Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

End Semester Examination

Name of the Course: Design And Analysis of Algorithm **Synoptic**

SE | CMPN | IT | Sem-IV

Question No.	Question
Q. 1 a)	

		$g(n)$		
		n	$n \lg n$	n^2
$f(n)$	n^2	Ω	Ω	Θ
	$n^{1.5}$	Ω	Ω	O
	$\sqrt{2^n}$	Ω	Ω	Ω
	$n\sqrt{\lg n}$	Ω	O	O
	$n \log_{30} n$	Ω	Θ	O
	n^3	Ω	Ω	Ω

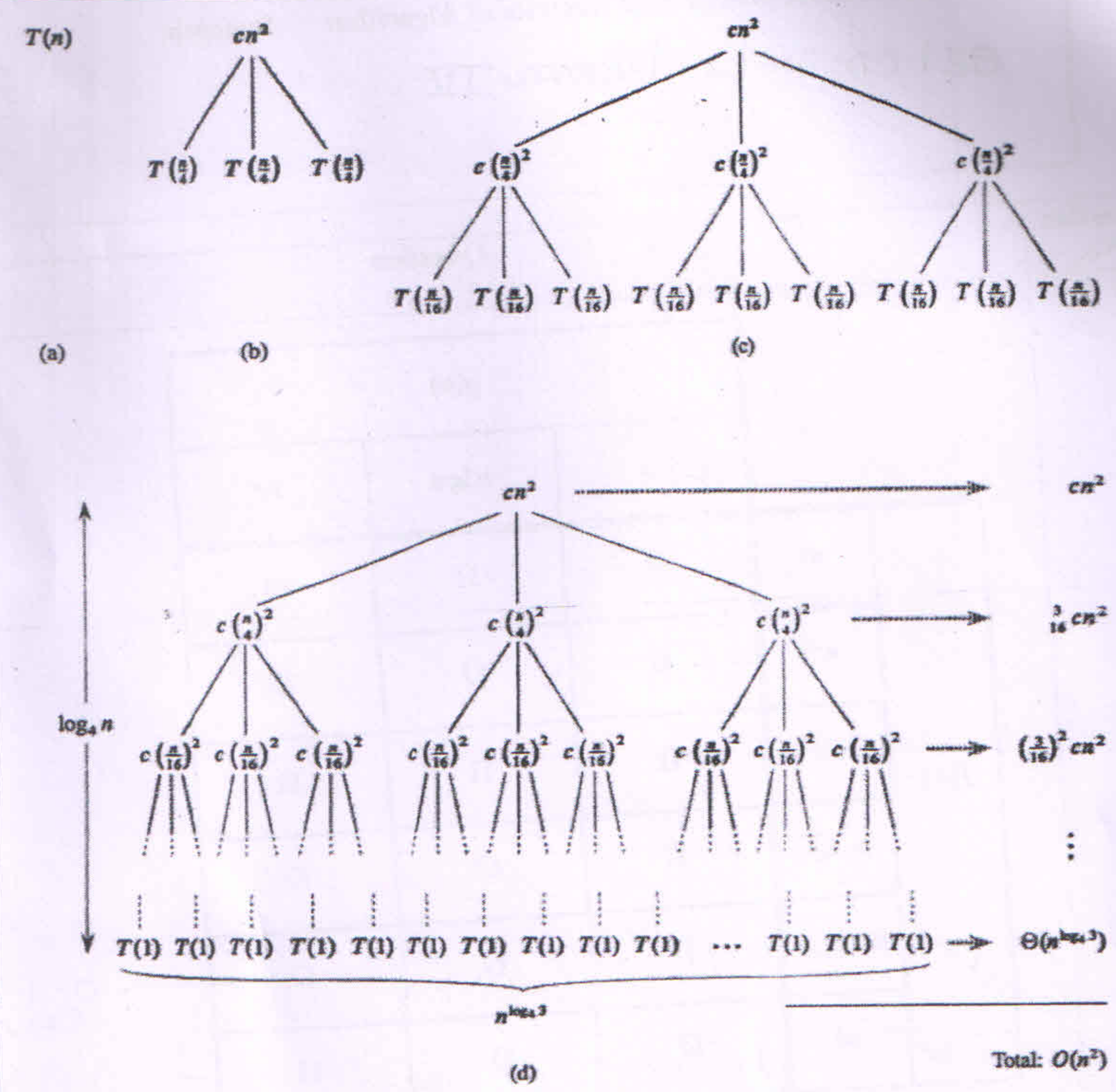
— 0.5 mark each

— 0.5 mark each.

