Hall Ticket Number:

20CB404/22CM404/20CS404/20DS404/20IT404

II/IV B. Tech (Regular) DEGREE EXAMINATION

July, 2025

Fourth Semester

Common to CB, CM, CS, DS & IT

Design and Analysis of Algorithms

Time: Three Hours

Maximum: 70 Marks

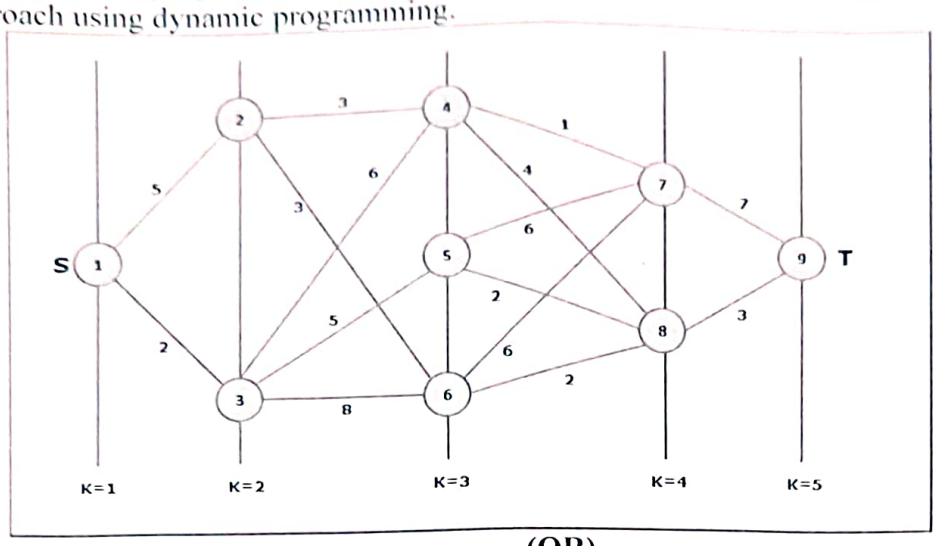
*Answer question 1 compulsorily.* (14X1 = 14 Marks)

*Answer one question from each unit.* (4X14 = 56 Marks)

| | | | CO | BL | M |
|---|---|---|---|---|---|
| 1 | a) | Define an Algorithm. | CO1 | L1 | 1M |
| | b) | What is a recurrence relation? | CO1 | L1 | 1M |
| | c) | What is the asymptotic notation for Worst-case Analysis? | CO1 | L1 | 1M |
| | d) | Write the General Method of Divide and Conquer. | CO2 | L1 | 1M |
| | e) | What is the recurrence relation for Merge Sort? | CO2 | L1 | 1M |
| | f) | What does Strassen's algorithm reduce in matrix multiplication? | CO2 | L2 | 1M |
| | g) | Which algorithm solves the single-source shortest path for non-negative weights? | CO2 | L1 | 1M |
| | h) | Define the key idea behind dynamic programming. | CO3 | L1 | 1M |
| | i) | Identify the main goal of the reliability design problem. | CO3 | L1 | 1M |
| | j) | Explain the term "articulation point". | CO3 | L1 | 1M |
| | k) | Highlight a key difference between DFS and BFS. | CO4 | L1 | 1M |
| | l) | Write the General Method of Backtracking. | CO4 | L1 | 1M |
| | m) | List two example problems solved using Branch and Bound. | CO4 | L1 | 1M |
| | n) | Name the class of problems that are the hardest among NP problems. | CO4 | L2 | 1M |
| | | **Unit-I** | | | |
| 2 | a) | Define an Algorithm. Explain characteristics of an Algorithm. | CO1 | L2 | 7M |
| | b) | Write an algorithm to find factorial using recursion and find its time complexity. | CO1 | L2 | 7M |
| | | **(OR)** | | | |
| 3 | a) | Explain pseudo code for expressing an algorithm with suitable example. | CO1 | L2 | 7M |
| | b) | Explain the Master's theorem and find time complexity of T(n) = 2T(n/2) + n | CO1 | L2 | 7M |
| | | **Unit-II** | | | |
| 4 | a) | Write an algorithm for Quick Sort and find its Complexity. | CO2 | L3 | 7M |
| | b) | Discuss in detail about Strassen's Matrix Multiplication. | CO2 | L3 | 7M |
| | | **(OR)** | | | |
| 5 | a) | Given the jobs, their deadlines (di) and associated profits (pi) as, (p1, p2, p3, p4) = (100, 10, 15, 27) and (d1, d2, d3, d4) = (2, 1, 2, 1). Calculate the maximum profit using Greedy Method. | CO2 | L3 | 7M |
| | b) | Explain about Krushkal's Algorithm with an example. | CO2 | L3 | 7M |
| | | **Unit-III** | | | |
| 6 | a) | Solve 0/1 knapsack problem using dynamic programming for n=4, M=6, profits(p1,p2,p3,p4) = (3,4,5,6) weights(w1,w2,w3,w4) = (2,3,4,5) and provide an optimal solution. | CO3 | L3 | 7M |

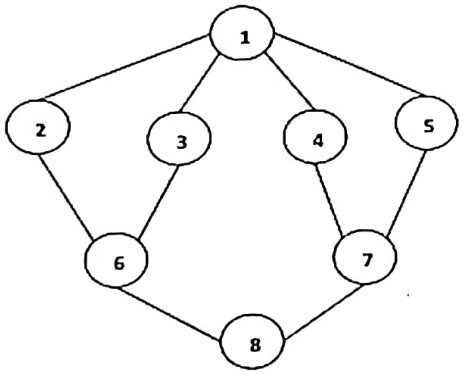
**P.T.O**

b) Apply multistage graph algorithm for the following graph that uses forward approach using dynamic programming. CO3 L3 7M




**(OR)**

7 a) In what order will the nodes be visited using BFS and DFS traversals for the following graph? Explain. CO3 L3 7M




b) Differentiate Bi Connected Components and Strongly Connected Components with an appropriate example. CO3 L3 7M

## Unit-IV

8 a) Draw and analyse the solution tree for the given Sum of Subsets problem using Backtracking. S = {3,5,6,7} and M = 15. CO4 L3 7M

b) Draw and analyse the solution tree for the 0/1 knapsack problem using LC Branch and Bound method for the following data M=15, n=4, (p1, p2, p3, p4) = (10, 10, 12, 18), (w1, w2, w3, w4) = (2, 4, 6, 9). CO4 L3 7M

**(OR)**

9 a) Explain the basic concepts of P, NP, NP-Complete, and NP-Hard problems with examples. CO4 L3 7M

b) State and explain Cook's Theorem CO4 L3 7M

# DESIGN AND ANALYSIS OF ALGORITHMS

Fourth Semester

1 Mark Questions:-

1. a) Define an Algorithm. [1M]

Algorithm is a step-by-step process which is used to solve the given prblm. (or)

An Algorithm is a finite set of instructions to accomplish the given task.

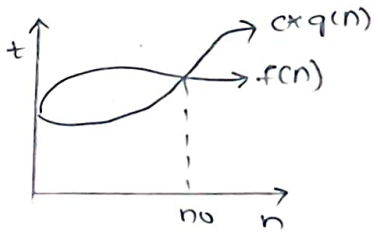
b) What is a recurrence relation? [1M]

A recurrence relation is a mathematical formula which is used to solve the given prblm and it is commonly used to express the time complexity of recursive algorithms.

general form : $T(n) = aT(n/b) + f(n)$

c) What is the asymptotic notation for worst-case Analysis? [1M]

The asymptotic notation used for worst case Analysis is Big-O (O) notation.




