le 40$

$2x + y - z \ge 10$

$-y + z \ge 10$

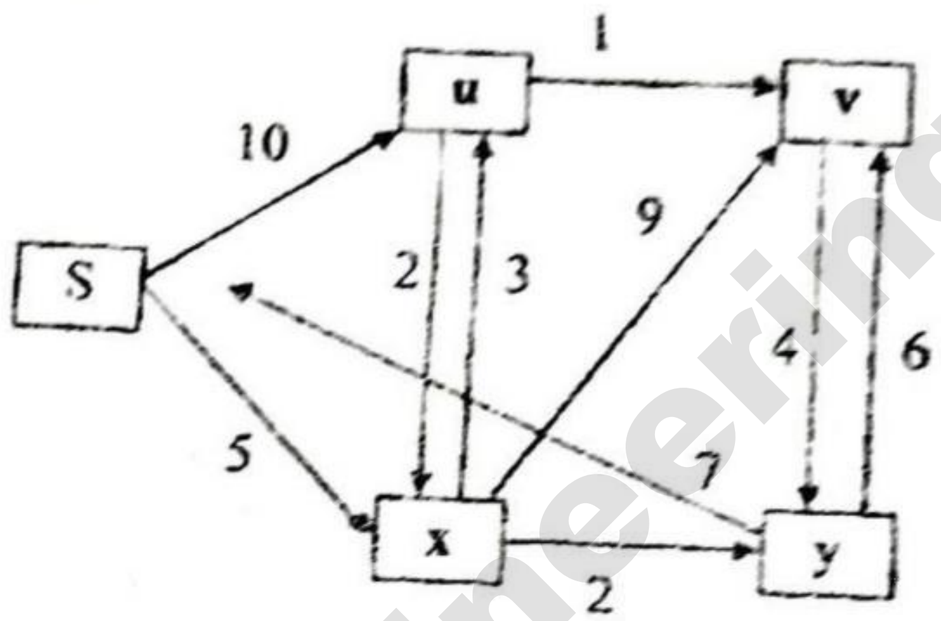
$x \ge 0,\ y \ge 0,\ z \ge 0.$ (8)

(ii) Write down the optimality condition and algorithmic implementation for finding M-augmenting paths in bipartite graphs. (8)

Or

(b) (i) Briefly describe on the Stable marriage problem. (6)

(ii) How do you compute maximum flow for the following graph using Ford-Fulkerson method? (10)



15. (a) (i) Suggest an approximation algorithm for traveling salesperson problem. Assume that the cost function satisfies the triangle inequality. (8)

(ii) Explain how job assignment problem could be solved, given n tasks and n agents where each agent has a cost to complete each task, using Branch and Bound technique. (8)

Or

(b) (i) The knight is placed on the first block of an empty board and, moving according to the rules of chess, must visit each square exactly once. Solve the above problem using backtracking procedure. (10)

(ii) Implement an algorithm for Knapsack problem using NP-Hard approach. (6)

Reg. No. :

Question Paper Code : 27168

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2015.

Third/Fourth Semester

Computer Science and Engineering

CS 6402 — DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Information Technology/Computer and Communication Engineering)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. The (*log n*)*th* smallest number of *n* unsorted numbers can be determined in $O(n)$ average-case time (True/False).

2 Write the recursive Fibonacci algorithm and its recurrence relation.

3. Give the mathematical notation to determine if a convex direction is towards left or right and write the algorithm.

4. Prove that any comparison sort algorithm requires Ω (n log n) comparisons in the worst case.

5. State how Binomial Coefficient is computed?

6. What is the best algorithm suited to identify the topography for a graph. Mention its efficiency factors.

7. Determine the Dual linear program for the following LP,

Maximize $3a + 2b - c$

Subject to,

$2a + b + c <= 3$

$a + b + c <= 4$

$3a + 3b + 6c <= 6$

$a, b, c >= 0.$

8. Define Network Flow and Cut.

9. Draw the decision tree for comparison of three values.

10. Depict the proof which says that a problem 'A' is no harder or no easier than problem 'B'.

Scanned by TapScanner

PART B — (5 × 16 = 80 marks)

11. (a) (i) Write the Insertion sort algorithm and estimate its running time. (8)

(ii) Find the closest asymptotic tight bound by solving the recurrence equation $T(n) = 8T(n/2) + n2$ with $(T(1) = 1)$ using Recursion tree method. [Assume that $T(1) \in \Theta(1)$]. (8)

Or

(b) (i) Suppose W satisfies the following recurrence equation and base case (where c is a constant) : $W(n) = c.n + W(n/2)$ and $W(1) = 1$. What is the asymptotic order of $W(n)$. (6)

(ii) Show how to implement a stack using two queues. Analyze the running time of the stack operations. (10)

12. (a) (i) Write the algorithm to perform Binary Search and compute its run time complexity. (8)

(ii) Compute the multiplication of given two matrices using Strassen's matrix multiplication method : (8)

$$A = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{bmatrix}.$$

Or

(b) (i) Write down the algorithm to construct a convex hull based on divide and conquer strategy. (8)

(ii) Find the optimal solution to the fractional knapsack problem with given data : (8)

Item	Weight	Benefit
A	2	60
B	3	75
C	4	90

13. (a) (i) The binary string below is the title of a song encoded using Huffman codes.