Q. 1 b)

i. Ans:O(nlogn) * Only answer 0.5 marks
method 1.5 marks

ii. Ans: O(n^2) — * ~~co~~ only correct answer mentioned 0.5 marks



T(n)
cn²
T(n/4) T(n/4) T(n/4)
c(n/4)²
T(n/16)
(a)
(b)
(c)
log₄ n
3/16 cn²
c(n/16)²
(3/16)² cn²
T(1)
Θ(n^log₄3)
n^log₄3
Total: O(n²)
(d)

Q. 1 c)	$S_1 = 8 - 2 = 6$ $S_2 = 1 + 3 = 4$ $S_3 = 7 + 5 = 12$ $S_4 = 4 - 6 = -2$ $S_5 = 1 + 5 = 6$ $S_6 = 6 + 2 = 8$ $S_7 = 3 - 5 = -2$ $S_8 = 4 + 2 = 6$ $S_9 = 1 - 7 = -6$ $S_{10} = 6 + 8 = 14$ * without following strassen's matrix multiplication only answer mentioned — 0 mark $P_1 = (A_{12} + A_{22})(B_{21} + B_{12})$ $P_2 = (A_{21} + A_{22}) B_{11}$ $P_3 = A_{11}(B_{12} - B_{22})$ $P_4 = A_{22}(B_{21} - B_{11})$ $P_5 = (A_{11} - A_{12}) B_{22}$ $P_6 = (A_{21} - A_{22})(B_{11} + B_{1}$ $P_7 = (A_{12} - A_{22})(B_{21} + B_{2}$ $P_1 = 6$ $P_2 = 8$ $P_3 = 72$ $P_4 = -10$ $P_5 = 48$ $P_6 = -12$ $P_7 = -84$ *11 formula's with correct calculation — 5 marks $C_{11} = 48 - 10 - 8 - 12 = 18 = P_1 + P_4 - P_5 + P_7$ $C_{12} = 6 + 8 = 14 = P_3 + P_5$ $C_{21} = 72 - 10 = 62 = P_2 + P_5$ $C_{22} = 48 + 6 - 72 + 84 = 66 = P_1 + P_3 - P_2 + P_6$ So, we get the final result: $\begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix} \Rightarrow$ Correct answer with DAC steps — 1 Ma
Q2. a	

```
Algorithm MaxMin(i, j, max, min)
// a[1 : n] is a global array. Parameters i and j are integers,
// 1 ≤ i ≤ j ≤ n. The effect is to set max and min to the
// largest and smallest values in a[i : j], respectively.
{
    if (i = j) then max := min := a[i]; // Small(P)
    else if (i = j − 1) then // Another case of Small(P)
        {
            if (a[i] < a[j]) then
            {
                max := a[j]; min := a[i];
            }
            else
            {
                max := a[i]; min := a[j];
            }
        }
    else
    {   // If P is not small, divide P into subproblems.
        // Find where to split the set.
            mid := ⌊(i + j)/2⌋;
        // Solve the subproblems.
            MaxMin(i, mid, max, min);
            MaxMin(mid + 1, j, max1, min1);
        // Combine the solutions.
            if (max < max1) then max := max1;
            if (min > min1) then min := min1;
    }
}
```

* Correct logic − 3 marks
* partially correct − 2 marks

gorithm 3.6 Recursively finding the maximum and minimum

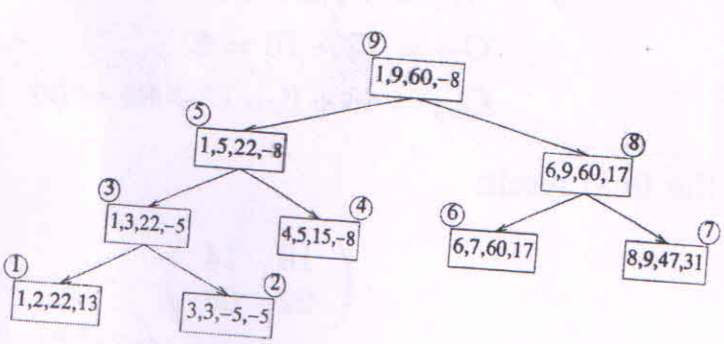


Figure 3.2 Trees of recursive calls of MaxMin

$T(n)$ represents this number, then the resulting recurrence relation is

$$T(n) = \begin{cases} T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil) + 2 & n > 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$ — 0.5 mark

When n is a power of two, $n = 2^k$ for some positive integer k, then

$$\begin{aligned} T(n) &= 2T(n/2) + 2 \\ &= 2(2T(n/4) + 2) + 2 \\ &= 4T(n/4) + 4 + 2 \\ &\vdots \\ &= 2^{k-1}T(2) + \textstyle\sum_{1 \le i \le k-1} 2^i \\ &= 2^{k-1} + 2^k - 2 = 3n/2 - 2 \end{aligned} \tag{3.3}$$ ⟶ 0.5 marks

Note that $3n/2 - 2$ is the best-, average-, and worst-case number of comparisons when n is a power of two.