$f(n) \leq c * g(n)$

d) Write the General Method for Divide and conquer. [1M]

In this approach we can divide the given prblm into no of sub-prblms after we should combine that sub prblms, then we can get the solution.

Divide: It is nothing but we can divide the given prblm into different sub prblms.

conquer: It is nothing but we can solve the diff. sub prblms.

combine or merge : It is nothing but we can combine the different sub prblms and finally we can get the solution.

e) What is the recurrence relation for Merge sort? [1M]

The recurrence relation for Merge sort is $T(n) = 2T(n/2) + n$.

f) What does Strassen's algorithm reduce in matrix Multiplication? [1M]

Strassen's Alg. reduces the no. of multiplication steps and it also reduces time complexity from $O(n^3)$ to $O(n^{2.807})$.

g) Which algorithm solves the single-source shortest path for non-negative weights? [1M]

Dijkstra's Algorithm solves the single-source shortest path for non-negative weights.

h) Define the key idea behind dynamic programming. [1M]

Dynamic programming solves complex prblms by breaking them into overlapping subproblems, storing the results of subprblms to avoid redundant computations.

→ It produces optimal solutions.

i) Identify the main goal of the reliability design prblm. [1M]

The main goal is to maximize the reliability of a system or product while minimizing cost or resource usage, often by using the redundancy.

j) Explain the term "articulation point". [1M]

The deleted vertices position from the given graph, which is used to produce different no. of biconnected components are called as Articulation point

k) Highlight a key difference b/w DFS and BFS. [1M]

BFS:-

BFS follows the queue datastructure i.e FIFO approach.

DFS:

DFS follows the stack datastructure i.e LIFO approach.

l) Write the General Method of Backtracking. [1M]

The General Method of backtracking approach is

1. Decision problem : These are also known as feasible solutions.
2. Optimization prblm : These prblms produce either max. or min. solutions. These are also known as better solutions.
3. Enumeration prblm : All the feasible solutions can be comes under enumeration and this prblms can choose all the possibilities.

* Backtracking can follows the "brute-force."

m. List two examples problems solved by using Branch and Bound. [1M]

The two ex. prblms solved by using branch and bound technique are

1. 0/1 knapsack prblm
2. Travelling tr sales person prblm.

[1M]

n. Name the class of problems that are the hardest among NP prblms?

The hardest problems in NP are called NP- complete problems.

Essay Questions

UNIT I

2 a) Define an Algorithm. Explian characteristics of an Algorithm. [7M]

Algorithm:-

An Algorithm is a step-by-step process used to solve a given problem.

(or)

An algorithm is a finite set of instructions to accomplish the given task.

characteristics of an Algorithm:-

1. Input: We should give atleast 0 or more no.of inputs to an algorithm.

2. Output: An algorithm should produce atleast one or more no.of outputs.

3. Finiteness: Each and every alg. must be terminated a finite no.of steps.

4. Definiteness: Each and every line of the alg. must be write a definite values, it can follows unambiquous data only.

5. Effectiveness: Each and every line of the alg. We must write in an effective manner.

2 b) Write an algorithm to find factorial using recursion and find its time complexity. [7M]

Recursion:- A fuctions that calls is itself directly or indirectly to solve a problem is called recursion.

Alg to find factorial using recursion:-

```
Alg for factorial(n)  — 0
{                     — 0
f = factorial(n)      — 0
```

```
for i ← 1 to n do          - n+1
                           - 0
{
if (n==0 || n==1) then     - 1
        return 1;          - 1
 else                      - 0
    return n*factorial(n-1); - 1
 }                         - 0
}                          - 0
                           O(n+4)
```

Time complexity is O(n)

(or)

3 a) Explain pseudo code for expressing an algorithm with suitable example.

- Pseudo code is an Hypothetical language, not a programming language.
- Whenever you can write the pseudo code for expressing an alg. the we must follow the steps:

1. comment:

comments are used to write to understand the prblm easily for the users.

There are two types of comments they are

1. single line comments and these are represented by //---
2. Multiple line comments and these are represented by

```
/*---
---*/
```

2. Blocks are indicating with matching braces. { ---- }

3. Identifiers must start with letters only.

A to Z (or) a to z.

4. Assiging values to variables by using Assignment operators.

:= column equal to

← backward indicating arrow.

5. Each and every statement should be seperated by using the semicolon. ;

6. looping statements: There are three types

1. for loop:

```
for i ← 1 to n do
  {
  ==
  }
```

2. While loop:

```
while (condition)
  {
  ==
  }
```

3. repeat-until:

```
Repeat
{
==
}until (condition).
```

7. conditional control statements: There are two types

1. if:

```
if (condition) then
  {
  statement;
  }
```

2. if-else:-

```
if (condition) then
  {
   statement-1;
  }
else {
       statement-2;
       }
```

8. Switch case:
To perform different types of operations we generally use switch case.

```
switch (choice)
   {
   case1: statement;
          break;
   ==
   default:
          default statement;
   }
```

9. Whenever we use logical and relational operators then the alg. will produce boolean values ie either true or false.

10. Each & every alg. must start with alg. followed by algorithm name & its body.

```
Algorithm Alg-name (parameter list)
  {
   //body
  }
```

Ex:- Algorithm for matrix Addition (n, B, C, i, j)

```
{
for i ← 0 to n-1 do
  {
   for j ← 0 to n-1 do
     {
      c[i,j] = A[i,j] + B[i,j];
     }
    return c[i,j];
  }
}
```

b.) Explain the Master's theorem and find time complexity of $T(n) = 2T(n/2) + 2n$.

Master's Theorem :-

This theorem is used to calculate the time complexity, this theorem can be apply on the recurrance relation.

Some of the recurrance relations cannot be solved by using this theorem.

$$T(n) = aT(n/b) + \Theta(n^k \log^p n)$$

where $a \geq 1$, $b > 1$, $k \geq 0$ and $p$ is real number.

case 1:

if $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$

case 2:

if $a = b^k$

a) if $p > -1$ then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

b) if $p = -1$ then $T(n) = \Theta(n^k \log \log n)$

c) if $p < -1$ then $T(n) = \Theta(n^{\log_b a})$

case 3:

if $a < b^k$

a) if $p \geq 0$ then $T(n) = \Theta(n^k \log^p n)$

b) if $p < 0$ then $T(n) = \Theta(n^k)$.

i) $T(n) = 2T(n/2) + n$

$a = 2,\ b = 2,\ k = 1,\ p = 0$

$b^k = 2$ which satisfies $a = b^k$ and $p = 0$ then

$T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$= \Theta(n^{\log_2 2} \log^1 n)$

$T(n) = \Theta(n \log n)$.

UNIT-II

4 a) Write an algorithm for Quick sort and find its complexity.

