Page No.
Date

★ Divide and Conquer

Quick sort

Partition (A, p, r)
x = A[r] → Pivot element
i = p-1
for j = p to r-1
  if A[j] ≤ x
    i = i+1
    exchange A[i] with A[j]
exchange A[i+1] with A[r]
return i+1

T.C
T(n) = T(n-q) + T(q) + Θ(n)
T(n) = T(n-1) + c.(n)
T(n-1) = T(n-2) + c(n-1)
|

t case T.C = Θ(n²) → If array is sorted
ge case = Θ(n log n)

W.C = T(n) = T(n-1) + n = Θ(n²) through substitution
B.C = T(n) = (T(n/2)*(n/2) + n)

Page No.
Date

BinarySearch (A[], x, first, last) {

mid = first + (last - first)/2
or mid = (first + last)/2

if x == A[mid]
    return mid
else if x <= A[mid]

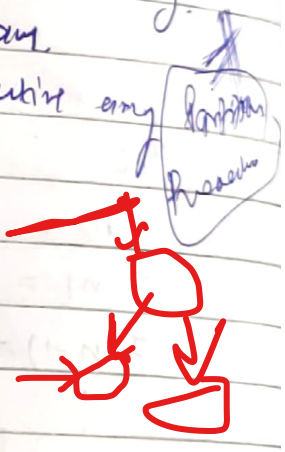
Median Finding

First sort the array ↳ O(n log n)

Median of Medians

5.5 → 5 (lower median)
↳ median → constant time

1. Divide entire array into T.C = O(n log n)
   sub array of size 5 elements.
2. Now compute the median of each sub array.
3. Now compute median of medians.
4. It would act as Pivot for the entire array
5. Quick sort if Pivot == Median[element]



Binary Tree

Tree → A Tree is a hierarchical collection of nodes with one of the node at the top of the hierarchy which is designated as a root node.

- Each node can have atmost one link coming into it. The node where link originates is called its parent.
- Root has no parent
- The links leaving a node point to its child node.
- Trees are recursive structure each child is itself

the root of a subtree.
Bottom have leaf nodes with no children

Degree of node
The no. of subtrees of a node is called its degree.

leaf node / terminal nodes will have degree zero.

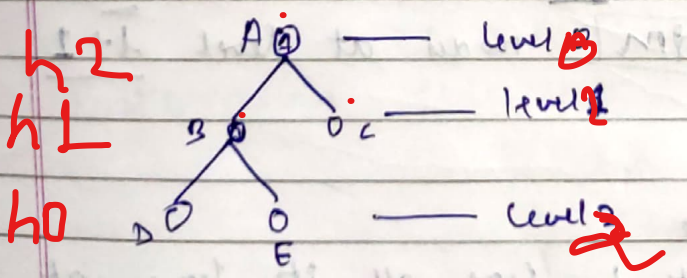
The degree of a tree is the maximum of the degree of the nodes in the tree.

Binary Tree

- In a BT each node can have atmost 2 children.
- Degree of BT = 2

Why use BT

1. It's implementation is easier. (Array / Linked list)
2. Any tree can be rep. using BT.



Maximum level = Height of Tree

Page No.
Date

A tree with n nodes has exactly n-1 edges

In a tree every node except the root node has exactly 1 parent. ~~except root node~~

There is exactly 1 path connecting any 2 nodes in a tree.

Maximum no. of nodes in a Binary Tree of height k =

$$\begin{cases} 2^{k+1} - 1 & \text{if } k \geq 0 \\ 2^k - 1 & \text{if } k \geq 1 \end{cases}$$

Max. no of nodes on level i of a B.T is

$$\begin{cases} 2^i & \text{if } i \geq 0 \\ 2^{i-1} & \text{if } i \geq 1 \end{cases}$$

For a full Binary Tree no of nodes on level i

$$2^{k+1} - 1 \qquad k \geq 0$$

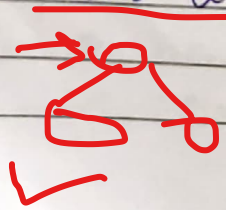
If h is the height of B.T then max no. of leaves

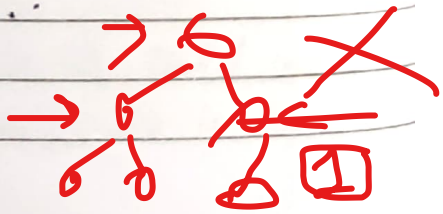
$$\begin{cases} 2^{h+1} - 1 & h \geq 0 \\ 2^h - 1 & h \geq 1 \end{cases}$$

If a B.T contains m nodes at level l then it contains $2m$ nodes at level $l+1$

Full Binary Tree / Strict

A FBT of height h has all its leaves at level h. In other words all non-leaf nodes of a full B.T will have 2 children.

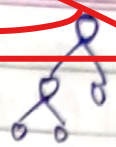




Page No.
Date

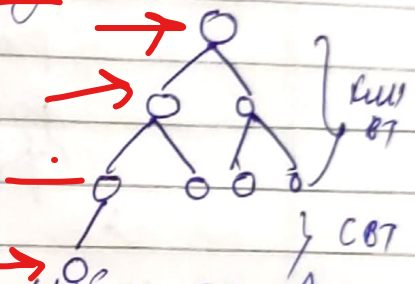
Strict Binary Tree

* Either 0 or 2 child

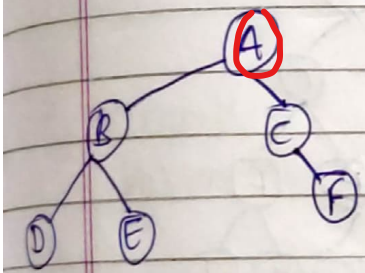


Complete Binary Tree

* A BT with ~~no. nodes~~ height h is said to be complete if it a full BT till level h-1 and level h is filled from left to right

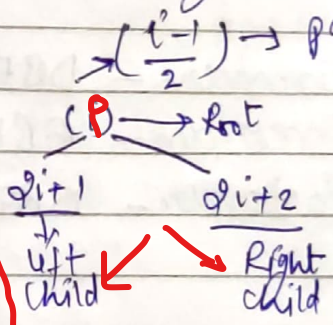


Implementation of Binary Tree using an Array



Array =

0	1	2	3	4	5	6
A	B	C	D	E	·	F



Inorder – Left root right

D B E A C F

Preorder – Root left Right

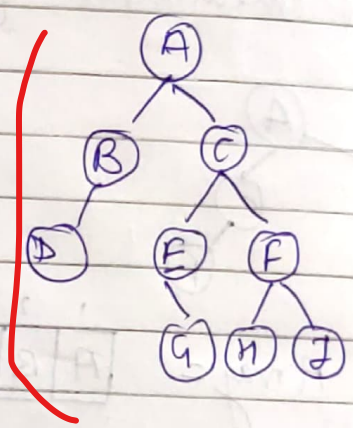
A B D E C F

Postorder – Left right root

D E B F C A

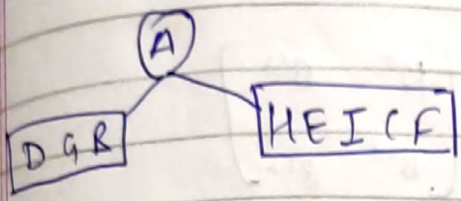
```
void Preorder (bNode *T)
{  printf ("%c", T->data);
   preorder(T->left);
   preorder(T->right);
}
   return;
void Postorder(bNode *T){
        postorder(T->left);
        postorder(T->right);
        printf("%c", T->data);
}
return;
```

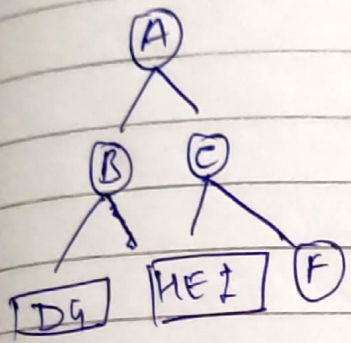
Inorder - DBEGCHFI
Preorder - ABDCEGFHI
Postorder - DBGEHIFCA