Q.2b

Total 3 steps - each step 1 mark

Steps to be followed to develop dynamic programming solution

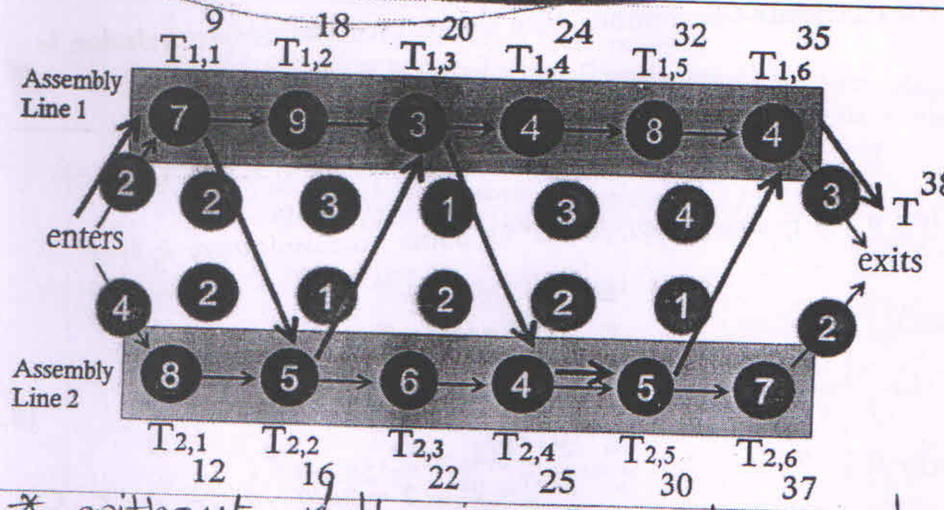
1. Structure of the Optimal Solution
   an optimal solution to the problem "find the fastest way through $S_{1,j}$" contains within it an optimal solution to subproblems: "find the fastest way through $S_{1,j-1}$ or $S_{2,j-1}$".

2. Recursive solution

$$f_1[j] = \begin{cases} e_1 + a_{1,1} & \text{if } j = 1 \\ \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j}) & \text{if } j \ge 2 \end{cases}$$

$$f_2[j] = \begin{cases} e_2 + a_{2,1} & \text{if } j = 1 \\ \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j}) & \text{if } j \ge 2 \end{cases}$$

3. Computing the Optimal Solution
   $f^* = \min(f_1[n] + x_1, f_2[n] + x_2)$
   $f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$
   $f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j})$



* without following procedure only ans. written
−0.5 mark

* Correct answer with matrix — 2.5 mark

OR

(i) By steps (1 mark for each)
at least 3 correct steps

1. A Recursive Solution

Case 1: $x_i = y_j$

e.g.: $X_i = \langle A, B, D, E\rangle$

$Y_j = \langle Z, B, E\rangle$

- Append $x_i = y_j$ to the LCS of X_{i-1} and Y_{j-1}
- Must find a LCS of X_{i-1} and Y_{j-1} $\Rightarrow$ optimal solution to a problem includes optimal solutions to subproblems

Case 2: $x_i \neq y_j$

e.g.: $X_i = \langle A, B, D, G\rangle$

$Y_j = \langle Z, B, D\rangle$

- Must solve two problems
  - find a LCS of X_{i-1} and Y_j: $X_{i-1} = \langle A, B, D\rangle$ and $Y_j = \langle Z, B, D\rangle$
  - find a LCS of X_i and Y_{j-1}: $X_i = \langle A, B, D, G\rangle$ and $Y_j = \langle Z, B\rangle$

2. Overlapping Subproblems

- To find a LCS of X and Y
  - we may need to find the LCS between X and Y_{n-1} and that of X_{m-1} and Y
  - Both the above subproblems has the subproblem of finding the LCS of X_{m-1} and Y_{n-1}