0011000101111101100111011101100000100111010010101

Given the letter frequencies listed in the table below, build the Huffman codes and use them to decode the title. In cases where there are multiple "greedy" choices, the codes are assembled by combining the first letters (or groups of letters) from left to right, in the order given in the table. Also, the codes are assigned by labeling the left and right branches of the prefix/code tree with '0' and '1', respectively. (10)

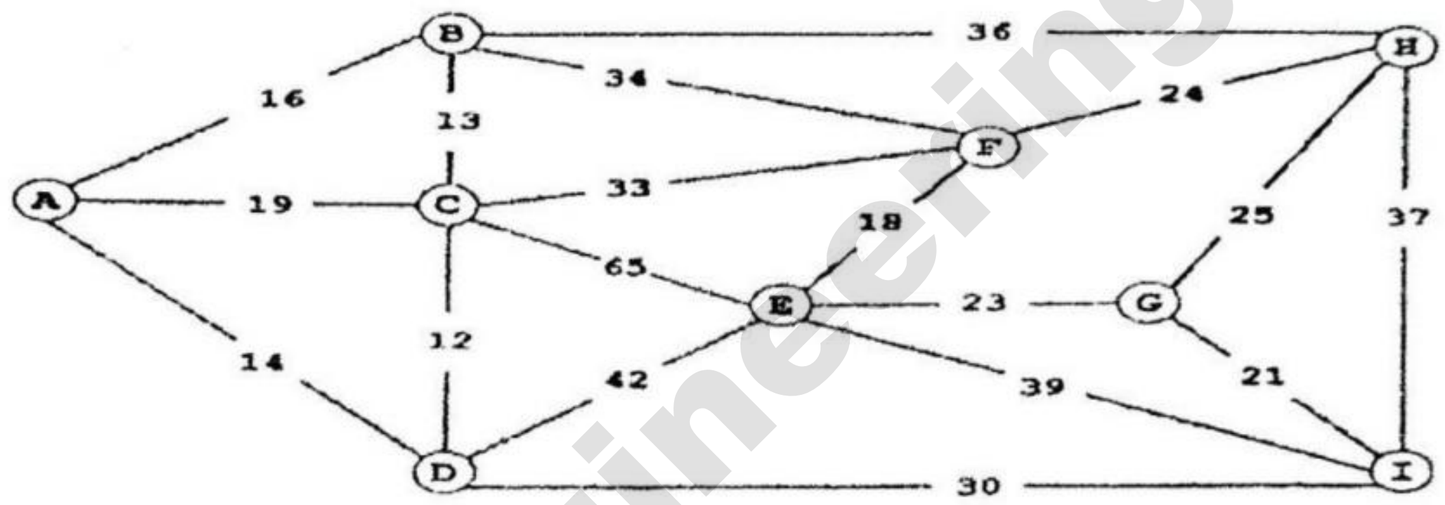
Letter	a	h	v	w	' '	e	t	l	o
Frequency	1	1	1	1	2	2	2	3	3

(ii) Write the procedure to compute Huffman code. (6)

Or

(b) (i) Write and analyze the Prim's Algorithm. (6)

(ii) Consider the following weighted graph.



Give the list of edges in the MST in the order that Prim's algorithm inserts them. Start Prim's algorithm from vertex A. (10)

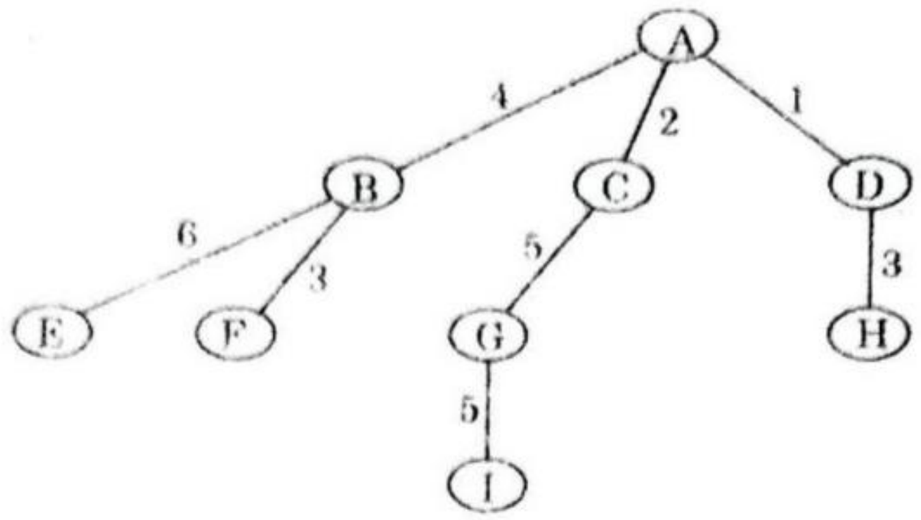
14. (a) (i) Use Simplex to solve the farmers problem given below :

A farmer has a 320 acre farm on which he plants two crops: rice and wheat. For each acre of rice planted, his expenses are 50 and for each acre of wheat planted, his expenses are 100. Each acre of rice requires 100 quintals of storage and yields a profit of 60; each acre of wheat requires 40 quintals of storage and yields a profit of 90. If the total amount of storage space available is 19,200 quintals and the farmer has only '20,000 on hand, how many acres of each crop should he plant in order to maximize his profit? What will his profit be if he follows this strategy? (12)

(ii) Write the procedure to Initialize Simplex which determines if a linear program is feasible or not? (4)

Or

(b) (i) Illustrate the workings of the maximum matching algorithm on the following weighted tree. (12)



(ii) Explain Max-Flow Problem. (4)

15. (a) (i) Using an example prove that, satisfiability of boolean formula in 3-Conjunctive Normal Form is NP – complete. (12)

(ii) State the relationships among the complexity class algorithms with the help of neat diagrams. (4)

Or

(b) (i) Show that the Hamiltonian Path problem reduces to the Hamiltonian Circuit Problem and vice versa. (10)

(ii) What is an approximation algorithm? Give example. (6)

Scanned by TapScanner

Reg. No.