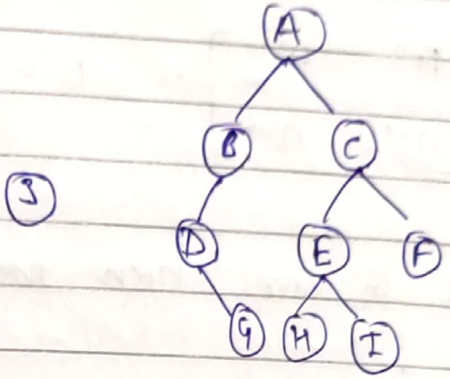


Page No.
Date

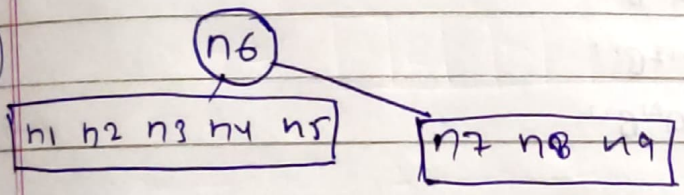
* Postorder A B D G C E H I F
  Inorder D G B A H E I C F
  (Left) (Right)

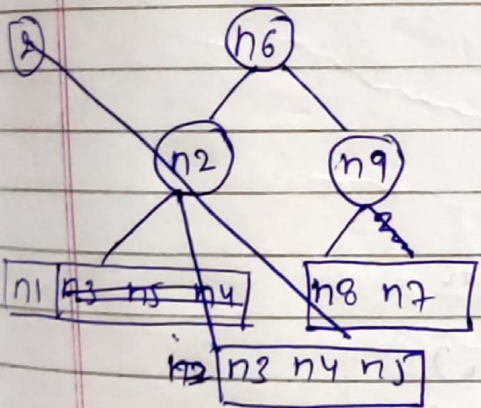
1. 

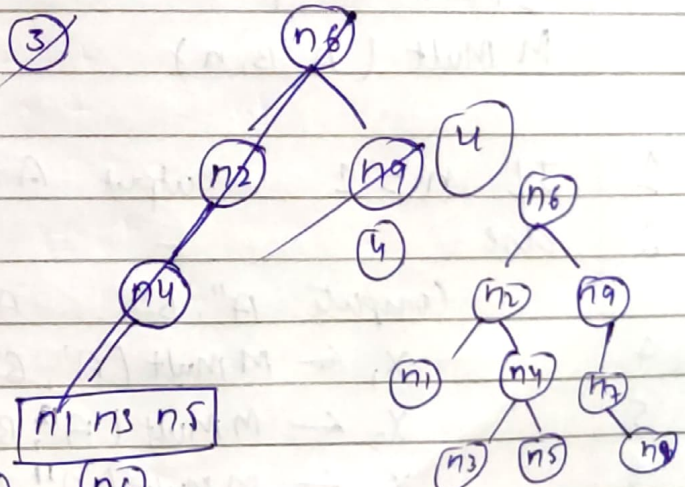
2. 

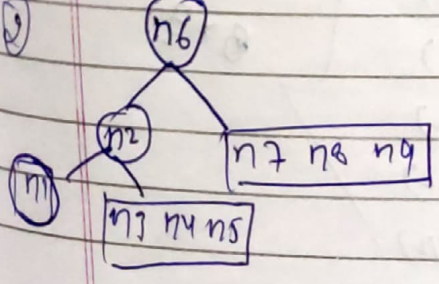
3. 

* Inorder n1 n2 n3 n4 n5 (n6) n7 n8 n9
  (Left) (Right)
  Postorder n1 n3 n5 n4 [n2] n8 n7 n9 [n6]
  II I

1. 

2. 

3. 

4. 

Strassen's

Page No.
Date

Matrix Multiplication :- $O(n^3)$

(1) For loops are used

Assumption :- Order = $n \times n$ and divisible by 2

$$A = \begin{bmatrix} A^{11} & A^{12} \\ A^{21} & A^{22} \end{bmatrix} \qquad B = \begin{bmatrix} B^{11} & B^{12} \\ B^{21} & B^{22} \end{bmatrix}$$

- A and B are $n \times n$ matrices.

A^{ij} & B^{ij} are $\frac{n}{2} \times \frac{n}{2}$ matrices

$$C = A \times B \begin{bmatrix} C^{11} & C^{12} \\ C^{21} & C^{22} \end{bmatrix}$$

$$C^{11} = A^{11}B^{11} + A^{12}B^{21}$$
$$C^{12} = A^{11}B^{12} + A^{12}B^{22}$$
$$C^{21} = A^{21}B^{11} + A^{22}B^{21}$$
$$C^{22} = A^{21}B^{12} + A^{21}B^{22}$$

Algo. => Matrix Multiplication

```
M Mult (A, B, n)

1  If n = 1   output A x B
2  else
       Compute A'', B'', ... A^21, B^22
4      X1 ← M Mult (A'', B'', n/2)
5      X2 ← M Mult (A^12, B^21, n/2)
6      X3 ← M Mult (A^11, B^12, n/2)
7      X4 ← M Mult (A^12, B^22, n/2)
8      X5 ← M Mult (A^21, B'', n/2)
9      X6 ← M Mult (A^22, B^21, n/2)
10     X7 ← M Mult (A^21, B^12, n/2)
11     X8 ← M Mult (A^21, B^22, n/2)
```

8 x $\frac{n}{2}$

$C^{11} \leftarrow X_1 + X_2$

$C^{12} \leftarrow X_3 + X_4$

$C^{21} \leftarrow X_5 + X_6$

$C^{22} \leftarrow X_7 + X_8$

output C

Matrix addition & loops are used

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

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

Strassen's Matrix Multiplication Algorithm

$P_1 = A^{11}(B^{12} - B^{22})$

$P_2 = (A^{11} + A^{12})B^{22}$

$P_3 = (A^{21} + A^{22})B^{11}$

$P_4 = A^{22}(B^{21} - B^{11})$

$P_5 = (A^{11} + A^{22})(B^{11} + B^{22})$

$P_6 = (A^{12} - A^{22})(B^{21} + B^{22})$

$P_7 = (A^{11} - A^{21})\cdot(B^{11} + B^{12})$

$C^{11} = P_5 + P_4 - P_2 + P_6$

$C^{12} = P_1 + P_2$

$C^{21} = P_3 + P_4$

$C^{22} = P_1 + P_5 - P_3 - P_7$

Algo.

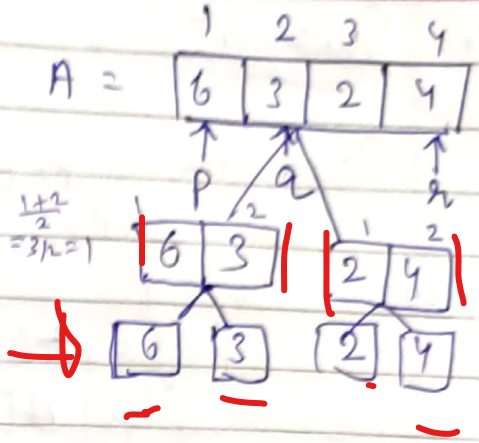
```
Strassen(A, B)
1.  If n = 1   O/P A x B
2.  Else
3.  Compute A^11, B^11, ... B^22
4   P1 ← Strassen(A^11, B^12 − B^22)
5   P2 ← Strassen(A^11 + A^12, B^22)
6   P3 ← Strassen(A^21 + A^22, B^11)
7   P4 ← Strassen(A^22, B^21 − B^11)
8   P5 ← Strassen(A^11 + A^22, B^11 + B^22)
9   P6 ← Strassen(A^12 − A^22, B^21 + B^22)
10  P7 ← Strassen(A^11 − A^21, B^11 + B^12)
1
11  C^11 ← P5 + P4 − P2 + P6
12  C^12 ← P1 + P2
13  C^21 ← P3 + P4
14  C^22 ← P1 + P5 − P3 − P7
15  O/P C
```