Alg for Quicksort pivot as a 1st element:

```
Alg for Quicksort(A, lb, ub) {
 if (lb < ub) then {
    p := partition(A, lb, ub);
    Quicksort(A, lb, p-1);
    Quicksort (A, p+1, ub); }
 }
Alg for partition (A, lb, ub) {
  start ← lb;
  end ← ub;
  pivot ← A(lb);
  While (lb < ub) {
 While(A(start] <= pivot) {
    start++; }
 while (A[end] > pivot){
          end--;
         }
 if (start < end) {
 Swap(A(start] with A(end]); }
 else {
  swap [A[lb] with A[end]); }
   return end;
    }
  }
```

Alg for Quicksort pivot as a last element:

```
Alg for Quicksort(A, lb, ub) {
if (lb < ub) then
 {
 p := partition (A, lb, ub);
 quicksort (A, lb, q-1);
 quicksort (A, q+1, ub); }
 }
Alg for partition (A, lb, ub) {
      j ← lb;
      i ← lb-1;
      pivot ← A[ub];
for j ← lb to ub-1 do
      {
    if (a[j] < pivot)
      {
       i++;
  Exchange A[j] with A[i];
      }
    }
 Exchange A[i+1] with A[ub];

    }
```

The best & Average case time complexity of quick sort :-

$T(n) = T(n/2) + T(n/2) + n$

$T(n) = 2T(n/2) + n \rightarrow (1)$

put $n = n/2$ in eq (1) then

$T(n/2) = 2T(n/4) + n/2 \rightarrow (2)$

sub (2) in (1)

$T(n) = 2[2T(n/4) + n/2] + n$

$T(n) = 4T(n/4) + 2n \rightarrow (3)$

put $n = n/4$ in (1)

$T(n/4) = 2[T(n/8)] + n/4 \rightarrow (4)$

sub (4) in (3)

$T(n) = 4[2T(n/8) + n/4] + 2n$

$T(n) = 8T(n/8) + 3n$

$= 2^3 T(n/8) + 3n$

$= 2^3 T(n/2^3) + 3n$

let $k = 3$ and $n = 2^k$

$k = \log_2 n$

$T(n) = 2^k T(n/2^k) + nk$

$T(n) = nT(n/n) + n\log_2 n$

$T(n) = n(T(1)) + n\log_2 n$

$T(n) = n + n\log_2 n$

$T(n) = n\log n$

$\cong \Theta(n \log n)$.

The worst case time complexity of quick sort :

$T(n) = T(n-1) + T(n) \rightarrow (1)$

put $n = n-1$ in (1) then

$T(n-1) = T(n-2) + T(n-1) \rightarrow (2)$

sub (2) in (1) then

$T(n) = T(n-2) + T(n-1) + T(n) \rightarrow (3)$

put $n = n-2$ in (1) then

$T(n-2) = T(n-3) + T(n-2) \rightarrow (4)$

sub (4) in (3) then

$T(n) = T(n-3) + T(n-2) + T(n-1) + T(n)$

It is in the form of sum of the series then

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

$T(n) = \frac{n^2+n}{2}$

We can remove constant then

$T(n) = n^2 + n$

The worst case time complexity is $\Theta(n^2)$.

6.) Discuss in detail about strassen's Matrix Multiplication.

1. It is one of the application in divide & conquer approach.
2. It is used to reduce no. of matrix multiplication steps and time complexity.
3. Whenever a matrix size may 2x2 then the formulas may as follows:

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

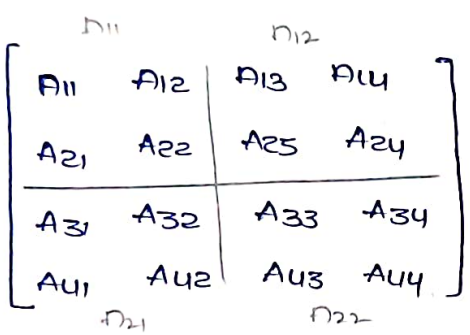
$C_{11} = (A_{11} * B_{11}) + (A_{12} * B_{21})$

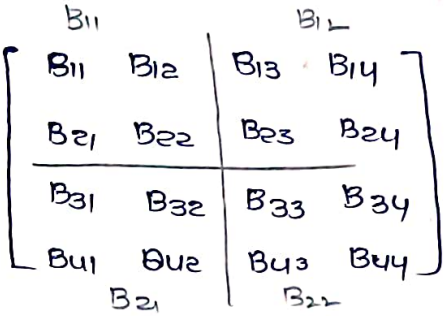
$C_{12} = (A_{12} * B_{12}) + (A_{12} * B_{22})$

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

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

4. Whenever a matrix size may be 4X4 (or) 8x8 then we can divide that matrix with size/2.




$C_{11} = MM(A_{11}, B_{11}, n/2) + MM(A_{12}, B_{21}, n/2)$

$C_{12} = MM(A_{11}\ B_{12}\ n/2) + MM(A_{12}\ B_{22}\ n/2)$

$C_{21} = MM(A_{21}\ B_{11}\ n/2) + MM(A_{22}\ B_{21}\ n/2)$

$C_{22} = MM(A_{21}\ B_{12}\ n/2) + MM(A_{22}\ B_{22}\ n/2)$

To reduce the the matrix multiplication count strassen's can introduce the following formulas.

$P = (A_{11} + A_{22}) \times (B_{11} + B_{22})$

$Q = B_{11}(A_{21} + A_{22})$

$R = A_{11}(B_{12} - A_{22})$

$S = B_{22}(B_{21} - B_{11})$

$T = A_{22}(A_{11} + A_{12})$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (B_{21} + B_{22})(A_{12} - A_{22})$

The final matrix will be

$$C_{11} = P+S-T+V$$

$$C_{12} = R+T$$

$$C_{21} = Q+S$$

$$C_{22} = P+R-Q+U$$

Time complexity for Strassen's matrix multiplication – $O(n^{2.807})$.

(Or)

5 a) Given the jobs, their deadlines (di) and associated profits (pi) as $(P_1 - P_4) = (100, 10, 15, 27)$ and $(d_1 -- d_4) = (2, 1, 2, 1)$. calculate the max. profit using Greedy Method.

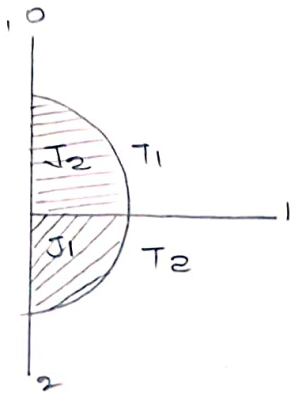
Given no of objects (n) = 4

$(P_1, P_2, P_3, P_4) = (100, 10, 15, 27)$ & $(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$.

First we should arrange jobs in descending order then

| $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|---|---|---|---|
| 100 | 27 | 15 | 10 |
| 2 | 1 | 2 | 1 |

Here the max. deadline is 2, then based on max. deadline we should divide the time slots as following.




* Job1 can be placed in $T_1$ and Job2 can be placed in $T_2$ and there is no chance to place Job3 and Job4.

| Assign Job | Assign Req. time slot | Solution | Profit. |
|---|---|---|---|
| {-} | - | $\phi$ | 0 |
| {J1} | {2,1} | 2 | 100 |
| {J2} | {2,1} | 1 | 27 |
| {J3} | {2,1} | - | - |
| {J4} | {2,1} | - | - |

Profit = 100 + 27 = 127.

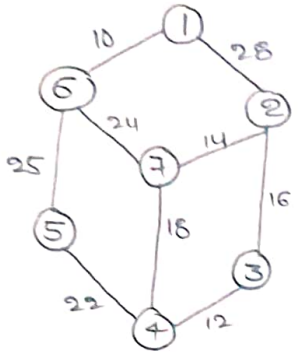
b) Explain about Kruskhal's Algorithm with an example.

1. Krushkal's method is one of the application in Greedy Method.
2. It is used to find minimum cost spanning tree.

Algorithm for Krushkal's method:

```
Alg for Krushkal's method (n, E, cost, t)
{
 i ← 0;
 min_cost ← 0;
 while (i<=n-1) do
 {
 Delete minimum cost edge (u,v);
    j ← find (u);
    k ← find (v);
    if (j≠k)
    {
      i ← i+1;
      t[i:1] ← u;
      t[i:2] ← v;
    min_cost := min_cost + cost (u,v);
    union (j,k);
    }
  }
}
```

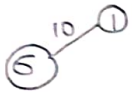
Ex: Solve the following graph by kruskhal's method.