Question Paper Code : 57249

B.E./B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016

Third/Fourth Semester

Computer Science and Engineering

CS 6402 – DESIGN AND ANALYSIS OF ALGORITHMS

(Regulations 2013)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions.

PART – A (10 × 2 = 20 Marks)

1. Give the Euclid's algorithm for computing gcd (m, n).
2. Compare the orders of growth of $n(n-1)/2$ and n^2.
3. Give the general strategy of Divide and Conquer Method.
4. What is the closest -pair problem ?
5. Define the Single Source Shortest Paths Problem.
6. State the assignment Problem.
7. What is a state space graph ?
8. State Extreme Point Theorem.
9. Give the purpose of lower bound.
10. What is Euclidean minimum spanning tree problem ?

PART – B (5 × 16 = 80 Marks)

11. (a) (i) Give the definition and Graphical Representation of O-Notation. (8)

(ii) Give the algorithm to check whether all the elements in a given array of n elements are distinct. Find Worst case complexity of the same. (8)

OR

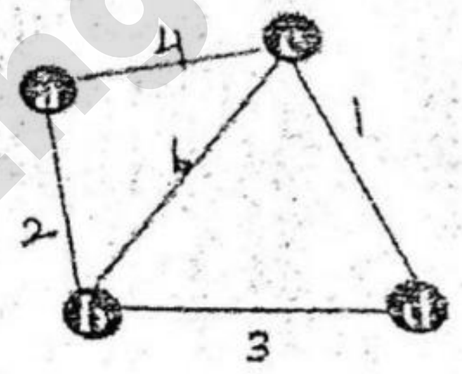
(b) Give the recursive Algorithm for finding the number of binary digits in n's binary representation, where n is a positive decimal integer. Find the recurrence relation and complexity. (16)

12. (a) State and Explain the Merge Sort algorithm and Give the recurrence relation and efficiency. (16)

OR

(b) Explain the method used for performing Multiplication of two large integers. Explain how Divide Conquer Method can be used to solve the same. (16)

13. (a) Discuss about the algorithm and Pseudocode to find the Minimum Spanning Tree using Prim's Algorithm . Find the Minimum Spanning tree for the graph shown below.

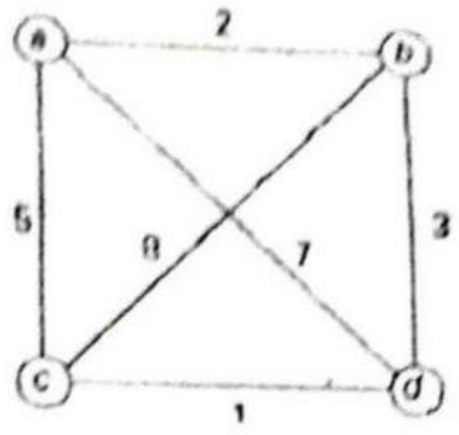


And Discuss about the efficiency of the Algorithm. (16)

OR

Scanned by TapScanner

(b) Find all the Solution to the travelling salesman problem (cities and distances shown below) by exhaustive search. Give the optimal solution. **(16)**

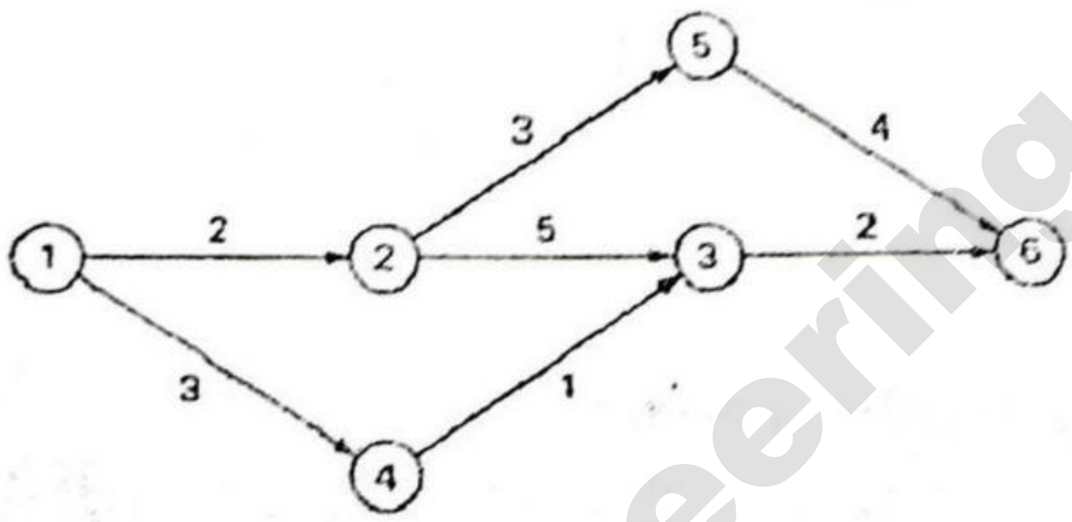


(a) (i) Summarize the simplex method. **(8)**

(ii) State and prove Max-Flow Min-Cut Theorem **(8)**

OR

(b) Apply the shortest-augmenting-path algorithm to the network shown below. **(16)**



(a) Give any five undecidable problems and explain the famous halting Problem. **(16)**

OR

(b) State the subset-sum problem and Complete state-space tree of the backtracking algorithm applied to the instance A = {3, 5, 6, 7} and d = 15 of the subset-sum problem. **(16)**

Reg. No. :

Question Paper Code : 80293

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2016.

Fourth Semester

Computer Science and Engineering

CS 6402 — DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Information Technology)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Design an algorithm to compute the area and Circumference of a circle.