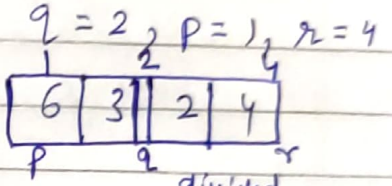
Divide and Conquer

Page No. _____
Date | $q = \frac{1+4}{2} = \frac{5}{2} = 2$

1. Merge Sort

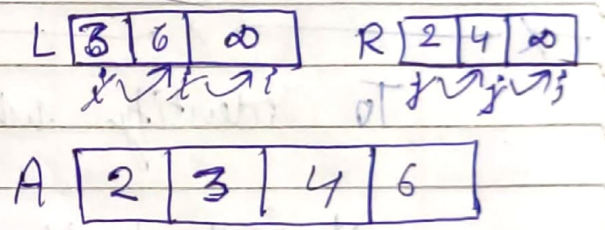
```
Merge-SORT (A, p, r)
1  If p < r
2     q = ⌊(p+r)/2⌋
3     MERGE-SORT (A, p, q)
4     MERGE-SORT (A, q+1, r)
5     MERGE (A, p, q, r)
```





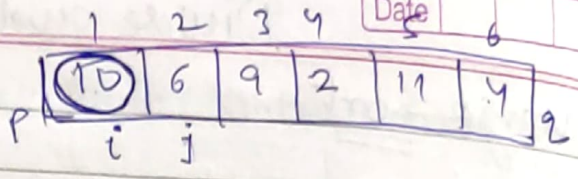
2 - 1 + 1 = 2 } size of divided array
4 - 2 = 2

```
MERGE (A, p, q, r)
1  n1 ← q - p + 1
2  n2 ← r - q
3  Create array L[1 ... n1+1] and R[1 ... n2+1]
4  For i ← 1 to n1
5     do L[i] ← A[p+i-1]
6  for j ← 1 to n2
7     do R[j] ← A[q+j]
8  L[n1+1] ← ∞
9  R[n2+1] ← ∞
10 i ← 1
11 j ← 1
12 for K ← p to r
13    do if L[i] ≤ R[j]
14       than A[K] ← L[i]
15            i++
16       else A[K] ← R[j]
17            j++
```



Page No.
Date

2. Quicksort




```
Partition (A, p, q)
    x = A[p]          ← Pivot
    i := p
    for (j = p+1; j ≤ q; j++)
        if (A[j] ≤ x)
            i += 1
            swap(A[i], A[j])
    swap(A[j], A[p])
```

To identify which items to include in the Knapsack

```
Let i = n, K = W
while i, K > 0
    if (V[i,K] ≠ V[i-1, K])
        Then do  i = i-1, K = K - Wi
        Mark the item i
    Else
        i = i - 1
```

0/1 Knapsack

Page No. _____
Date _____

Dynamic Programming (brute force) → choose / don't choose

1. Optimal substructure
2. Overlapping sub problems.

Given Problem: Weights {3, 4, 6, 5} W = 8 → size of knapsack
Profits {2, 3, 1, 4} selected item x_i = {1, 0, 0, 0} n = 4 → no. of items/objects
$2^n = 2^4 = 16$

Sol. to this problem using Dynamic approach. Sol. vector → {1, 0, 0, 1}

weight is arrange in asc order

Profit → pi, n → object, wi ← weight

pi	wi	i \ W	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2 ←	(2)	2	2	2	2
3	4	2	0	0	0	2	(3)	3	3	5	5 ←
4	5	3	0	→0	0	2	3	4	4	→5	6 ←
1	6	4	0	0	0	2	3	4	4	5	(6) ← max. profit

weight ↓

- First row & col = 0 bcoz it doesn't have any item.

4 → max(3+0, 2), max(3+0, 2), max(3+0, 2), max(3+2, 2), max(3+2, 2)
5 → max(4+0, 3), max(4+0, 5), max(4+2, 5)
6 → max(1+0, 4), max(1+0, 5), max(1+0, 6)

matrix x_i = {1, 0, 0, 1}

formula → $m[i, w] = \max(m[i-1, w], m[i-1, w - w[i] + P[i]])$

copy the above row; greater than equal to

- if W < Wi (knapsack weight is < item's weight) then fill 0 in the cell.
- if W > Wi then fill its profit in the cell. else copy the above cell value down.

Ex:- $m(4,7) = \max(m[3,7], m[3, 7-6] + 1)$
$\max(5, 0+1)$
max = (5)

3 → i-1
7 → W
6 → Wi
1 → Pi

Page No.
Date

0/1 Knapsack Problem

In a Knapsack problem, we are given a set of n items where each item i is specified by a size w_i and a profit p_i. And knapsack size is W.

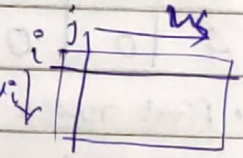
There are 2 versions of this problem.

- (0-1) Knapsack Problem : Items are indivisible. You either take an item or not. Solved with dynamic Programm.
- Fractional Knapsack Problem : Items are divisible. You can take any fraction of an item. Solved with a greedy algo.

Recurrence Relation

$$m[i,w] = \begin{cases} \max(m[i-1,w], m[i-1, w-w[i]] + p_i) & \text{if } w_i \le W \\ m[i-1,w] & \text{if } w_i < W \end{cases}$$

Algorithm



obj. size, Profit, KS weight

```
KNAPSACK (w_i, p_i, W)
 for i ← 0 to n do
    for j ← 0 to W do
       if i=0 or j=0 then }  for 1st row & 1st col. where W=0
          P[i,j] = 0
       else
          if j < w_i then →  (j → knapsack)
             P[i,j] = P[i-1, w_i]   (if w_i < W then copy the upper cell do item)
          else
             ~~P[i,j] = max[i-1, W]~~
             P[i,j] = max(P[i-1,j], P[i-1, j-w_i] + p_i)
                                   (subs the obj. weight from knapsack size.)
          end if
       end if
    end for
 end for
 return P[n, W]
```