Subproblems share subsubproblems

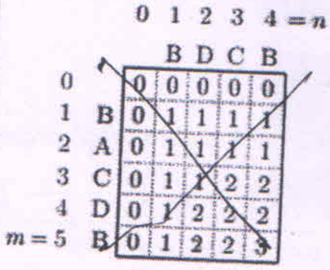
3. Computing the Length of the LCS

$$c[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1 & \text{if } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]) & \text{if } x_i \neq y_j \end{cases}$$

4. Constructing a LCS
- Start at b[m, n] and follow the arrows
- When we encounter a " ↖ " in b[i, j] $\Rightarrow x_i = y_j$ is an element of the LCS

(ii)

LCS length



1 (mark)

$X = \langle BACDB \rangle$

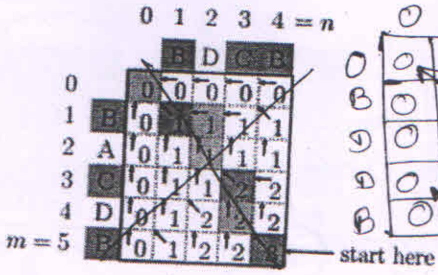
$Y = \langle BDCB \rangle$

BDB

LCS = ~~⟨BCB⟩~~

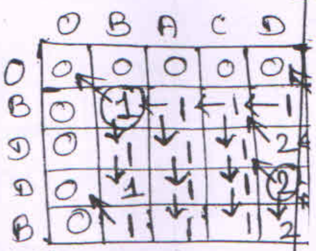
1 (mark)

... with hints



1 mark

2 marks



LCS ✓ B D B

Q.3 b)

```
Huffman(A)
{n = |A|;
 Q = A;
 for i = 1 to n - 1
{  z = new node;
   left[z] =Extract-Min(Q);
   right[z] =Extract-Min(Q);
   f[z] = f[left[z]] + f[right[z]];
   Insert(Q, z);
}
return Extract-Min(Q) root of the tree
}
```

* psedo code expected as steps

An optimal Huffman code would be
0000000 → a
0000001 → b
000001 → c
00001 → d
0001 → e
001 → f
01 → g
1 → h

OR

Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

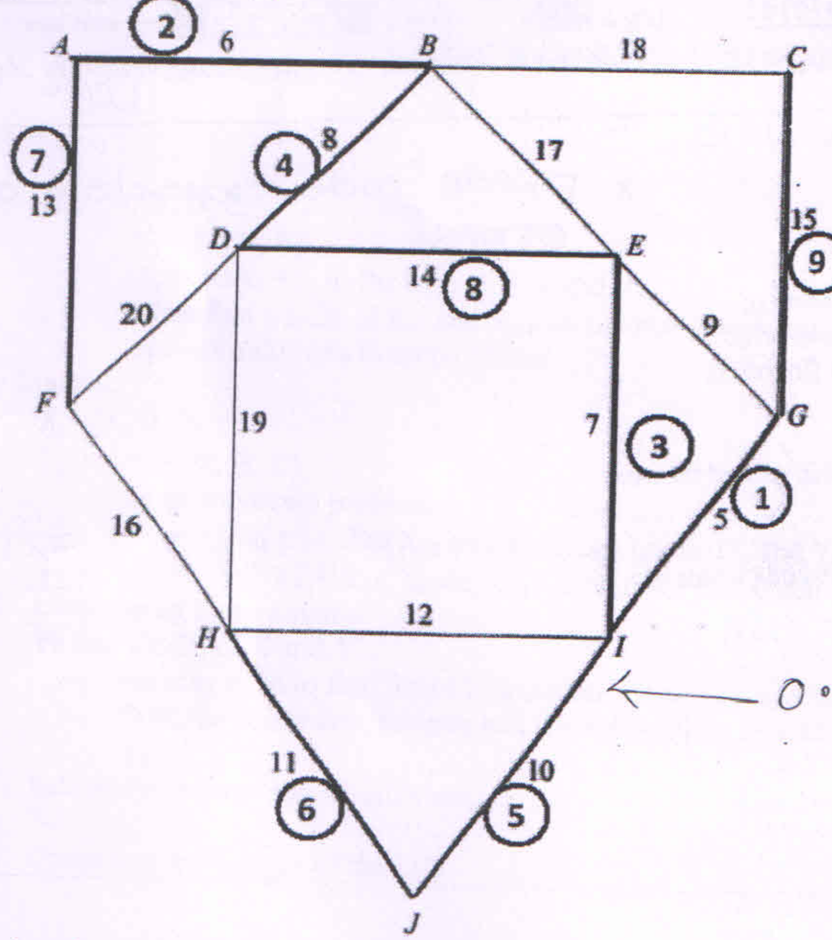
KRUSKAL(V, E, w)