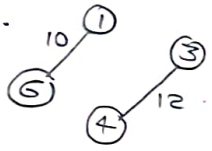

1. To solve the prblm by using krushkal's method we should arrange the edge cost in increasing order.

| edge | cost |
|---|---|
| (1,6) | 10 |
| (3,4) | 12 |
| (2,7) | 14 |
| (2,3) | 16 |
| (7,4) | 18 |
| (5,4) | 22 |
| (6,7) | 24 |
| (6,5) | 25 |
| (1,2) | 28 |

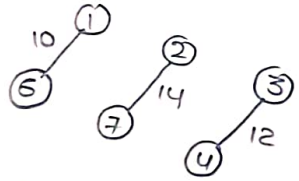
1. The first edge is (1,6) and its min. cost is 10.



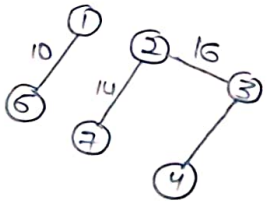

2. The second edge is (3,4) and its min. cost is 12.




3 The third edge is (2,7) and its min. cost is 14

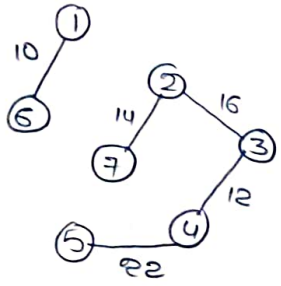



4. The next edge is (2,3) and its min. cost is 16, we can directly draw edge blw 2,3.

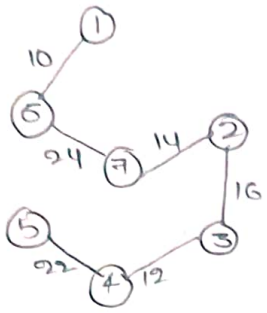



5. The next edge (7,4) forms a cycle so remove that edge.

6. The sixth edge is (5,4) and its min. cost is (22).

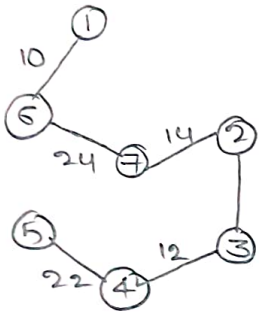


7.) The seventh edge is (6,7) and its min cost is 24 and we can draw direct edge b/w 6,7.




8.) The eigth edge (6,5) and last edge (1,2) both forms a cycle then we can remove them.

Req. min-cost spanning tree is




Minimum cost for the above spanning tree is

$10 + 24 + 14 + 6 + 12 + 22$

$= 98.$

98 is the min-cost.

---

UNIT-III

6 a.) Solve 0/1 knapsacks problem using dynamic programming for n=4, M=6, profits $(P_1, P_2, P_3, P_4) = (3, 4, 5, 6)$ weights $(W_1, - W_4) = (2, 3, 4, 5)$ and provide an optimal solution.

Given $(P_1 - P_4) = (3, 4, 5, 6)$

$(W_1 - W_4) = (2, 3, 4, 5)$

The 0/1 knapsack prblm always maximizes $\sum_{1 \le x_i \le n} x_i p_i$ and

subjected to $\sum_{0 \le x_i \le 1} w_i x_i$

2. To solve the 0/1 knapsack by using dynamic programming approach then we must follow two operations for each & every stage.

Addition : $S_1^i = S^{i-1} + (P_i, W_i)$ Union : $S^i = S^{i-1} + S_1^i$

intially $S^0 = \{0,0\}$

i=1:

$S_1^1 = S^0 + (P_1 W_1)$

$= (0,0) + (3,2)$

$S_1^1 = (3,2)$

$S^1 = S^0 + S_1^1$

$S^1 = (0,0)\ (3,2)$

i=2:

$S_1^2 = S^1 + (P_2, W_2)$

$= (0,0)\ (3,2) + (4,3)$

$S_1^2 = (4,3)\ (7,5)$

$S^2 = S^1 + S_1^2$

$S^2 = (0,0)\ (3,2)\ (4,3)\ (7,5)$

i=3:

$S_1^3 = S^2 + (P_3, W_3)$

$= (0,0)\ (3,2)\ (4,3)\ (7,5) + (5,4)$

$S_1^3 = (5,4)\ (8,6)\ (9,7)\ (12,9)$

$S^3 = (0,0)\ (3,2)\ (4,3)\ (7,5)\ (5,4)\ (8,6)\ (9,7)\ (12,9)$ $[S^3 = S^2 + S_1^3]$

i=4:

$S_1^4 = S^3 + (P_4, W_4)$

$= (0,0)\ (3,2)\ (4,3)\ (7,5)\ (5,4)\ (8,6)\ (9,7)\ (12,9) + (6,5)$

$S_1^4 = (6,5)\ (9,7)\ (10,8)\ (13,10)\ (11,9)\ (14,11)\ (15,12)\ (18,14)$

$S^4 = S^3 + S_1^4$

$S^4 = (0,0)\ (3,2)\ (4,3)\ (7,5)\ (5,4)\ (8,6)\ (9,7)\ (12,9)\ (6,5)\ (9,7)\ (10,8)$
$(13,10)\ (11,9)\ (14,11)\ (15,12)\ (18,14)$

now we should arrange the above sets ac in ascending order with corresponding to weights then

$S^4 = (0,0)\ (3,2)\ (4,3)\ (5,4)\ (6,5)\ (7,5)\ (8,6)\ (9,7)\ (12,9)$

$S^4 = (0,0)\ (3,2)\ (4,3)\ (5,4)\ (6,5)\ (7,5)\ (8,6).$

Now we should apply purging rule :

If $(P_i, W_i) \in S^n$ and $(P_i, W_i) \notin S^{n-1}$ then $x_i = 1$ otherwise $x_i = 0$.

$(8,6) \in S^4$ and $(8,6) \notin S^3$ false then $x_4 = 0$.

$(8,6) \in S^3$ and $(8,6) \notin S^2$ true then $x_3 = 1$ and

$(8,6) - (5,4) = (3,2)$

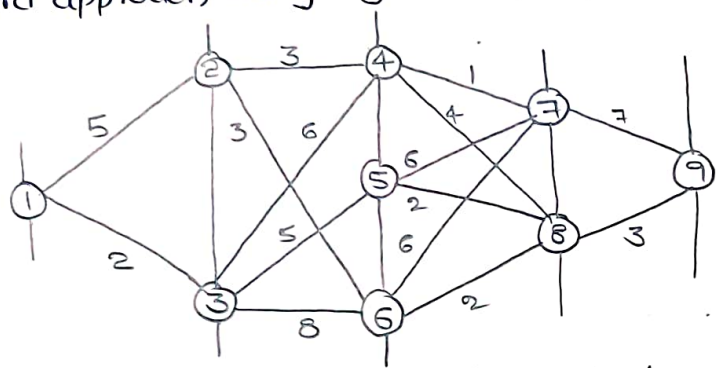
$(3,2) \in S^2$ and $(3,2) \notin S^1$ false then $x_2 = 0$ and

$(3,2) \in S^1$ and $(3,2) \notin S^0$ true then $x_1 = 1$.

now $x_1 = 1$, $x_2 = 0$ $x_3 = 1$ and $x_4 = 0$.

Weight $= 2 + 4 = 6$.

6) Apply multistage graph algorithm for the following graph that uses forward approach using dynamic programming.