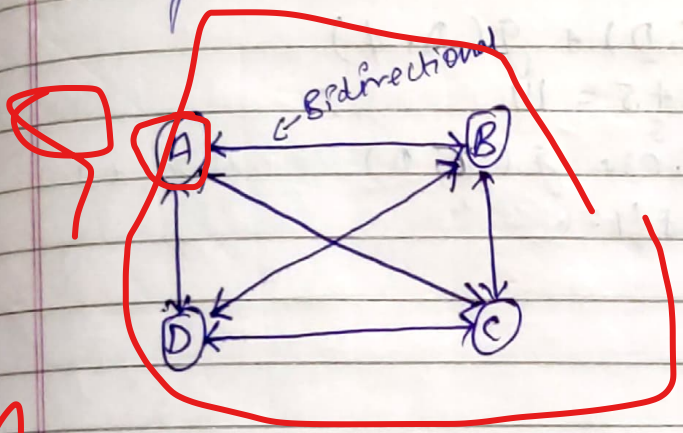
copy

profit

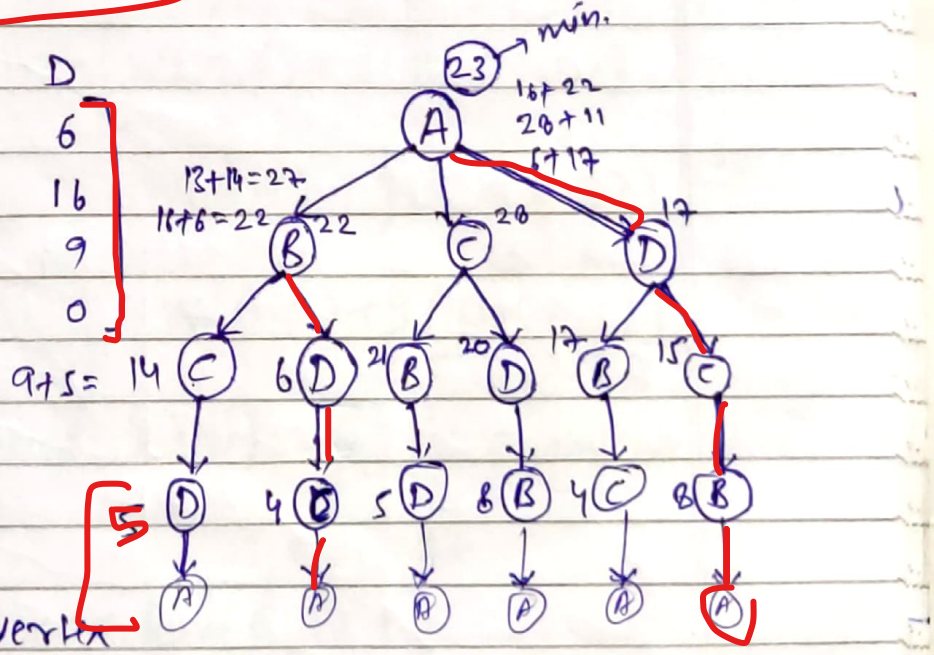
Travelling Salesman Problem

In travelling salesman Problem
1. The salesman must visit 'n' given cities.
2. Each city should be visited exactly once.
3. He should finish at the city he starts from.
4. The total cost to make round is minimum.
5. So, we can say that for a TSP we are given a complete weited graph. and we want to find a cycle through all the vertices of minimum weight.

Ex:- A → B → C → D → A or
A → D → C → B → A ...



	A	B	C	D
A	0	16	11	6
B	8	0	13	16
C	4	7	0	9
D	5	12	2	0



i = starting vertex
s = set of vertices
salesman would visit once (exception)

$$g(i,s) = \min_{j \in s} [w(i,j) + g(j, s-j)]$$

$$g(i,s) = w(i,n) \quad \text{if } s = \phi \qquad g(n,\phi) = w(n,i)$$

$g(B, \{C,D\}) = \min[w(B,C) + g(C,\{D\})$

$S.\{C,D\} - \{C\}$

$S = \{D\}$

$S = \{D\} - \{D\}$

$S = \phi$

$g(C,\{D\}) = \min w(C,D) + g(D,\phi)$

$= x + w(D,C)$

$= x + w(D,A)$

$= x + y = z$

$g(A, \{B,C,D\})$

$= \min[w(A,B) + g(B,\{C,D\});$
$w(A,C) + g(C,\{B,D\});$
$w(A,D) + g(D,\{B,C\});]$

$g(B,\{C,D\})$

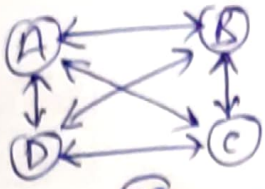
$= \min[w(B,C) + g(C,D);$
$w(B,D) + g(D,C)]$

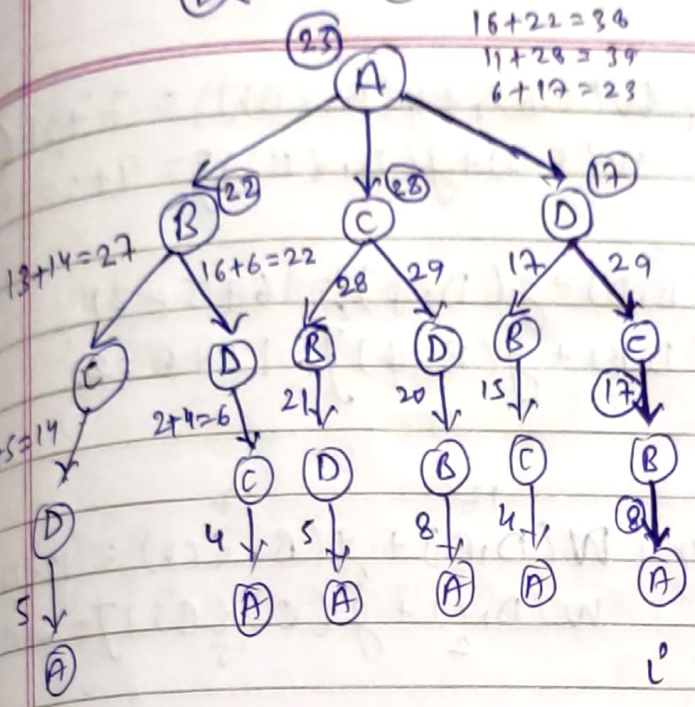
$g(C,D) = w(C,D) + g(D,\phi)$

$= 9 + 5 = 14$

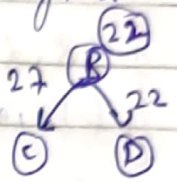
$g(D,C) = w(D,C) + g(C,\phi)$

$= 2 + 4 = 6$





* If you have 2 choices then consider the one that gives min. value.



i = Starting vertex and
S = Set of vertices that salesman visit exactly once except the starting or root vertex.

formula

$$g(i,S) = \min\left[W(i,j) + g(j, \{S-j\})\right]$$

Sol. 1

$$g(A,\{B,C,D\}) = \min\left[W(A,B) + g(B,\{C,D\}),\ W(A,C) + g(C,\{B,D\}),\ W(A,D) + g(D,\{B,C\})\right]$$

(A, {B,C,D}: A = i, {B,C,D} = set of vertices; B = j)

$16+22=38$; $11+28=39$; $6+17=23$

$(S-j)$: $\{B,C,D\}-\{B\} = \{C,D\}$

This is recursive call

The fun^n is called recursively so again we have to solve it.

(1) $g(B,\{C,D\}) = \min\left[W(B,C) + g(C,\{D\}),\ W(B,D) + g(D,\{C\})\right]$

$13+14=27$; $16+6=22$ (min.)

$i=B$, $S=\{C,D\}$, $j=C/D$

$i=C$, $S=\{D\}$, $j=D$ → $g(C,\{D\}) = \min\left[W(C,D) + g(D,\phi)\right]$

$i=D$, $S=\{C\}$, $j=C$ → $g(D,\{C\}) = W(D,C) + g(C,\phi)$

$9+5=14$; $2+4=6$

$g(n,\phi) = g(n,i)$, $n = C, D$

i = A

The salesman is going back to his city.

Page No.
Date

(2) $g(C, \{B, D\}) = \min[\, w(C,B) + g(B, \{D\}), = 7 + 21 = 28$
$w(C,D) + g(D, \{B\})] = 9 + 20 = 29$

$g(B, \{D\}) = \min[\, w(B,D) + g(D, \phi), = 16 + 5 = 21$
$g(D, \{B\}) = w(D,B) + g(B, \phi)] = 12 + 8 = 20$

(3) $g(D, \{B, C\}) = \min[\, w(D,B) + g(B, \{C\}), = 12 + 17 = 29$
$w(D,C) + g(C, \{B\})] = 2 + 15 = 17$