2. Define recurrence relation.

3. Write an algorithm for brute force closest —pair problem.

4. What is worst case complexity of binary search?

5. What is meant by principle of optimality?

6. How to calculate the efficiency of Dijkstra's Algorithm?

7. Define the iterative improvement technique

8. What is maximum cardinality matching?

9. Write the formula for decision tree for searching a sorted array.

10. State the reason for terminating search path at the current node in branch and bound algorithm.

Scanned by TapScanner

PART B — (5 × 16 = 80 marks)

11 (a) (i) Use the most appropriate notation to indicate the time efficiency class of sequential search algorithm in the worst case, best case and the average case. (8)

(ii) State the general plan for analyzing the time efficiency of nonrecursive algorithms and explain with an example (8)

Or

(b) Solve the following recurrence relations

- $x(n) = x(n-1) + 5$ for $n > 1 x(1) = 0$
- $x(n) = 3x(n-1)$ for $n > 1 x(1) = 4$
- $x(n) = x(n-1) + n$ for $n > 0\ x(0) = 0$
- $x(n) = x(n/2) + n$ for $n > 1\ x(1) = 1$ (solve for $n = 2^k$)
- $x(n) = x(n/3) + 1$ for $n > 1\ x(1) = 1$ (solve for $n = 3^k$) (16)

12. (a) There are 4 people who need to be assigned to execute 4 jobs (one person per job) and the problem is to find an assignment with the minimum total cost. The assignment costs is given below, solve the assignment problem by exhaustive search. (16)

	Job 1	Job 2	Job 3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Perron 3	5	8	1	8
Person 4	7	6	9	4

Or

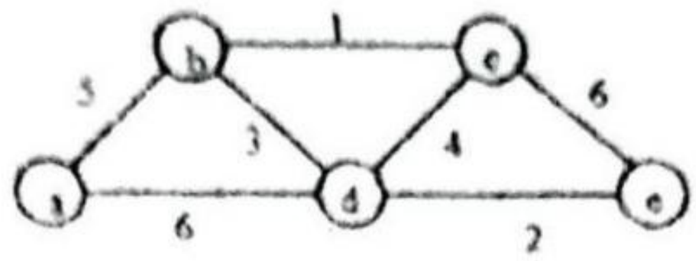
(b) Give the algorithm for Quicksort. With an example show that Quicksort is not a stable sorting algorithm. (16)

13. (a) Solve the all-pairs shortest-path problem for the digraph with the following weight matrix: (16)

$$\begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

Or

(b) Apply Kruskal's algorithm to find a minimum spanning tree of the following graph. (16)



14. (a) (i) State and prove Max-Flow Min-Cut Theorem

(ii) Summarize the steps of the simplex method. (16)

Or

(b) (i) Explain briefly about Stable marriage algorithm. (10)

(ii) Determine the time-efficiency class of the stable marriage algorithm. (6)

15. (a) (i) Draw a decision tree and find the number of key comparisons in the worst and avarage cases for the three-element bubble sort (8)

(ii) Write backtracking algorithm for 4-Queen's problem and discuss the possible solution. (8)

Or

(b) Solve the following instance of Knapsack problem by branch and bound algorithm. (16)

Item	weight	profit	
1	5	$40	
2	7	$35	
3	2	$18	W=15
4	4	$4	
5	5	$10	
6	1	$2	

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017.

Third/Fourth Semester

Computer Science and Engineering

CS 6402 — DESIGN AND ANALYSIS OF ALGORITHMS

(Common to information Technology)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is an Algorithm?

2. Write an algorithm to compute the greatest common divisor of two numbers.

3. Devise an algorithm to make for 1655 using the Greedy strategy. The coins available are {1000, 500, 100, 50, 20, 10, 5}.

4. What is closest-pair problem?

5. State the general principle of greedy algorithm.

6. What do you mean by dynamic programming?

7. What do you mean by 'perfect matching' in bipartite graphs?

8. State: Planar coloring graph problem.

9. What is an articulation point in a graph?

10. Define P and NP problems.

Scanned by TapScanner

PART B — (5 × 13 = 65 marks)

11 (a) Briefly explain the mathematical analysis of recursive and non-recursive algorithm (13)

Or

(b) Explain briefly **Big oh Notation, Omega Notation and Theta Notations.** Give examples (13)

12 (a) What is divide and conquer strategy and explain the binary search with suitable example problem. (13)

Or

(b) Solve the following using **Brute-Force algorithm:** (13)

Find whether the given string follows the specified pattern and return 0 or 1 accordingly.

Examples

(i) Pattern: "abba", input: "redblueredblue" should return 1

(ii) Pattern: "aaaa", input: "asdasdasdasd" should return 1

(iii) Pattern: "aabb" input: "xyzabcxzyabc" should return 0

13 (a) Solve the following instance of the 0 / 1, knapsack problem given the knapsack capacity in W = 5 using dynamic programming and explain it. (13)

Items	Weight	Value
1	4	10
2	3	20
3	2	15
4	5	25

Or

(b) Write the Huffman's Algorithm. Construct the Huffman's tree for the following data and obtain its Huffman's Code. (13)

Character	A	B	C	D	E	–
Probability	0.5	0.35	0.5	0.1	0.4	0.2

14 (a) Describe in detail the simplex algorithm methods. (13)

Or

(b) Explain KMP string matching algorithm for finding a pattern on a text and analyze the algorithm. (13)