Multistage graph using forward approach :-

| vertices | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| cost | 12 | 8 | 10 | 7 | 5 | 5 | 7 | 3 | 0 |
| destination | 3 | 6 | 5 | 8 | 8 | 8 | 9 | 9 | 9 |

Stage 5: $i = 5$ and $j = 9$

cost $(5,9) = (0,0)$

Stage 4:

$i = 4$ and $j = 7, 8$ and for 7, 8 destination (e) is 9

cost $(4,7) = \min\{\text{cost}(7,9) + \text{cost}(5,9)\}$

$= \min\{7 + 0\}$

cost $(4,7) = 7$

cost $(4,8) = \min\{\text{cost}(8,9) + \text{cost}(5,9)\}$

$= \min\{3 + 0\}$

cost $(4,8) = 3$.

Stage 3:

i=3 and j= 4,5,6 destination vertex (d) for 4,5,6 is 7,8.

$$cost(3,4) = \min\begin{Bmatrix} cost(4,7) + cost(4,7) \\ cost(4,8) + cost(4,8) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 1+7 \\ 4+3 \end{Bmatrix} = \min\begin{Bmatrix} 8 \\ 7 \end{Bmatrix}$$

$cost(3,4) = 7.$

$$cost(3,5) = \min\begin{Bmatrix} cost(5,7) + cost(4,7) \\ cost(5,8) + cost(4,8) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 6+7 \\ 2+3 \end{Bmatrix} = \min\begin{Bmatrix} 13 \\ 5 \end{Bmatrix}$$

$cost(3,5) = 5$

$$cost(3,6) = \min\begin{Bmatrix} cost(6,7) + cost(4,7) \\ cost(6,8) + cost(4,8) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 6+7 \\ 2+3 \end{Bmatrix} = \min\begin{Bmatrix} 13 \\ 5 \end{Bmatrix}$$

$cost(3,6) = 5$

Stage 2:

i=2 and j= 2,3

$$cost(2,2) = \min\begin{Bmatrix} cost(2,4) + cost(3,4) \\ cost(2,6) + cost(3,6) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 3+7 \\ 3+5 \end{Bmatrix} = \min\begin{Bmatrix} 10 \\ 8 \end{Bmatrix}$$

$cost(2,2) = 8.$

$$cost(2,3) = \min\begin{Bmatrix} cost(3,4) + cost(3,4) \\ cost(3,5) + cost(3,5) \\ cost(3,6) + cost(3,6) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 6+7 \\ 5+5 \\ 8+5 \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 13 \\ 10 \\ 13 \end{Bmatrix}$$

$cost(2,3) = 10.$

Stage 1:

i=1 , j=1 destination vertices for 1 is 2,3

$$cost(1,1) = \min \begin{cases} cost(1,2) + cost(2,2) \\ cost(1,3) + cost(2,3) \end{cases}$$

$$= \min \begin{cases} 5+8 \\ 2+10 \end{cases} = \min \begin{cases} 13 \\ 12 \end{cases}$$

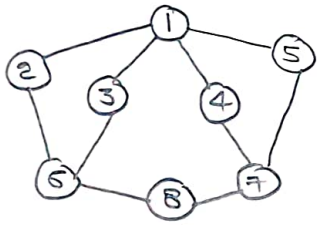
cost (1,1) = 12.

path : 1 → 3 → 5 → 8 → 9

2+5+2+3 = 12.

(OR)

4 a) In what order will the nodes be visited using BFS and DFS traversals for the following graph? Explain.



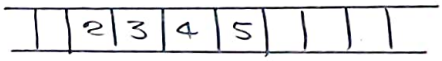

By using BFS:-

BFS follows the queue data structure ie FIFO approach.

The size of the queue is 8

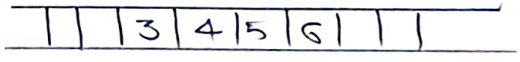
1. The selected starting vertex is 1 and that can be enqueue into the queue


2. The unvisited adjacency vertices for 1 are 2, 3, 4, 5 then 1 can be added to result field.



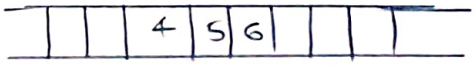

Result:


3. The next selected vertex is 2 and the unvisited adjacency vertex is only 6 and 2 can be added in result field.





4. The next selected vertex is 3 and unvisited adjacency vertex is only not there then simply remove 3 and add to result field.



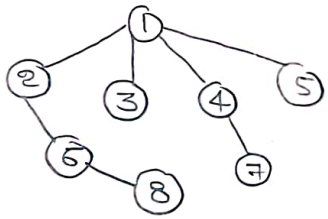

5. The next vertex is 4 and unvisited adjacency vertex is only 4 and 4 can be added to result field.


6. The next vertex is 5 and there are no unvisited adjacency vertices then simply remove 5 and add to result field.


7. The next vertex is 6 and the unvisited adjacency vertex is 8, & 6 can be add to result field.

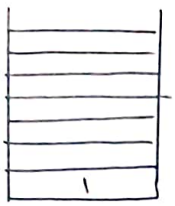

8. For both 7 and 8 there no unvisited adjacency vertices then simply remove them and they can be added to result field.


Spanning tree:

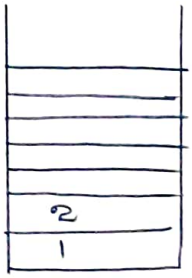


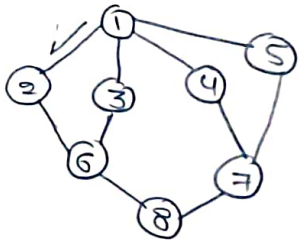

(ii) By using DFS approach:-

1. DFS follows the queue data stack datastructure ie LIFO approach.
2. The size of the stack is 8.
3. The selected starting vertex is 1. and that can be push into the stack.

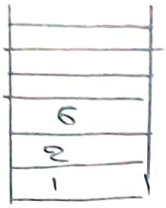


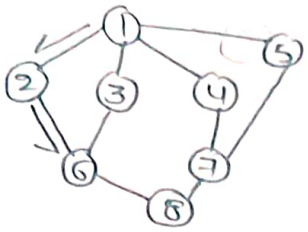

4. The unvisited adjacency vertex for 1 is e. & that can be push into the stack.



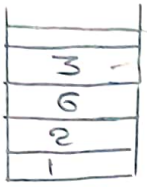



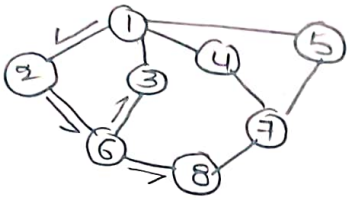
5) The unvisited adjacency vertex for 2 is 6 and that can be push into stack.