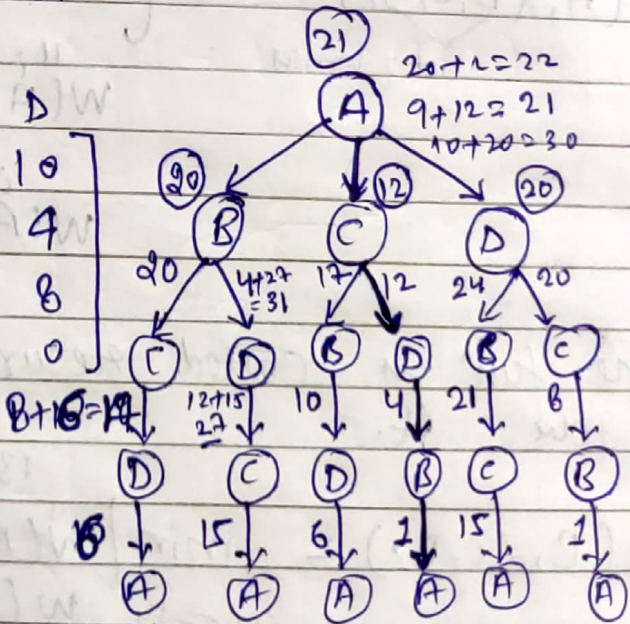
$g(B, \{C\}) = w(B,C) + g(C, \phi) = 13 + 4 = 17$
$g(C, \{B\}) = w(C,B) + g(B, \phi) = 7 + 8 = 15$

$A \to D \to C \to B \rightarrow$ 23

Time complexity = d^n

Ques. =

$$C = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 2 & 9 & 10 \\ 1 & 0 & 6 & 4 \\ 15 & 7 & 0 & 8 \\ 6 & 3 & 12 & 0 \end{bmatrix} \end{array}$$



(1) $g(A, \{B, C, D\}) = \min[\, w(A,B) + g(B, \{C, D\}) = 2 + 30 = 32$
$w(A,C) + g(C, \{B, D\}) = 9 + 12 = 21$
$w(A,D) + g(D, \{C, B\})] = 10 + 20 = 30$

$g(B, \{C, D\}) = \min\{ W(B,C) + g(C, \{D\}), \Rightarrow 6+14 = 20$
$W(B,D) + g(D, \{C\})\} \Rightarrow 4+27 = 31$

$g(C, \{D\}) = W(C,D) + g(D, \phi)$ (8 + 6 = 14)
$g(D, \{C\}) = W(D,C) + g(C, \phi)$
$12 + 15 = 27$

② $g(C, \{B, D\}) = \min\{ W(C,B) + g(B, \{D\}), \Rightarrow 7+10 = 17$
$W(C,D) + g(D, \{B\})\} \Rightarrow 8+4 = 12$

$g(B, \{D\}) = W(B,D) + g(D, \phi) \Rightarrow 4+6 = 10$
$g(D, \{B\}) = W(D,B) + g(B, \phi) \Rightarrow 3+1 = 4$

③ $g(D, \{C, B\}) = \min\{ W(D,C) + g(C, \{B\}), \Rightarrow 12+8 = 20$
$W(D,B) + g(B, \{C\})\} \Rightarrow 3+21 = 24$

$g(C, \{B\}) = W(C,B) + g(B, \phi) \Rightarrow 7+1 = 8$
$g(B, \{C\}) = W(B,C) + g(C, \phi) \Rightarrow 6+15 = 21$

A → C → B → B = 21

Page No.
Date

11

Binomial coefficient

$$^{n}C_{k} = \begin{cases} 1 & \text{if } k=0, \text{ or } k=n \\ {}^{n-1}C_{k-1} + {}^{n-1}C_{k} & \text{if } n>k>0 \end{cases}$$

or

$$C(n,k) = \begin{cases} 1 & \\ C(n-1, k-1) + C(n-1, k) & \end{cases}$$

```
for i = 0 to n
    for j = 0 to k
        if j == 0 or j == i
            c[i, j] = 1
        else
            c[i, j] = c[i-1, j-1] + c[i-1, j]
return c[n, k]
```

T.C = O(n * k)

Ex:- $^{5}C_{3}$

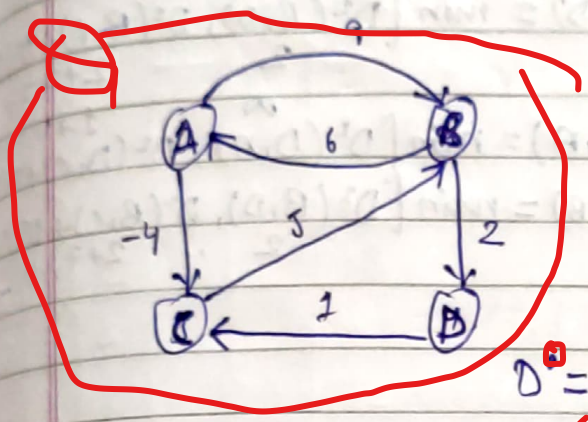
n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

The shortest path for each pair of vertices

All Pair Shortest Path Problem

Floyd Warshall Algo.

↳ Work for both +ve and -ve weighted edges.



$D^0 =$

	A	B	C	D
A	0	9	-4	∞
B	6	0	∞	2
C	∞	5	0	∞
D	∞	∞	1	0

① Via A

$D^1 =$

	A	B	C	D
A	0	9	-4	∞
B	6	0	2	2
C	∞	5	0	∞
D	∞	∞	1	0

$D^1(B,C) = \min[D^0(B,C), D^0(B,A) + D^0(A,C)]$

$= \infty, 6 + (-4) = 2$

$D^1(B,D) = \min[D^0(B,D), D^0(B,A) + D^0(A,D)]$

$2, 6 + \infty$

$D^1(C,B) = \min[D^0(C,B), D^0(C,A) + D^0(A,B)]$

$5, \infty + 9$

$D^1(C,D) = \min[D^0(C,D), D^0(C,A) + D^0(A,D)]$

$\infty, \infty + \infty$

② Via B

$D^2 =$

	A	B	C	D
A	0	9	-4	11
B	6	0	2	2
C	11	5	0	7
D	∞	∞	1	0

$D^2(A,C) = \min[D^1(A,C), D^1(A,B) + D^1(B,C)]$

$= -4, 9 + \infty$

$D^2(A,D) = \min[D^1(A,D), D^1(A,B) + D^1(B,D)]$

$= \infty, 9 + 2 = 11$

$D^2(C,A) = \min[D^1(C,A), D^1(C,B) + D^1(B,A)]$

$= \infty, 5 + 6 = 11$

$D^2(C,D) = \min[D^1(C,D), D^1(C,B) + D^1(B,D)]$

$\infty, 5 + 2 = 7$

$D^2(D,A) = \min[D^1(D,A), D^1(D,B) + D^1(B,A)]$

$= \infty, \infty + 6$

$D^3(A,B) = \min[D^2(A,B), D^2(A,C)+D^2(C,B)]$
$\quad = \min[11, -4+5] = 1$

(3) Via - C

$D^3(A,D) = \min[D^2(A,D), D^2(A,C)+D^2(C,D)]$
$\quad = \min[3, -4+7] = 3$

$D^3(B,A) = \min[D^2(B,A), D^2(B,C)+D^2(C,A)]$
$\quad = \min[6, 2+11] = 13$

$D^3(B,D) = \min[D^2(B,D), D^2(B,C)+D^2(C,D)]$
$\quad = \min[2, 2+7] = 9$

$D^3(D,A) = \min[D^2(D,A), D^2(D,C)+D^2(C,A)]$
$\quad = \min[\infty, 1+11] = 12$

$D^3(D,B) = \min[D^2(B,D), D^2(B,C)+D^2(C,D)]$
$\quad = \min[2, 2+7] = 9$

$D^3 =$

	A	B	C	D
A	0	1	-4	3
B	6	0	2	2
C	11	5	0	7
D	12	2	1	0

(4) Via - D

$D^4(A,B) = \min[D^3(A,B), D^3(A,D)+D^3(D,B)]$
$\quad = \min[1, 3+2] = 1$

$D^4(A,C) = \min[D^3(A,C), D^3(A,D)+D^3(D,C)]$
$\quad = \min[-4, 3+1] = -4$

$D^4(B,A) = \min[D^3(B,A), D^3(B,D)+D^3(D,A)]$
$\quad = \min[6, 2+12] = 6$

$D^4(B,C) = \min[D^3(B,C), D^3(B,D)+D^3(D,C)]$
$\quad = \min[2, 2+1] = 2$

$D^4(C,A) = \min[D^3(C,A), D^3(C,D)+D^3(D,A)]$
$\quad = \min[11, 7+12] = 11$

$D^4(C,B) = \min[D^3(C,B), D^3(C,D)+D^3(D,B)]$
$\quad = \min[5, 7+2] = 5$

$D^4 =$

	A	B	C	D
A	0	1	-4	3
B	6	0	2	2
C	11	5	0	7
D	12	2	1	0

$d^K(i,j) = \min\{ d^{K-1}(i,K), d^{K-1}(K,j) \}$.