71679

Scanned by TapScanner

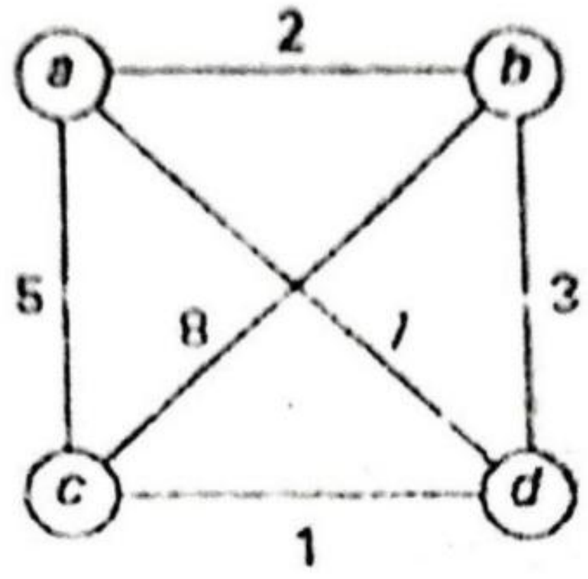
15. (a) Discuss the approximation algorithm for NP- hard problems. (13)

Or

(b) Describe the backtracking solution to solve 8-queens problem. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Apply Branch and Bound algorithm to solve the Travelling Salesman Problem for (15)



Or

(b) Write an algorithm for quick sort and write its time complexity with example list are 5, 3, 1, 9, 8, 2, 4, 7. (15)

Scanned by TapScanner

Reg. No. : 5 1 0 4 1 4 1 0 4 0 3 8

Question Paper Code : 50388

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2017

Third/Fourth Semester

Computer Science and Engineering

CS 6402 – DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Information Technology)

(Regulations 2013)

Time : Three Hours | Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. How to measure an algorithm's running time ?
2. What do you mean by "Worst case-efficiency" of an algorithm ?
3. Give the general plan of divide and conquer algorithms.
4. Write the advantages of insertion sort.
5. What does Floyd's algorithm do ?
6. Define principle of Optimality.
7. What are Bipartite Graphs ?
8. State extreme point theorem.
9. Explain promising and nonpromising node.
10. Differentiate feasible solution and optimal solution.

PART – B **(5×13=65 Marks)**

11. a) Discuss the steps in Mathematical analysis for recursive algorithms. Do the same for finding the factorial of a number.

(OR)

b) What are the Rules of Manipulate Big-Oh Expressions and about the typical growth rates of algorithms ?

12. a) Explain the Bruteforce method to find the two closest points in a set of n points in k-dimensional space.

(OR)

b) Explain the working of Merge Sort Algorithm with an example.

Scanned by TapScanner

13. a) Explain the working of Prim's Algorithm.

(OR)

b) Explain the Dijkstra's shortest path algorithm and its efficiency.

14. a) List the steps in Simplex Method and give the efficiency of the same.

(OR)

b) What is stable marriage problem ? Give the algorithm and analyze it.

15. a) Find the Optimal solution using Branch and Bound for the following assignment problem.

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

(OR)

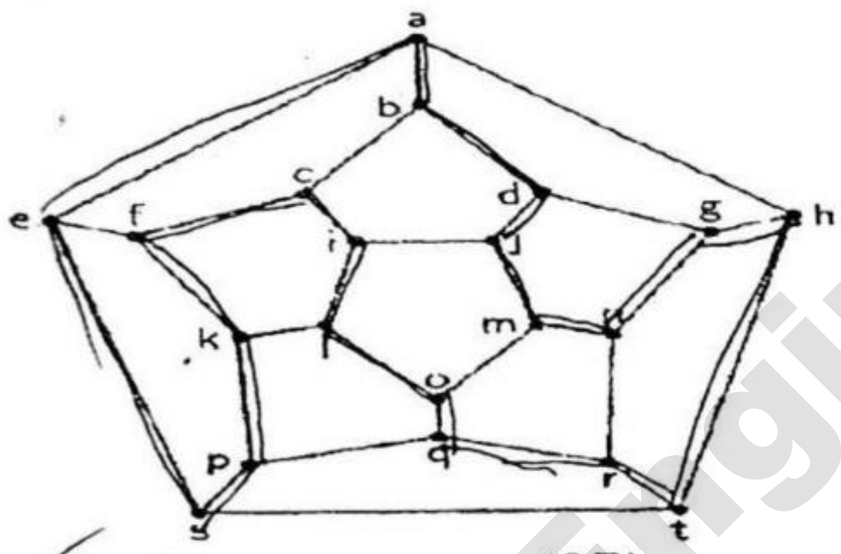
b) Give the methods for Establishing Lower Bounds.

PART – C

(1×15=15 Marks)

(Application/Design/Analysis/Evaluation/Creativity questions) (Case Study/ Comprehensive questions)

16. a) Find a Hamiltonian circuit or disprove its existence in the graph given below.



(OR)

b) Explain the steps in Building a Huffman Tree. Find the codes for the alphabets given below according to the frequency.

– (Space) 4
A 2
E 5
H 1
I 2
L 2
M 2
P 2
R 1
S 2
X 1

Scanned by TapScanner

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 40906

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2018

Third/Fourth Semester

Computer Science and Engineering

CS 6402 – DESIGN AND ANALYSIS OF ALGORITHMS

(Common to : Information Technology)

(Regulations 2013)

Time : Three Hours

Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. Give the Euclid's algorithm for computing gcd of two numbers.
2. What is a basic operation ?
3. What is an exhaustive search ?
4. State Master's theorem.
5. Define transitive closure of a directed graph.
6. Define the minimum spanning tree problem.
7. How is a transportation network represented ?
8. What is meant by maximum cardinality matching ?
9. How is lower bound found by problem reduction ?
10. What are tractable and non-tractable problems ?

PART – B **(5×13=65 Marks)**

11. a) Define Big O notation, Big Omega and Big Theta Notation. Depict the same graphically and explain.

(OR)

b) Give the General Plan for Analyzing the Time Efficiency of Recursive Algorithms and use recurrence to find number of moves for Towers of Hanoi problem.