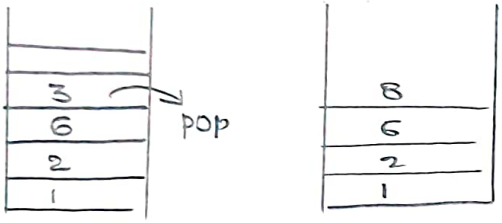



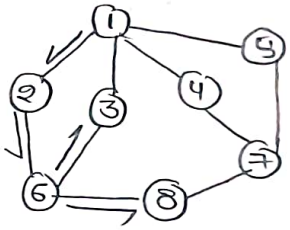

6. The unvisited adjacency vertex for 6 is 3 and that can be push into stack.



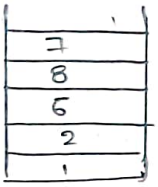



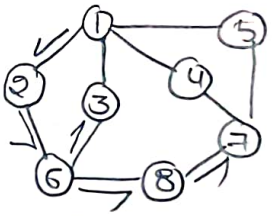

7. Now pop element 3 and another unvisited adjacency vertex for 6 is 8.



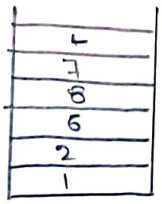



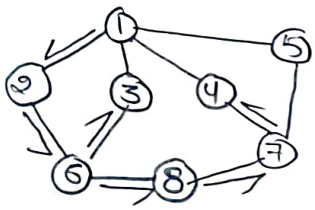

8. The unvisited adjacency vertex for 8 is 7 and that can be push into stack.



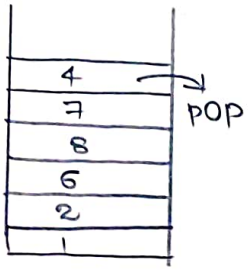



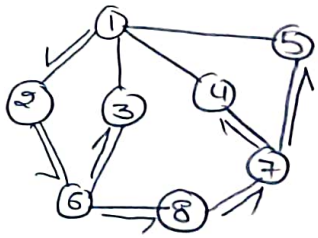

9. The unvisited adjacency vertex for 7 is 4 & that can be place into stack.



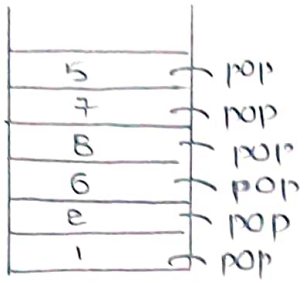



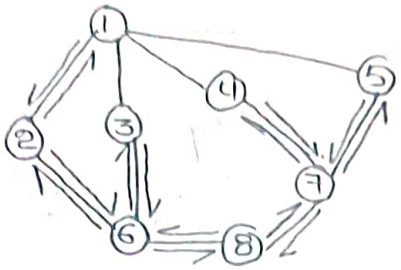

10. Now pop element 4 and another unvisited adjacency vertex for 7 is 5 and that can be place into stack.



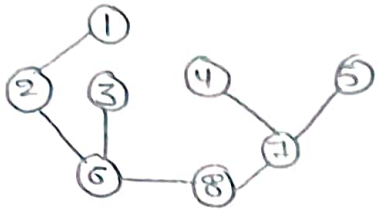



11. Now pop all the elements one by one.



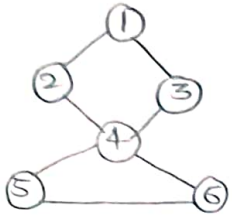

The spanning tree is



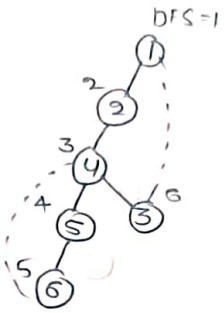

b) Differentiate Bi-connected components and strongly connected components with an appropriate example.

| Bi-connected components | strongly connected components. |
|---|---|
| 1. These are the components that we can remove any vertices in the given graph then that graph can produce different no. of bi-connected components. | These are nothing but each and every vertices of a graph can connected together. |
| 2. The deleted vertices position is called articulation point. | In directed graph only we can find out the strongly connected components. |
| 3. This can be solved either in the directed graph or undirected graph. | It follows KOSARAJU'S ALGORITHM. |
| 4. Algorithms like DFS are used to find connected components & articulation points. | It is a subgraph where every pair of vertices has a path in both directions. |

1. Find out the articulation point for the following graph.



DFS spanning tree for the above graph is




| Vertices | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| DFS time | 1 | 2 | 6 | 3 | 4 | 5 |
| least cost | 1 | 1 | 1 | 1 | 4 | 4 |

To find out the articulation point use condition $L[v] \geq D[u]$ if get's true then u is articulation point. v → child u → parent.

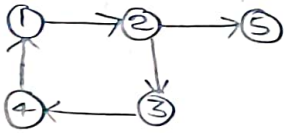
$L[4] \geq D[2]$ $L[5] \geq D[4]$

$1 \geq 2$ (x) $4 \geq 3$ (✓)

4 is articulation point.

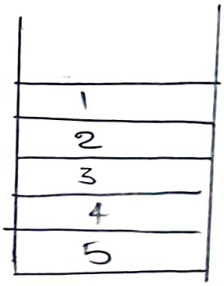
Bi-connected components are:

2. Solve the graph by strongly connected components.

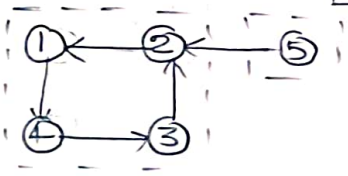


By using Kosaraju's alg:-

1. According to this alg. first we should sort all the edges in the given graph with corresponding DFS finishing time.



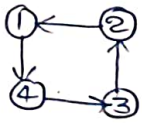

2. Now we should reverse the graph



3. Now pop the elements, pop element 1,

The connected edges for 1 are

1 to 4, 4 to 3, 3 to 2 then that graph can be taken as SCC-1



SCC-1

4. Next pop 5 and for 5 there are no connected edges then only 5 can be taken as SCC-2.

SCC-2

UNIT - IV

8 a) Draw and analyse the solution tree for the given sum of subsets prblm using Backtracking $S = \{3, 5, 6, 7\}$ and $M = 15$

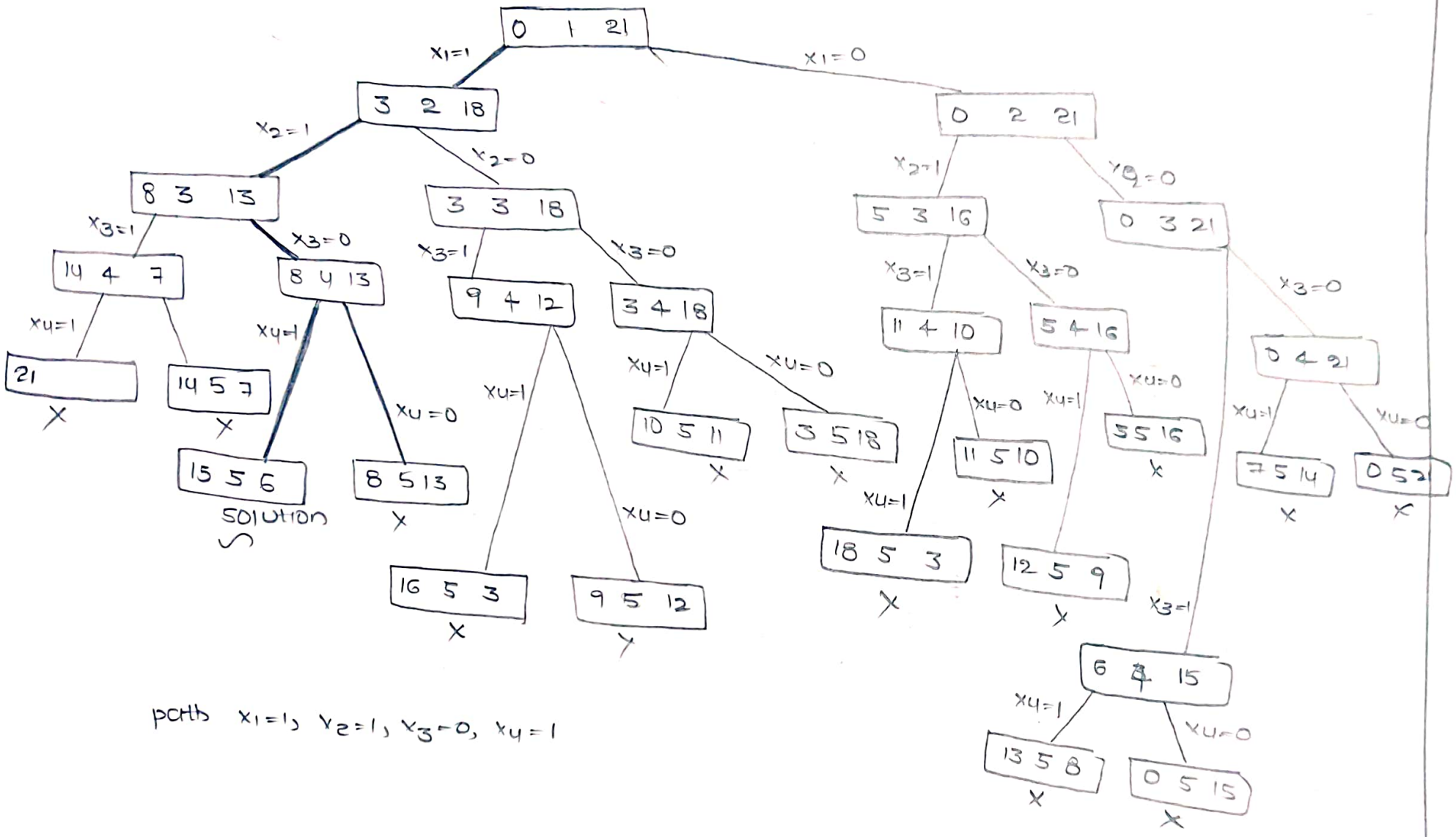
1. It is one of the application in backtracking approach.
2. Here we take the no of objects as 'n' and the given sum of the object value as 'm'.
3. The sum of objects should not be exceeded the given sum value.