5×4 4×3 5×3×4 = 60

Page No.
Date

Dynamic Programming

Try all the possibilities and pick up the best one

Matrix chain Multiplication

* cost of single matrix = 0
* $A_{m \times n} \times B_{n \times p} = AB_{m \times p \times n}$

Ex :- A_1 A_2 A_3
3×5 5×2 2×3

1. $(A_1 \cdot A_2) \cdot A_3 = 3 \times 5 \times 2 + 0 + 3 \times 2 \times 3 = 30 + 18 = 48$
   $B_{3 \times 2}$ (2×3) ← min. cost

2. $A_1 \cdot (A_2 \cdot A_3) = 0 + 5 \times 2 \times 3 + 3 \times 5 \times 3 = 30 + 45 = 78$
   (3×5) (5×3)

matrix → $A_1 \rightarrow A_n$
order $P_0 \rightarrow P_n$

$p_i \times p_j \; p_j$

$p_1 p_2 \; p_2 p_3$
$p_3 p_4 \; p_4 p_5$
p_i

Ex :- A_1 A_2 A_3 A_4
5×4 4×6 6×2 2×7

p_0 p_1 p_2 p_3 p_4
d_0 d_1 d_2 d_3 d_4

i \ j	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

$p_0 p_1$

To fill the table we use bottom up approach

Single matrix: $m[1,1]$, $m[2,3]$ $A_3 = 0$
$A_1 = 0$ $A_2 = 0$ $A_4 = 0$
↳ It is not multiplied with anything so the cost is 0.

2 Matrix: $m[1,2]$ $m[2,3]$

i \ j	1	2	3	4
1		1 -	1 -	3 -
2			2	3
3				3
4				

$A_1 \cdot (A_2)$ (5×4) (4×6) $= 5 \times 4 \times 6 = 120$
$A_2 \cdot (A_3) \rightarrow 4 \times 6 \times 6 \times 2$ $4 \times 6 \times 2 = 48$
$m[3,4]$
$A_3 \cdot (A_4)$ (6×2) (2×7) $= 6 \times 2 \times 7 = 84$

3 Matrix: $m[1,3]$ $A_1 \cdot (A_2 \cdot A_3)$ 2 3 4

$\min \{ m[1,1] + m[2,3] + p_0 p_1 p_3 ;$
$m[1,2] + m[3,3] + p_0 p_2 p_3 ; \}$
$(A_1 \cdot A_2) \cdot A_3$

Page No.
Date

40 = 88

$\min\{ 0 + 48 + 5\times4\times2 ;$
$120 + 0 + 5\times6\times2 ; \}$
60 180

$m[2,4]$ $A_2 \cdot (A_3 \cdot A_4)$

$\min\{ m[2,2] + \min[3,4] + p_1 \times p_2 \times p_4 ;$
$m[2,3] + \min[4,4] + p_1 \times p_3 \times p_4 \}$
$(A_2 \cdot A_3) \cdot A_4$

$= \min\{ 0 + 84 + 4\times6\times7 = 252 ;$
$48 + 0 + 4\times2\times7 = 48 + 56 = 104 ; \}$

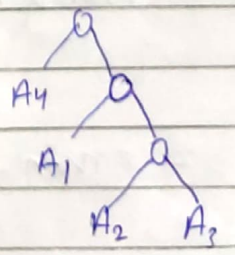
$m[1,4]$ $A_1\ A_2\ A_3\ A_4$
$p_0 p_1$ $p_1 p_2\ p_4$

$\min\{ m[1,1] + m[2,4] + p_0 p_1 p_4 ;$
$m[1,2] + m[3,4] + p_0 p_2 p_4$
$p_0 p_2$ $p_2 p_4$
$m[1,3] + m[4,4] + p_0 p_3 p_4 \}$
$p_0 p_3 p_4$ $p_3 p_4$

$= \min\{ 0 + 104 + 5\times4\times7 = 244$ $A_1 (A_2 A_3 A_4)$
$120 + 84 + 5\times6\times7 = 414$ $(A_1 A_2)(A_3 A_4)$
$88 + 0 + 5\times2\times7 = 158 \}$ $(A_1 A_2 A_3) A_4$

$\Rightarrow (A_1 A_2 A_3) A_4$

$((A_1)(A_2 A_3)) A_4$



$T.C = \frac{n(n+1)}{2}$
$= n^2 \times n = n^3$
$\Theta(n^3)$

$m[i,j] = \min\{ m[i,k] + m[k+1,j] + d_{i-1} \times d_k \times d_j \}$

Page No.
Date

Ques. 2

A_1	A_2	A_3	A_4	A_5
4×10	10×3	3×12	12×20	20×7
p_0 p_1	p_1 p_2	p_2 p_3	p_3 p_4	p_4 p_5

$m[1,1] \to A_1 \Rightarrow 0$, $m[2,2] \to A_2$

$m[3,3] \to A_3$, $m[4,4] \to A_4$, $m[5,5] \to A_5$

i \ j	1	2	3	4	5
1	0	120	264	1080	1344
2		0	360	1320	1350
3			0	720	1140
4				0	1680
5					0

s \ j	1	2	3	4	5
1		1	2	2	2
2			2	2	2
3				3	4
4					4
5					

$m[1,2]$

$A_1 A_2 = 4\times10\times3 = 120$

4×10 10×3

$m[2,3] = 10\times3\times12 = 360$

$A_2 A_3$

10×3 3×12

$m[3,4] = 3\times12\times20$

$A_3 A_4$

3×12 12×20

$m[4,5] = 12\times20\times7$

$A_4 A_5$

12×20 20×7

$A_1 (A_2 \cdot A_3)$

$m[1,3] = \min\{ m[1,1] + m[2,3] + p_0 p_1 p_3,$
$m[1,2] + m[3,3] + p_0 p_2 p_3 \}$ $(A_1 A_2) A_3$

$= \min\{ 0 + 360 + 4\times10\times12 = 840$
$120 + 0 + 4\times3\times12 = 264 \}$

$A_2 (A_3 A_4)$

$m[2,4] = \min\{ m[2,2] + m[3,4] + p_1 p_2 p_4,$
$m[2,3] + m[4,4] + p_1 p_3 p_4 \}$
$(A_2 A_3) A_4$

$= \min\{ 0 + 720 + 10\times3\times20 = 1320$
$360 + 0 + 10\times12\times20 = 2760 \}$

$m[i,k] = \min\{ m[i,k] + m[k+1,j] + p_{i-1} * p_k * p_j \text{ if } i \le k < j$

$0 \quad \text{if } i = j \}$

$A_3 (A_4 A_5)$

$m[3,5] = \min\{ m[3,3] + m[4,5] + p_2 p_3 p_5,$
$m[3,4] + m[5,5] + p_2 p_4 p_5 \}$
$(A_3 A_4) A_5$

$= \min\{ 0 + 1680 + 3 \times 12 \times 7 = 1932$
$720 + 0 + 3 \times 20 \times 7 = 1140$
$A_1 (A_2 A_3 A_4)$

$m[1,4] = \min\{ m[1,1] + m[2,4] + p_0 p_1 p_4,$
$m[1,2] + m[3,4] + p_0 p_2 p_4,$
$m[1,3] + m[4,4] + p_0 p_3 p_4 \}$
~~$m[1,4] +$~~