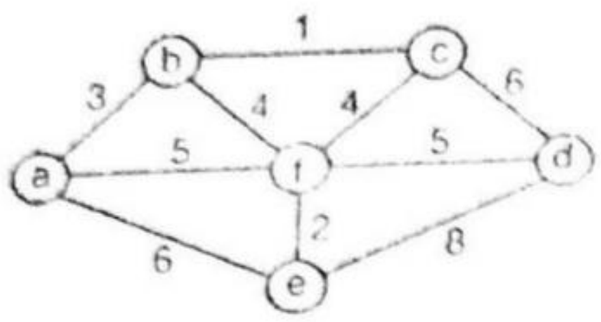
Scanned by TapScanner

12. a) Explain Merge sort algorithm with an example.

(OR)

b) Explain the working of Strassen's Matrix Multiplication with the help of divide and conquer method.

13. a) Give the Pseudo code for Prim's algorithm and apply the same to find the minimum spanning tree of the graph shown below :



(OR)

b) Explain the memory function method for the knapsack problem and give the algorithm.

14. a) Give the summary of the simplex method.

(OR)

b) Prove that the stable marriage algorithm terminates after no more than n^2 iterations with a stable marriage output.

15. a) What is Class NP ? Discuss about any five problems for which no polynomial - time algorithm has been found.

(OR)

b) Elaborate on the nearest-neighbor algorithm and multifragment-heuristic algorithm for TSP problem.

PART – C (1×15=15 Marks)

16. a) Consider the problem of finding the smallest and largest elements in an array of n numbers.

i) Design a presorting-based algorithm for solving this problem and determine its efficiency class.

Scanned by TapScanner

ii) Compare the efficiency of the three algorithms :

(A) the brute-force algorithm. (B) this presorting-based algorithm, and

(C) the divide-and conquer algorithm. **(8)**

(OR)

i) Apply Warshall's algorithm to find the transitive closure of the digraph defined by the following adjacency matrix

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

ii) Prove that the time efficiency of Warshall's algorithm is cubic. **(7)**

ii) Explain why the time efficiency of Warshall's algorithm is inferior to that of the traversal-based algorithm for sparse graphs represented by their adjacency lists. **(8)**

Reg. No. A U H I P P O . C O M

Question Paper Code : 20364

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2018.

Third/Fourth Semester

Computer Science and Engineering

CS 6402 — DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Information Technology)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define algorithm. List the desirable properties of an algorithm.

2. Define best, worst, average case time complexity.

3. What are the differences between dynamic programming and divide and conquer approaches?

4. Give an example for Hamiltonian circuit.

5. Define multistage graphs. Give an example.

6. How dynamic programming is used to solve Knapsack problem?

7. Describe iterative improvement technique.

8. What is solution space? Give an example.

9. Define P and NP problems.

10. Give an example for sum-of-subset problem.

Scanned by TapScanner

PART B — (5 × 13 = 65 marks)

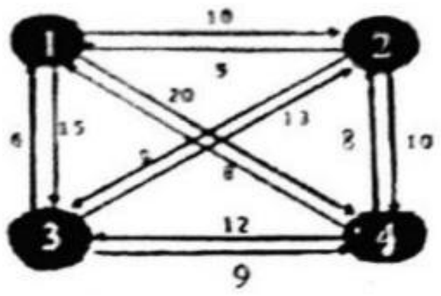
11. (a) (i) Prove that if $g(n)$ is $O(f(n))$ then $f(n)$ is $O(g(n))$. (5)

(ii) Discuss various methods used for mathematical analysis of recursive algorithms. (8)

Or

(b) Write the asymptotic notations used for best case, average case and worst case analysis of algorithms. Write an algorithm for finding maximum element in an array. Give best, worst and average case complexities. (13)

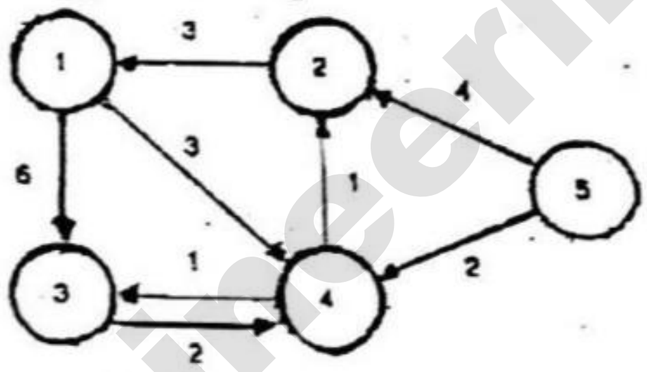
12. (a) Solve travelling salesman problem using brute force approach for the given example. How the solution can be obtained using branch and bound method? (10 + 3)



Or

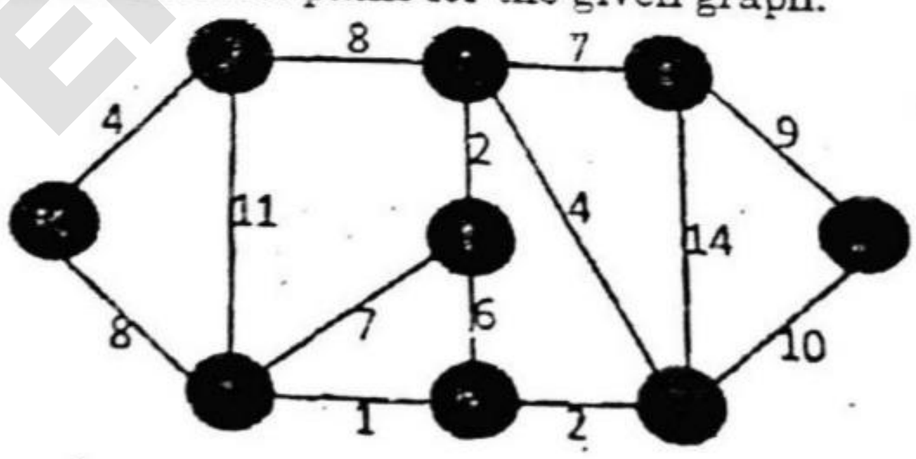
(b) Write the algorithm for quick sort. Provide a complete analysis of quick sort for the given set of numbers 12, 33, 23, 43, 44, 55, 64, 77 and 76. (13)