Given $n = 4$ $S = \{3, 5, 6, 7\}$

$m = 15$

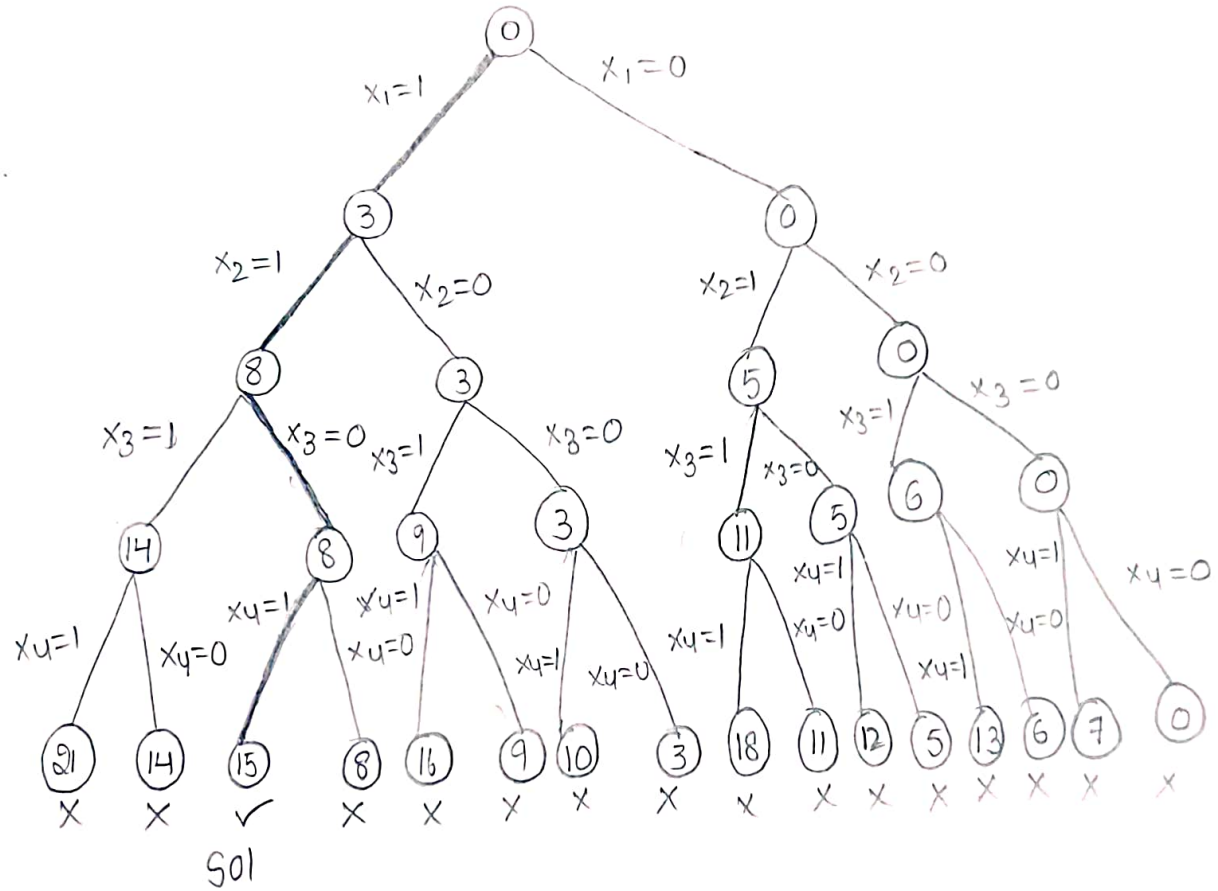
| subset | sum | solution |
|---|---|---|
| $\{-\}$ | 0 | feasible sol. |
| $\{3\}$ | $0+3=3$ | feasible sol. |
| $\{3,5\}$ | $3+5=8$ | feasible sol. |
| $\{3,5,6\}$ | $8+6=14$ | feasible sol. |
| $\{3,5,6,7\}$ | $14+7=21$ | sum exceeded 15 then go back to previous stage & select other path. |
| $\{3,5,7\}$ | $8+7=15$ | solution. |

3151671




path $x_1=1, x_2=1, x_3=0, x_4=1$

(OR)


0
x1=1
x1=0
3
0
x2=1
x2=0
x2=1
x2=0
8
3
5
0
x3=1
x3=0
x3=1
x3=0
x3=1
x3=0
x3=1
x3=0
14
8
9
3
11
5
6
0
xy=1
xy=0
xy=1
xy=0
xy=1
xy=0
xy=1
xy=0
xy=1
xy=0
xy=1
xy=0
xy=1
xy=0
21
14
15
8
16
9
10
3
18
11
12
5
13
6
7
0
X
X
✓
X
X
X
X
X
X
X
X
X
X
X
X
X
Sol

b.) Draw and analyse the solution tree for the 0/1 knapsack prblm using LC Branch and Bound method for the following data M=15, n=4, $(P_1, \ldots P_4) = (10,10,12,18)$, $(W_1, \ldots W_4) = (2,4,6,9)$.

First we should assign the negative sign to the profits

$(P_1 \ldots P_4) = (10,10,12,18)$

$(W_1 \ldots W_4) = (2,4,6,9)$.

$m = 15-2 = 13-4 = 9-6 = 3$

$\hat{u} = -10 + (-10) - 12 = -32$

$\hat{c} = -10 + (-10) - 12 + (3/9 \times (-18))$

$= -20 - 12 - 6$

$\hat{c} = -38$

case i,

$x_1 = 1$:

$m = 15-2 = 13-4 = 9-6 = 3$

$\hat{u} = -10 + (-10) - 12 = -32$

$\hat{c} = -10 + (-10) - 12 + (3/9 \times (-18))$

$\hat{c} = -38$

$x_1 = 0$:

$m = 15-4 = 11-6 = 5$

$\hat{u} = -10-12 = -22$

$\hat{c} = -10-12 - \frac{5}{9} \times (18)$

$\hat{c} = -32$

* should select $x_1 = 1$ path.

case ii,

$x_1 = 1, x_2 = 1$:

$m = 15-2 = 13-4 = 9-6 = 3$

$\hat{u} = -10 + (-10) + (-12) = -32$

$\hat{c} = -10 + (-10) + (-12) + (3/9 \times -18)$

$\hat{c} = -38$

$x_1 = 1, x_2 = 0$;

$m = 15-2 = 15-6 = 7$

$\hat{u} = -10-12 = -22$

$\hat{c} = -10-12 + (7/9 \times -18)$

$\hat{c} = -36$.