* pseudo code expected
not steps to be followed
* cycle detection logic
must be explained

```
A ← Ø
for each vertex v ∈ V
    do MAKE-SET(v)
sort E into non-decreasing order by w
for each (u, v) taken from the sorted list
    do if FIND-SET(u) ≠ FIND-SET(v)
        then A ← A ∪ {(u, v)}
             UNION(u, v)
return A
```

i. the answer for this part need must give details of edges selected, and in what order

* answer with concept of CC — 2 marks
steps



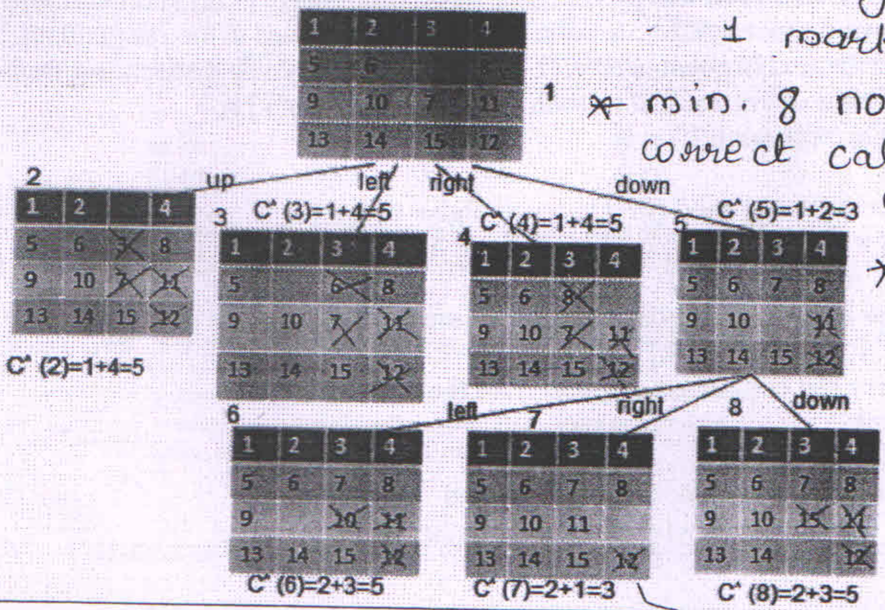
← 0.5 mark

ii. The length of the minimum spanning tree is 89. —— 0.5 mark

* only sorted edges added one by one without logic of cycle detection will not be considered

Q.3 ~~b)~~ a

15 puzzle problem



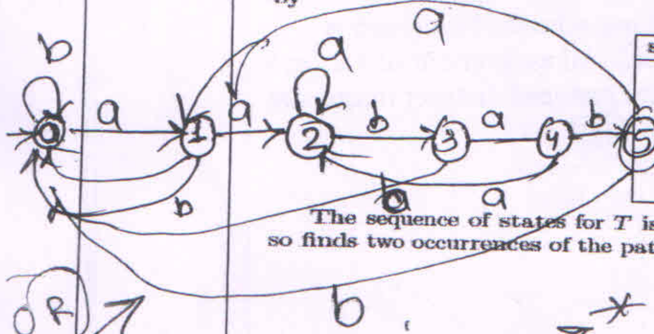
* up to 3 wrong calculation - 1 mark deduct
* min. 8 node with correct calculation - 0.5 each node - (4)
* path - 2
final ans. - 1

= 3 + 0 = 3

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

down

Q.4 a) The states will be $\{0, 1, 2, 3, 4, 5, 6\}$ and having a transition function given by



state	a	b	
0	1	0	a
1	2	0	a
2	2	3	b
3	4	0	a
4	2	5	b
5	1	0	$

— 1 mark.

OR 1 mark

The sequence of states for T is 0, 1, 2, 2, 3, 4, 5, 1, 2, 3, 4, 2, 3, 4, 5, 1, 2, 3, and so finds two occurrences of the pattern, one at $s = 1$ and another at $s = 9$. — 0.5 mark — 0.5 mark

* 0.5 mark deduction up to 3 wrong state seq.