$= \min\{ 0 + 1320 + 4 \times 10 \times 20 = 2120$ $A_1 (A_2 A_3 A_4)$
$120 + 720 + 4 \times 3 \times 20 = 1080$ $(A_1 A_2)(A_3 A_4)$
$264 + 0 + 4 \times 12 \times 20 = 1224 \}$ $(A_1 A_2 A_3) A_4$

$m[2,5] = \min[m[2,2] + m[3,5] + 10 \times 3 \times 7,$
$m[2,3] + m[4,5] + 10 \times 12 \times 7,$
$m[2,4] + m[5,5] + 3 \times 20 \times 7 \}$

$\min\{ 0 + 1140 + 10 \times 3 \times 7 = 1350$ $A_2 (A_3 A_4 A_5)$
$360 + 1680 + 10 \times 12 \times 7 = 2880$ $(A_2 A_3)(A_4 A_5)$
$1320 + 0 + 3 \times 20 \times 7 = 1460$ $(A_2 A_3 A_4) A_5$

$m[1,5] = \min\{ m[1,1] + m[2,5] + p_0 p_1 p_5,$
$m[1,2] + m[3,5] + p_0 p_2 p_5,$
$m[1,3] + m[4,5] + p_0 p_3 p_5,$
$m[1,4] + m[5,5] + p_0 p_4 p_5 \}$

$= \min\{ 0 + 1350 + 4 \times 10 \times 7 = 1630$ $A_1 (A_2 A_3 A_4 A_5)$
$120 + 1140 + 4 \times 3 \times 7 = 1344$ $(A_1 A_2)(A_3 A_4 A_5)$
$264 + 1680 + 4 \times 12 \times 7 = 2016$ $(A_1 A_2 A_3)(A_4 A_5)$
$1080 + 0 + 4 \times 20 \times 7 = 1640 \}$ $(A_1 A_2 A_3 A_4) A_5$

Page No.
Date

Algorithm

```
Matrix chain (p, n)
for (i = 1 to n)
        m[i, j] = 0
for (l = 2 to n)
    for (i = 1 to n - l + 1)        >> Loop half of the table
        j = i + l - 1
        m[i, j] = ∞
        for (k = i to j - 1)
            q = m[i, k] + m[k+1, j] + p[i-1] * p[k] * p[j]
            if q < m[i, j]
                m[i, j] = q
                s[i, j] = k
return m & s
```

T.C = $O(n^3)$

S.C = $O(n^2)$

ENQUEUE → Adding element into queue
DEQUEUE → Deleting " from "

Page No. _____
Date

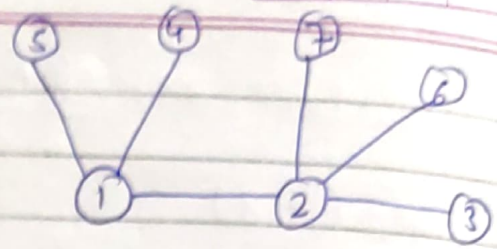
BFS (Breadth First Search)

```
BFS(G, s)
for each vertex u ∈ G.V - {s}
        u.color = WHITE          setting color
        u.d = ∞                  Distance = ∞
        u.π = NIL                Parent = nil
s.color = GREY
s.d = 0
s.π = Nil
Q = ϕ
ENQUEUE(Q, s)                    Insert source into queue
while Q ≠ ϕ
    u = DQUEUE(Q)                deleting element from queue.
    for each v ∈ Adj[u]
        if v.color == WHITE
            v.color = GREY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
u.color = BLACK
```

Page No.
Date

* (*) Start from any vertex
1. Visiting a vertex
2. Exploration of vertex

Ex1



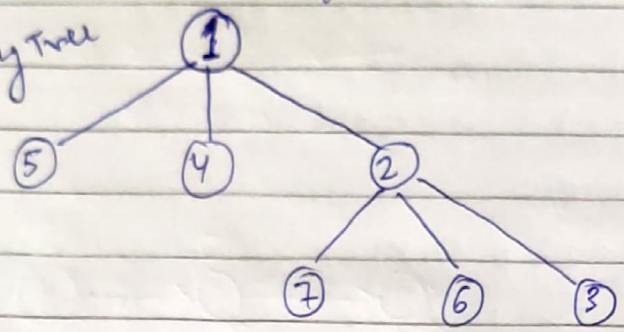
BFS: 1, 5, 4, 2, 7, 6, 3

1 — 5, 4, 2
2 — 7, 6, 3

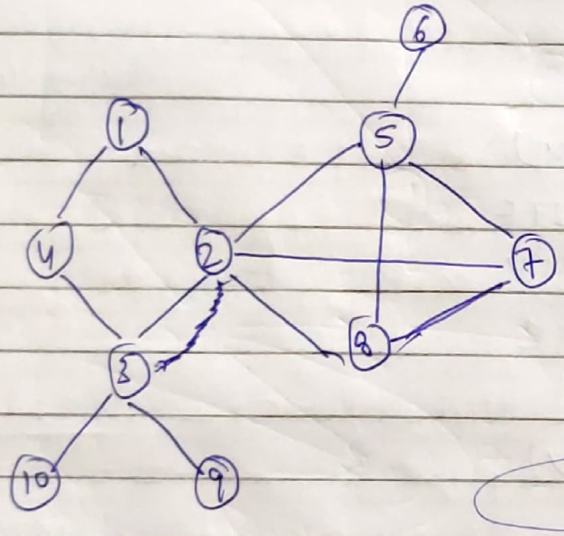
Q | 1 | 5 | 4 | 2 | 7 | 6 | 3 |

↳ Taking out 1 from queue and exploring it

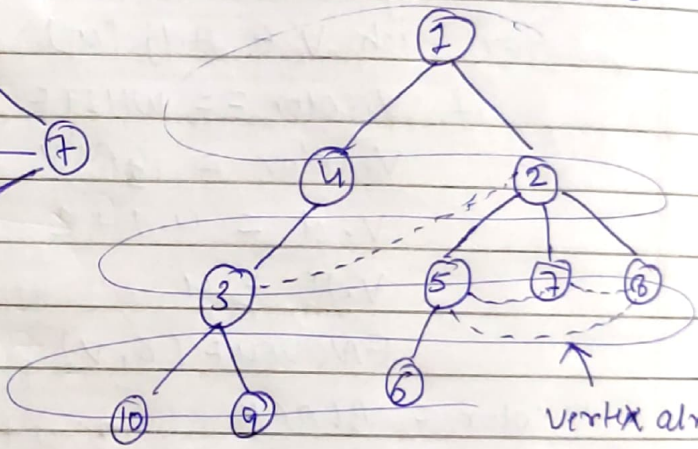
BFS Spanning Tree



Ex2



BFS Spanning Tree



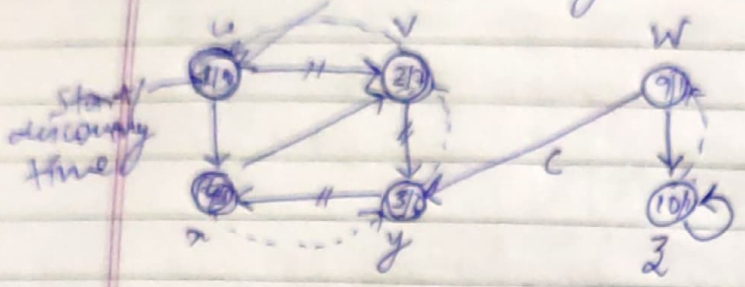
BFS: 1, 4, 2, 3, 5, 7, 8, 10, 9, 6

Q | 1 | 4 | 2 | 3 | 5 | 7 | 8 | 10 | 9 | 6 |

* Start BFS from any vertex
* When you are exploring any vertex then you can visit its adjacent vertex in any order.
* We should select our next vertex for exploration from a queue only.