13. (a) Explain Floyds – Warshall algorithm using dynamic programming. Trace the algorithm for the given example. (13)



Or

(b) Explain how greedy approach is used in Dijkstra's algorithm for finding the single–source shortest paths for the given graph. (13)



14. (a) Illustrate the steps of the simplex methods with an example. (13)

Or

(b) Write the stable marriage algorithm and trace it with an instance. Analyze its running time complexity. (13)

15. (a) Consider the travelling salesperson instance defined by the following cost matrix. (13)

∞	20	30	10	11
15	∞	16	4	2
3	5	∞	2	4
19	6	18	∞	3
16	4	7	16	∞

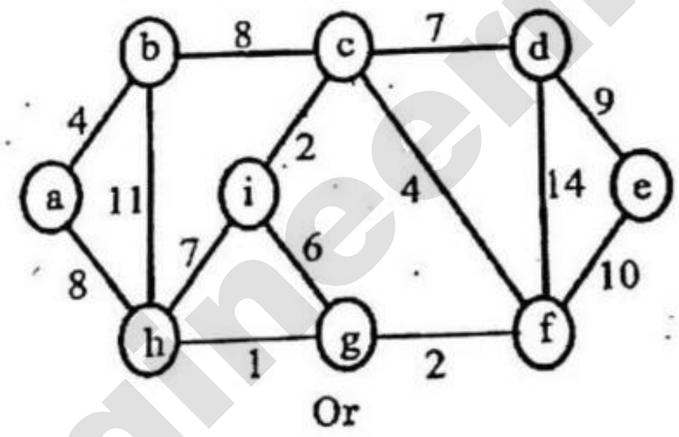
Draw the state space tree and show the reduced matrices corresponding to each of the node.

Or

(b) Discuss the approximation algorithm for NP-hard problems. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Apply the greedy technique to find the minimum spaning tree using Prim's algorithm for the given graph. (15)



Or

(b) Explain the 4-Queen's problem using backtracking. Write the algorithms. Give the estimated cost for all possible solutions of 4-Queen's problem. Specify the implicit and explicit constraints. (15)

Reg. No. :

Question Paper Code : 80097

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2019.

Fourth Semester

Computer Science and Engineering

CS 8451 — DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Information Technology/Computer and Communication Engineering)

(Regulation 2017)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. How do you measure the efficiency of an algorithm?

2. Prove that the if $f(n) = O(g(n))$ and $g(n) = O(f(n))$, then, $f(n) = \theta\, g(n)$.

3. Write the brute force algorithm to string matching.

4. What is the time and space complexity of Merge sort?

5. State the principle of optimality.

6. What is the constraint for binary search tree insertion?

7. State the principle of duality.

8. Define the capacity constraint in the context of maximum flow problem.

9. Define NP completeness and NP hard.

10. State Hamiltonian Circuit problem.

Scanned by TapScanner

PART B — (5 × 13 = 65 marks)

11. (a) (i) Solve the following recurrence equation :

(1) $T(n) = T(n/2) + 1$, where $n = 2^k$ for all $k \geq 0$ (4)

(2) $T(n) = T(n/3) + T(2n/3) + cn$, where 'c' is a constant and 'n' is the input size. (4)

(ii) Explain the steps involved in problem solving. (5)

Or

(b) (i) Write an algorithm for determining the uniqueness of an array. Determine the time complexity of your algorithm (10)

(ii) Explain time-space trade off of the algorithm designed. (3)

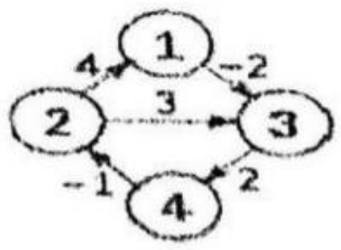
12. (a) What is the convex hull problem? Explain the brute force approach to solve convex-hull with an example. Derive the time complexity. (2 + 7 + 4)

Or

(b) Write the quicksort algorithm and explain it with an example. Derive the worst case and average case time complexity. (5 + 4 + 4)

13. (a) (i) Write the Floyd algorithm to find all pairs shortest path and derive its time complexity. (4 + 3)

(ii) Solve the following using Floyd's algorithm. (6)