Solution:

We first compute the prefix function.

q	1	2	3	4	5	6	7
$\pi[q]$	0	0	0	0	1	2	3

— 1.5

a b d c a b d

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	a	b	d	c	a	b	a	b	d	c	a	b	d	c	b
q	0	1	2	3	4	5	6	1	2	3	4	5	6	3	4
							2						7		0
							0						3		

— 1.5

occurance — 0.5

Q.4 b) Nextvalue(k)

	X[1]…x[k-1] have been assigned integer value in the range [1,m] such that adjacent vertices have distinct integer. A value for x[k] is determined in the range [0,m]. X[k] is assigned the next highest numbered color while maintaining distinctness from the adj. vertices of vertex k. if no such color exists, then x[k] is 0
	1. Algorithm Nextvalue(k)
	2. { repeat
	3. { X [k]:= (x[k] +1) mod (m+1) ; //next higher color
	4. If (x[k] = 0) then return; // all colors have been used
	5. for j := 1 to n do
	6. { // check if this color is distinct from adjacent colors
	7. If ((G[k, j] != 0) and (x[k] = x[j]))
	8. //if (k , j)is and edge if adj. vertices have the same color.
	9. then break;
	10. }
	11. If (j = n+1) then return; //new color found
	12. } until (false); //otherwise try to find another color
	13. }
	
	Graph Coloring
	This algorithm was formed using recursive backtracking schema. The graph is represented by its boolean adjacency matrix G[1:n,1:n]. All assignment of 1,2..,m to the vertices of the graph such that adjacent vertices are assigned distinct integer are printed. K is the index of the next vertex to color
	1. Algorithm mcoloring(k)
	2. { repeat
	3. { // generate all legal assignment for x[k]
	4. nextvalue(k); //assign to x[k] a legal value
	5. If (x[k] = 0) then return ; //no new color possible
	7. If (k = n) then // at most m color have been used to color the n vertices
	8. Write (x[1 : n];
	9. Else mcoloring(k+1)
	10. } until (false)
	11. }

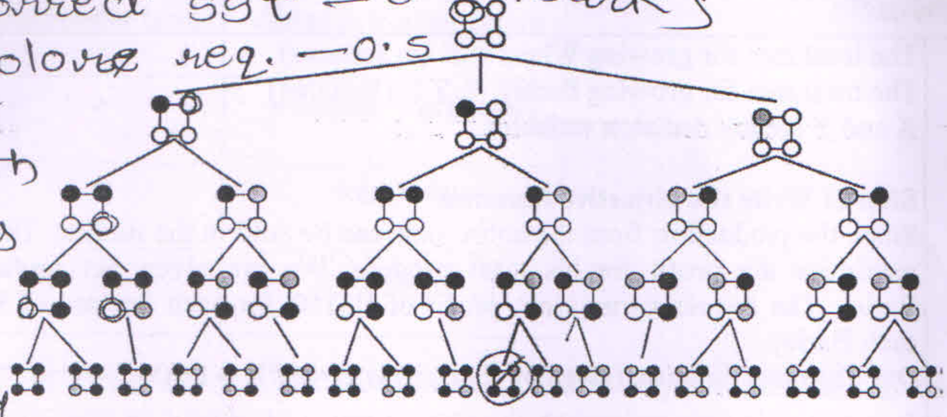
* psudo code as algorithm expected, not steps to be followed.

* Recursive call with backtrack logic expected

* Correct sst – 02 marks

* colour req. – 0.5

* no. of
* with
vertices
colour
mentioned – 0.5 mark



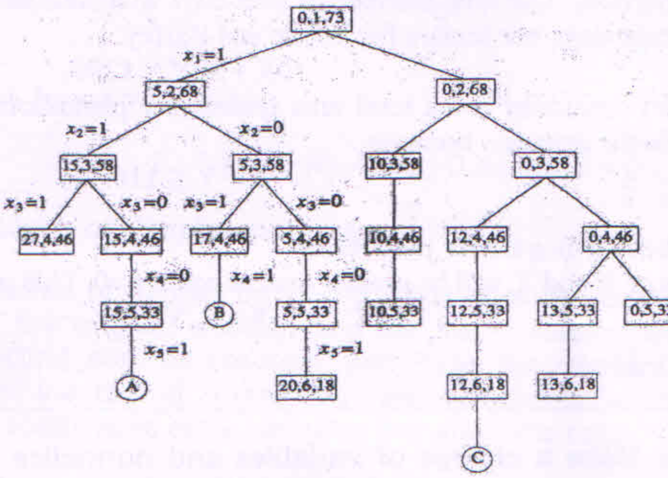
Enumerate all possible states; identify solutions at the leaves.
Naively, non-solutions will be generated. More on that later.
Note, most non-solutions not shown in this cartoon.