Depth first search (DFS)

end/finishing time

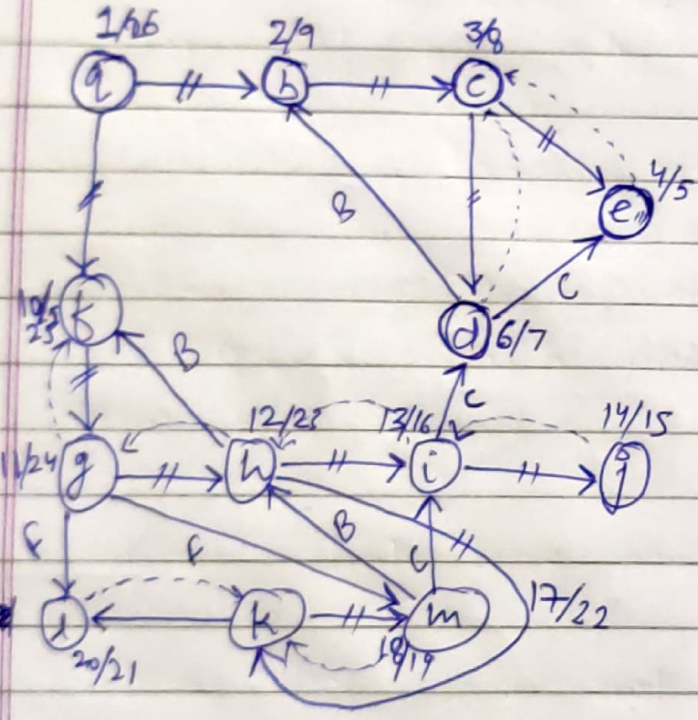


start/discovery time

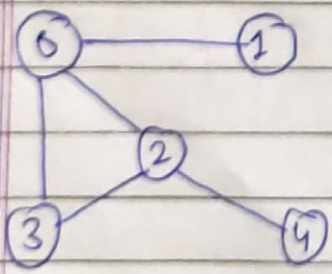
(u, x) forward edge
(x, v) backward edge (have path)
(w, y) cross edge (no path)

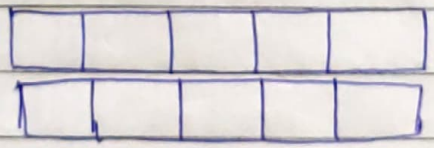
~~u, v, x, y, w, z~~

u v y x w z

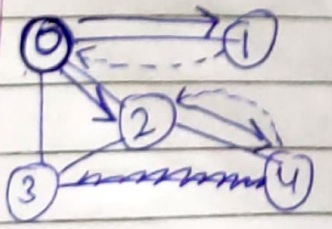


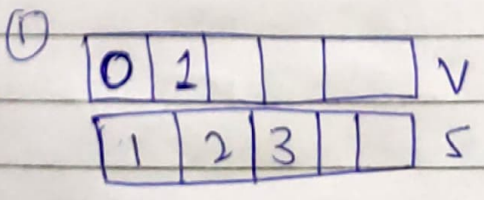
q b c e d f g h i j k m

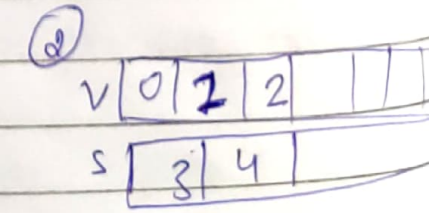


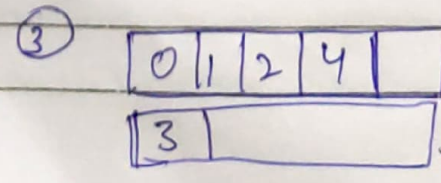


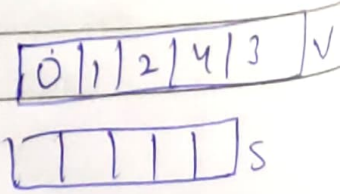
Visited
stack











Page No.
Date

Algorithm :-

```
DFS (G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color = WHITE
7          DFS Visit (G,u)          T.C = O(V+E)
```

```
DFS Visit (G,u)
1.  time = time + 1
2.  u.d = time
3.  u.color = GRAY
4.  for each v ∈ G.Adj[u]
5.      if u.color == WHITE
6.          v.π = u
7.          DFS Visit (G,v)
8.  u.color = BLACK
9.  time = time + 1
10. u.f = time
```

Page No.
Date

P, NP & NP Complete & NP Hard

#P

P → Problem can be solved in Polynomial times
NP → Problems which can be solved in a Non-deterministic machine in Polynomial time.

* A Polynomial time algo. is the one whose worst case running time is bounded above by a polynomial function.
* Polynomial time is imp. because for large problem sizes all non-poly. time algo. will take forever to execute.
* A problem is called intractable if it is impossible to solve it with poly. time algo.
* We can group the problems into 3 categories
  1. Problems for which polynomial times algorithm have been found.
     Ex:- BFS, DFS
  2. Problems that have been proven to be intractable
     → Hemantonian cycle
  3. Problems that have not been proven to be intractable but for which poly. time algo. have never been found.

#1 Polynomial Time category → Any time problem for which we have found a poly. time algo.
Ex:- sorting, searching, Matrix Multiplication, Chain Matrix Mul. Shortest Path, MST etc

#2 Intractable category → 2 types of problems
2.1. Those that req. a non-polynomial amount of output.

P -> set of all decision problems that can be solved in polynomial time.

Page No.
Date

Ex: Determining all Hamiltonian circuits
Ex2: There that produce a reasonable amount of output but the processing time is too long.
Ex:- Halting Problem.

* 3. Unknown category → Many problems belong to this category
Ex:- 0-1 Knapsack problem, Travelling salesman Problem, Any problem which is solved using backtracking and branch & bound.

* NP

Decision Problem → The problems that has Yes/No as answers
→ We can always convert a non-decision problem (any optimization problem) into a decision problem.
Ex:- In 0-1 Knapsack instead of just asking for the optimal profit we can instead ask if the optimal profit exceeds some given number.

Definitions

Set P: The set P is the set of all decision problems that can be solved by polynomial time algorithms.

Set NP: The set NP is the set of all decision problems that can be solved by a polynomial time non-deterministic algo.

* A Polynomial time non-deterministic algo. is an algo. i.e. broken into 2 stages.
1. Guessing Stage (Non-deterministic)
2. Verification stage (Deterministic)
↳ always done in polynomial times.

CNF Satisfiability problem

$x_i = \{x_1, x_2, x_3\}$

$(x_1 \vee \bar{x} \vee x_3) \wedge (\bar{x_3} \vee x_2 \vee \bar{x_3})$

x_1	x_2	x_3
0	0	0
0	0	1
0	1	1

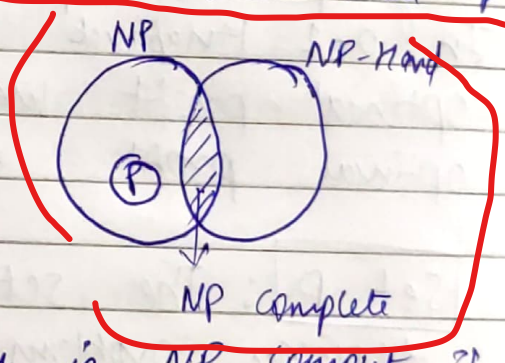
3 CNF

Reducibility

$A \alpha_p B$

NP-Hard

If all NP problems can be reduced to a given problem in polynomial time then that problem is NP Hard.



NP-Complete :- A problem is NP complete if

(i) In NP (i) It is in NP and

(ii) In NP Hard (ii) for every problem A in NP A is polynomial time reducable to B. ($A \alpha_p B$)

Cooks Theorem

If SAT is in P then P = NP