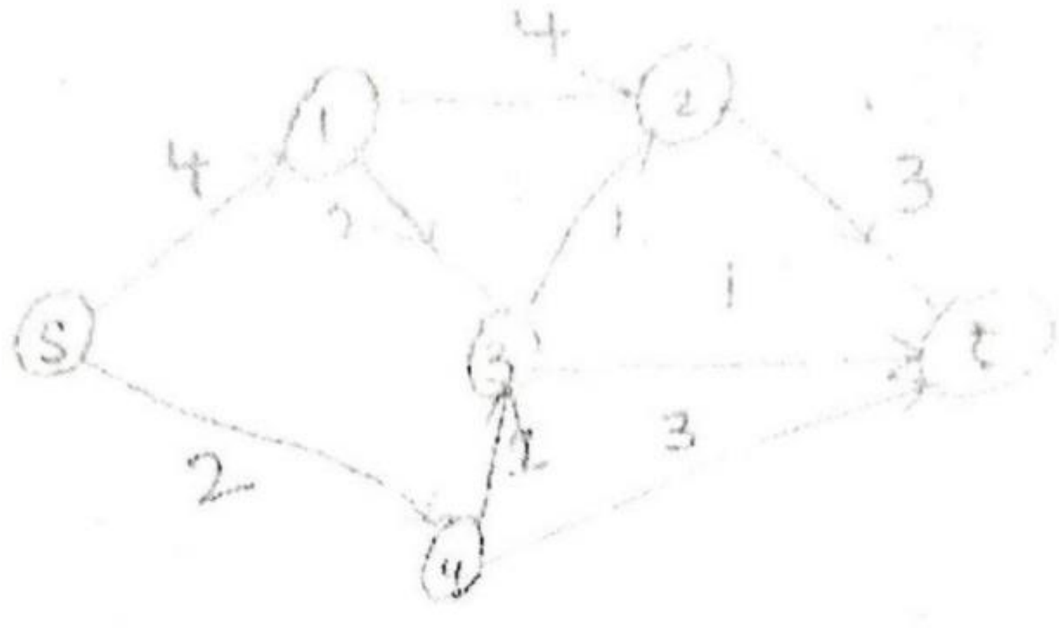
Or

(b) (i) Write the Huffman code algorithm and derive its time complexity. (5 + 2)

(ii) Generate the Huffman code for the following data comprising of alphabet and their frequency. (6)

$a : 1, b : 1, c : 2, d : 3, e : 5, f : 8, g : 13, h : 21$

14. (a) Determine the max flow in the following network. (13)



Or

(b) Solve the following set of equations using Simplex algorithm : (13)

Maximize : $18x_1 + 12.5x_2$

Subject to : $x_1 + x_2 \leq 20$

$x_1 \leq 12$

$x_2 \leq 16$

$x_1, x_2 \geq 0.$

15. (a) Write an algorithm to solve the Travelling salesman problem and prove that it is a 2 time approximation algorithm. (13)

Or

(b) Write an algorithm for subset sum and explain with an example. (13)

PART C — (1 × 15 = 15 marks)

16. (a) (i) Given a matrix of order $M \times N$, and two coordinates (p, q) and (r, s), which represents the top-left and bottom-right of a sub-matrix of the matrix, $M \times N$, calculate the sum of elements present in the sub-matrix in $O(1)$ time using dynamic programming. Determine the optimal sub-structure and write an algorithm. (10)

(ii) Prove that any algorithm that sorts by comparison, requires $\Omega(n \lg n)$ time. (5)

Or

(b) (i) The longest common subsequence (LCS) is the problem of finding the longest subsequence that is present in the given two sequences in the same order but not necessarily contiguously. Write an algorithm using dynamic programming that determines the LCS of two strings, 'x' and 'y' and returns the string 'z'. (10)

(ii) Prove that any algorithm that searches need to necessarily do $\Omega(\lg n)$ comparisons. (5)

Scanned by TapScanner

Reg. No. : 5 1 0 4 1 7 1 0 4 0 0 8

Question Paper Code : 90154

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2019

Fourth Semester

Computer Science and Engineering

CS8451 – DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Computer and Communication Engineering / Information Technology)

(Regulations 2017)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. State the transpose symmetry property of O and Ω.
2. Define recursion.
3. State the convex hull problem.
4. Outline the knapsack problem.
5. What is Brute Force method ?
6. Define a binary search tree.
7. When a linear program is said to be unbounded ?
8. What is a residual network in the context of flow networks ?
9. When is a problem said to be NP hard ?
10. State the Hamiltonian circuit problem.

PART – B **(5×13=65 Marks)**

11. a) i) Solve the following recurrence equations using iterative method or tree. **(7)**

ii) Elaborate asymptotic analysis of an algorithm with an example. **(6)**

(OR)

b) Write an algorithm using recursion that determines the GCD of two numbers. Determine the time and space complexity.

Scanned by TapScanner

12. a) State the travelling salesman problem. Elaborate the steps in solving the travelling salesman problem using brute force approach. **(13)**

(OR)

b) Write the algorithm to find the closest pair of points using divide and conquer and explain it with an example. Derive the worst case and average case time complexity. **(5+4+4)**

13. a) i) Outline the Dynamic programming approach to solve the Optimal Binary search tree problem and analyse its time complexity. **(4+2)**

ii) Construct the Optimal binary search tree for the following 5 keys with probabilities as indicated. **(7)**

i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

(OR)

b) Write a Greedy algorithm to solve the 0/1 knapsack problem. Analyse its time complexity. Show that this algorithm is not optimal with an example. **(5+2+6)**

14. a) What is iterative improvement ? Elaborate the steps in the simplex method with an example. **(13)**

(OR)

b) i) What is a bipartite graph ? Is the subset of a bipartite graph bipartite ? Outline with an example. **(2+1+4)**

ii) Outline the stable Marriage problem with an example. **(6)**

15. a) Elaborate how backtracking technique can be used to solve the n-queens problem. Explain with an example. **(13)**

(OR)

b) Outline the steps to find an approximate solution to NP-hard optimization problems using approximation algorithms with an example. **(13)**

PART – C **(1×15=15 Marks)**

16. a) Sort the following numbers using quick sort.

999, 888, 777, 666, 555, 444, 333, 222, 111, 11, 22, 33, 44, 55, 66, 77, 88, 99.

Illustrate each step in the sorting process. **(15)**

(OR)

b) i) The Longest Increasing Subsequence (LIS) problem is to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order. Write an algorithm using dynamic programming that determines the LIS of a string 'x'. For example, the length of LIS for {10, 22, 9, 33, 21, 50, 41, 60, 80} is 6 and LIS is {10, 22, 33, 50, 60, 80}. **(10)**

ii) Determine the Time and Space complexity of the above algorithm. **(5)**

Scanned by TapScanner