60

OR

* pseudo code of sum of subset – 03 mark

*



Q. 5a) A farmer has recently acquired a 110 hectares piece of land. He has decided to grow Wheat and barley on that land. Due to the quality of the sun and the region's excellent climate, the entire production of Wheat and Barley can be sold. He wants to know how to plant each variety in the 110 hectares, given the costs, net profits and labor requirements according to the data shown below:

Variety	Cost (Price/Hec)	Net Profit (Price/Hec)	Man-days/Hec
Wheat	100	50	10
Barley	200	120	30

The farmer has a budget of US$10,000 and an availability of 1,200 man-days during the planning horizon.

Step 1: Identify the decision variables

The total area for growing Wheat = X (in hectares)
The total area for growing Barley = Y (in hectares)
X and Y are my decision variables.

Step 2: Write the objective function

Since the production from the entire land can be sold in the market. The farmer would want to maximize the profit for his total produce. We are given net profit for both Wheat and Barley. The farmer earns a net profit of US$50 for each hectare of Wheat and US$120 for each Barley.

Our objective function (given by Z) is, **Max Z = 50X + 120Y** —— 1 mark

Step 3: Writing the constraints

1. It is given that the farmer has a total budget of US$10,000. The cost of producing Wheat and Barley per hectare is also given to us. We have an upper cap on the total cost spent by the farmer. So our equation becomes:

$$100X + 200Y \leq 10,000$$ —— 1

2. The next constraint is, the upper cap on the availability on the total number of man-days for planning horizon. The total number of man-days available are 1200. As per the table, we are given the man-days per hectare for Wheat and Barley.

$$10X + 30Y \leq 1200$$ —— 1

3. The third constraint is the total area present for plantation. The total available area is 110 hectares. So the equation becomes,

$$X + Y \leq 110$$ —— 1

Step 4: The non-negativity restriction

The values of X and Y will be greater than or equal to 0. This goes without saying.

$$X \geq 0, Y \geq 0$$

——

Q.5. b)

1. **Ans: Make a change of variables and normalize the sign of the independent terms.**

A change is made to the variable naming, establishing the following correspondences:

- x becomes X_1
- y becomes X_2

As the independent terms of all restrictions are positive no further action is required. Otherwise there would be multiplied by "-1" on both sides of the inequality (noting that this operation also affects the type of restriction).

Normalize restrictions.

The inequalities become equations by adding *slack*, *surplus* and *artificial variables* as the following table:

Inequality type	Variable that appears
≥	- surplus + artificial
=	+ artificial
≤	+ slack

In this case, a slack variable (X_3, X_4 and X_5) is introduced in each of the restrictions of ≤ type, to convert them into equalities, resulting the system of linear equations:

$2 \cdot X_1 + X_2 + X_3 = 18$

$2 \cdot X_1 + 3 \cdot X_2 + X_4 = 42$

$3 \cdot X_1 + X_2 + X_5 = 24$

Match the objective function to zero.

$$Z - 3 \cdot X_1 - 2 \cdot X_2 - 0 \cdot X_3 - 0 \cdot X_4 - 0 \cdot X_5 = 0$$

Write the initial tableau of Simplex method.

The initial tableau of Simplex method consists of all the coefficients of the decision variables of the original problem and the slack, surplus and artificial variables added in second step (in columns, with P_0 as the constant term and P_i as the coefficients of the rest of X_i variables), and constraints (in rows). The C_b column contains the coefficients of the variables that are in the base.

The first row consists of the objective function coefficients, while the last row contains the objective function value and *reduced costs* $Z_j - C_j$.

The last row is calculated as follows: $Z_j = \Sigma(C_{bi} \cdot P_j)$ for $i = 1..m$, where if $j = 0$, $P_0 = b_i$ and $C_0 = 0$, else $P_j = a_{ij}$. Although this is the first tableau of the Simplex method and all C_b are null, so the calculation can simplified, and by this time $Z_j = -C_j$.

Tableau I . 1st iteration							
			3	2	0	0	0
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5

P3	0	18	2	1	1	0	0
P4	0	42	[illegible]	3	0	1	0
P5	0	24	3	1	0	0	1
Z		0	-3	-2	0	0	0

Stopping condition.

If the objective is to maximize, when in the last row (indicator row) there is no negative value between discounted costs (P_1 columns below) the stop condition is reached.

In that case, the algorithm reaches the end as there is no improvement possibility. The Z value (P_0 column) is the optimal solution of the problem.

Another possible scenario is all values are negative or zero in the input variable column of the base. This indicates that the problem is not limited and the solution will always be improved.

Otherwise, the following steps are executed iteratively.