* should select $x_1 = 1$ and $x_2 = 1$ path.

case iii,

$x_1 = 1, x_2 = 1, x_3 = 1$

$m = 15-2 = 13-4 = 9-6 = 3$

$\hat{u} = -10 + (-10) + (-12) = -32$

$\hat{c} = -10 + (-10) + (-12) + (3/9 \times -18)$

$\hat{c} = -38$

$x_1 = 1, x_2 = 1, x_3 = 0$.

$m = 15-2 = 13-4 = 9-9 = 0$

$\hat{u} = -10 + (-10) + (-18) = -38$

$\hat{c} = -10 + (-10) + (-18) = -38$.

* should select $x_3 = 0$ path.

case iv

$x_1=1,\ x_2=1,\ x_3=0,\ x_4=1$

$m=15-2=13-4=9-9=0$

$\hat{u}=-10+(-10)+(-18)=-38$

$\hat{c}=-10+(-10)+(-18)=-38$

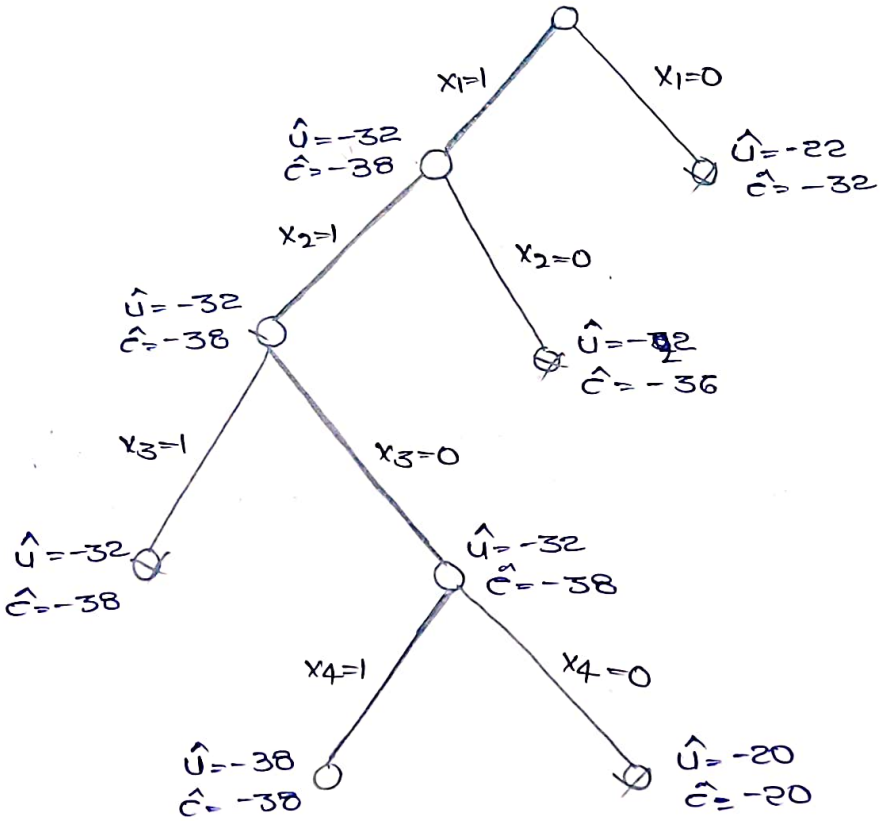
* Should select $x_4=1$ path.

$x_1=1,\ x_2=1,\ x_3=0,\ x_4=0$

$m=15-2=13-4=9$

$\hat{u}=-10+(-10)=-20$

$\hat{c}=-10+(-10)=-20.$




| $x_1=1,\ x_2=1,\ x_3=0,\ x_4=1$ |
|---|

Weight = 2+4+9

Weight = 15

(OR)

9 a.) Explain the basic concepts of P, NP, NP complete, and NP-Hard prblms with examples.

1. Polynomial time: The algorithm (or) prblm that can be executed within a specified time.

eg: 1. Searching techniques → linear search, Binary Search.
2. Sorting techniques → Quick sort, Merge sort.

2. Exponential time: The algorithm (or) prblm that can be executed in more than the specified time.

eg: 1. 0/1 knapsack prblm.
2. N-queens prblm.

P-class:-

1. It is also called as Deterministic Algorithms.
2. These are decision prblms that can be solved by a deterministic algorithms.
3. This algorithms can be executed within a polynomial time.

ex: Searching in a sorted array.

NP-class:

1. It is also known as Non-deterministic algorithms.
2. These algorithms can be executed in non-polynomial time means it takes more time than the specified time.


"P ⊆ NP"

3. sometimes NP prblms can be executed within a polynomial time then the polynomial time (p) will be increased and vice-versa.

NP-complete problems:-

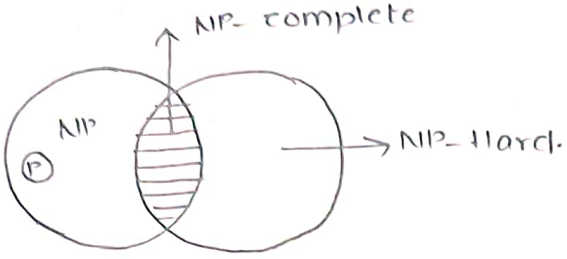
1. Whenever the given algorithms can be executed within a polynomial time then that algorithms comes under NP-complete.
2. These are the "hardest" problems within NP.
3. A prblm is NP-complete if it is in NP, and every other prblm in NP can be reduced to it in polynomial time. ex: Travelling sales person

NP-Hard problems:-

Whenever the given algorithm doesn't execute within a polynomial time then that prblms comes under NP-Hard.

2. These prblms are atleast as hard as any NP-complete prblm.

3. An NP-Hard prblm does not necessarily have to be in NP itself.




⇒ Sometimes NP-Hard prblms can be executed within a polynomial time then NP-Hard size will be decreased.

b) State and explain Cook's Theorem.

Cook's theorem :-

1. The scientist "Stephen Cook" in 1971 stated that boolean satisfibility prblm is in NP-complete prblm.

2. Cook's theorem states that the satisfibility (Sat) is in P, if and only if P = NP.

Proof: Show that satisfibility prblm is in NP-complete.

1. Cook's states that satisfibility in P if P = NP.

2. Sat is in NP because a Non-deterministic Alg. can guess an Assignment truth values of variables, an expression is satisfible if it value result is true.

eg: CNF (Conjugate Normal Form)

$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$

If $x_1 = 1$, $x_2 = 0$ and $x_3 = 0$.

3. Hence we can prove satisfibility is in NP-complete because Sat ∈ NP.

certain assumptions considered in Cook's theorem are:

1. Execution of Alg. 'A' is done on a word oriented system where length of each word is 'w bits'.

2. All variables in A are defined using either integer (or) boolean.
3. Input provided to 'A' is only through the arguments. (const).
4. Additional statements like success(), failure(), in the Non-deterministic algorithm.

P. Bhanu Chand (AIML)

V.

K. Hari Deep