Choice of the input and output base variables.

First, input base variable is determined. For this, column whose value in Z row is the lesser of all the negatives is chosen. In this example it would be the variable X_1 (P_1) with -3 as coefficient.

If there are two or more equal coefficients satisfying the above condition (case of tie), then choice the basic variable.

The column of the input base variable is called *pivot column* (in green color).

Once obtained the input base variable, the output base variable is determined. The decision is based on a simple calculation: divide each independent term (P_0 column) between the corresponding value in the pivot column, if both values are strictly positive (greater than zero). The row whose result is minimum score is chosen.

If there is any value less than or equal to zero, this quotient will not be performed. If all values of the pivot column satisfy this condition, the stop condition will be reached and the problem has an unbounded solution (see Simplex method theory).

In this example: 18/2 [=9] , 42/2 [=21] and 24/3 [=8]

The term of the pivot column which led to the lesser positive quotient in the previous division indicates the row of the slack variable leaving the base. In this example, it is X_5 (P_5), with 3 as coefficient. This row is called *pivot row* (in **green**).

If two or more quotients meet the choosing condition (case of tie), other than that basic variable is chosen (wherever possible).

The intersection of *pivot column* and *pivot row* marks the *pivot value*, in this example, 3.

Update tableau.

The new coefficients of the tableau are calculated as follows:

- In the pivot row each new value is calculated as:

New value = Previous value / Pivot

- In the other rows each new value is calculated as:

*New value = Previous value - (Previous value in pivot column * New value in pivot row)*

So the pivot is normalized (its value becomes 1), while the other values of the pivot column are canceled (analogous to the Gauss-Jordan method).

Calculations for P_4 row are shown below:

Previous P_4 row	42	2	3	0	1	0
	-	-	-	-	-	-
Previous value in pivot column	2	2	2	2	2	2
	x	x	x	x	x	x
New value in pivot row	8	1	1/3	0	0	1/3
	=	=	=	=	=	=
New P_4 row	26	0	7/3	0	1	-2/3

The tableau corresponding to this second iteration is:

Tableau II . 2nd iteration							
			3	2	0	0	0

Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5
P_3	[illegible]	2	0	1/3	1	0	-2/3
P_4	0	26	0	7/3	0	1	-2/3
P_1	3	8	1	1/3	0	0	1/3
Z		24	0	-1	0	0	1

When checking the stop condition is observed which is not fulfilled since there is one negative value in the last row, -1. So, continue iteration steps 6 and 7 again.

- 6.1. The input base variable is X_2 (P_2), since it is the variable that corresponds to the column where the coefficient is -1.
- 6.2. To calculate the output base variable, the constant terms P_0 column) are divided by the terms of the new pivot column: 2 / 1/3 [=6] , 26 / 7/3 [=78/7] and 8 / 1/3 [=24]. As the lesser positive quotient is 6, the output base variable is X_3 (P_3).
- 6.3. The new pivot is 1/3.
- 7. Updating the values of tableau again is obtained:

Tableau III . 3rd iteration

			3	2	0	0	0
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5
P_2	2	6	0	1	3	0	-2
P_4	0	12	0	0	-7	1	4
P_1	3	6	1	0	-1	0	1
Z		30	0	0	3	0	-1

Checking again the stop condition reveals that the pivot row has one negative value, -1. It means that optimal solution is not reached yet and we must continue iterating (steps 6 and 7):

- 6.1. The input base variable is X_5 (P_5), since it is the variable that corresponds to the column where the coefficient is -1.
- 6.2. To calculate the output base variable, the constant terms (P_0) are divided by the terms of the new pivot column: 6/(-2) [=-3] , 12/4 [=3] , and 6/1 [=6]. In this iteration, the output base variable is X_4 (P_4).
- 6.3. The new pivot is 4.
- 7. Updating the values of tableau again is obtained:

Tableau IV . 4th iteration

			3	2	0	0	0
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5
P_2	2	12	0	1	-1/2	1/2	0
P_5	0	3	0	0	-7/4	1/4	1
P_1	3	3	1	0	3/4	-1/4	0
Z		33	0	0	5/4	1/4	0

End of algorithm.

It is noted that in the last row, all the coefficients are positive, so the stop condition is fulfilled.

The optimal solution is given by the val-ue of Z in the constant terms column (P_0 column), in the example: 33. In the same column, the point where it reaches is shown, watching the corresponding rows of input decision variables: $X_1 = 3$ and $X_2 = 12$.

Undoing the name change gives x = 3 and y = 12.