30/1/23

Module 1

Q) Algo Add()

```
{
    S=0                          ⟶ 1
    for(i=0; i<n; i++)           ⟶ n+1
    {
      S= S+ A[i];                ⟶ n
    }
     return(s);                  ⟶ 1
}
```

Complexity : $1 + n+1 + n + 1$
$\longrightarrow 2n+3$
$= O(n)$

Q) Add (A, B, n)

```
{
   for(i=0; i<n; i++)            ⟶ n+1
   {
      for(j=0; j<n; j++)         ⟶ n(n+1)
      {
         start;                  ⟶ n*n = n^2
      }
   }
```

Complexity : $n+1 + n(n+1) + n^2$
$= n+1+n^2+n+n^2$
$= 2n^2+2n+1$
$\longrightarrow O(n^2)$

PAGE No.
DATE / /

Q)
```
p = 0;
for(i=1; p<=n; i++)
{
    p = p + i;
}
```

for P: $1 + 2 + 3 + \cdots + k$

$$\frac{k(k+1)}{2} > n \quad \text{(Slopping condition)}$$

$$k^2 > n \qquad (k > \sqrt{n})$$

$$\therefore O(\sqrt{n})$$

Q)
```
for(i=1; i<n; i=i*2)
{
    statement
}
```

(i)
1
2
4
8
16

$i = 1 \qquad i = 1*2 = 2^1 = 2$

$i = 2 \qquad = 2*2 = 2^2$

$\vdots \qquad = 2*2*2 = 2^3$

$i = k \longrightarrow i = 2^k$

$\therefore 2^k > n$

$k = \log_2 n$

$\therefore$ Complexity $= O(\log_2 n)$

Q)
```
for(i=n; i>=1; i=i/2)
{
    statement
}
```

$\therefore O(\log n)$ = complexity.

PAGE No.
DATE / /

Q)
```
for (i=0; i*i<n; i++)
{
    statement
}
```

$i^2 > n \longrightarrow i = \sqrt{n}$

$O(\sqrt{n})$ → complexity

Q)
```
p=0;
for(i=1; i<n; i=i*2)        // i>n
{
    p++;
}
for(j=1; j<p; j=j*2)
{
    statement;
}
```

$i*2 > n$

$\therefore n/2$

i	p
1	1
2	2
4	3
8	

$P = \log_2 n$

$P: \log_2(n)$

$j > \log_2 n$

$\log(\log n)$

$\therefore j = \log_2(\log_2 n)$

Q)
```
for (i=0; i<n; i++)        → (n+1)
{
    for(j=1; j<n; j=j*2)   → n(log2 n)
    {
        statement          → n log n
    }
}
```

$O(n \log n)$ [takes more time ($O \log(\log n)$)]

Q.) Algo mult (A, B, n)

```
{
  for (i=0; i<n; i++)                    ⟶ n+1
  {
    for (j=0; j<n; j++)                  ⟶ n(n+1)
    {
      statement;                         ⟶ n²
      for (k=0; k<n; k++)                ⟶ n²(n+1)
      {
        statement;                       ⟶ n³
      }
    }
  }
}
```

Total computation : $n+1 + n(n+1) + n^2 + n^2(n+1) + n^3$

$= n+1+n^2+n+n^2+n^3+n^2+n^3$

$= 2n^3 + 3n^2 + 2n + 1$

$= \underline{O(n^3)}$

Q.) void function (int n)

```
{
  int count = 0;                         ⟶ 1
  for (int i = n/2 ; i<=n; i++)          ⟶ n/2 + 1
  {
    for (j=1; j<=n; j=2*j)               ⟶ n/2 (log n)
    {
      for (k=1; k<=n; k=k*2)             ⟶ n/2 (log n)(log n)
      {
        count++;
      }
    }
  }
}
```

$\therefore \frac{n}{2}(\log n)^2$

∴ when not nested.

```
fn
for(int i=n/2; i<=n; i++) ⟶ n/2 + 1
{}
for(int j=1; j<=n; j=j*2) ⟶ log n
{}
for(k=1; k<=n; k=k*2) ⟶ log n
{
  count++;
}
```

$$\therefore \frac{n}{2} + 1 + 2\log(n) \longrightarrow \therefore \underline{O(n)}$$

20 mks - weightage.
Master method / Recursion tree } ☆
Quick Sort tree with best / worst case.

PAGE No.
DATE / /

Module - 2

Binary Search Maxmin problem.
Quick Sort & Merge Sort.

∴ General form.

Recurrence Relation

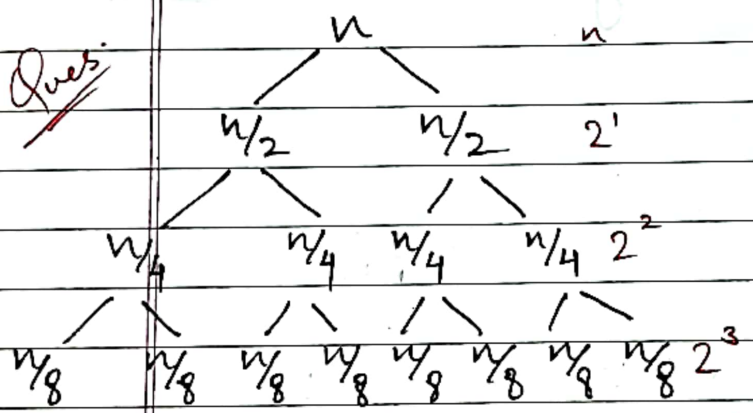
$$T(n) = aT(n/b) + f(n)$$

n → size of original prob.
a → no. of subproblems
b → size of each subprob.
f(n) → time to divide & combine subproblems

Recurrence Eqn of Binary Search for divide & conquer

$$\Rightarrow T(n) = 1 \cdot T(n/2) + 1$$
$$= 1 \cdot \log_2 n + 1$$
$$\therefore T(n) = O(\log_2 n)$$

Ques:



$n = 2^k$ ✓

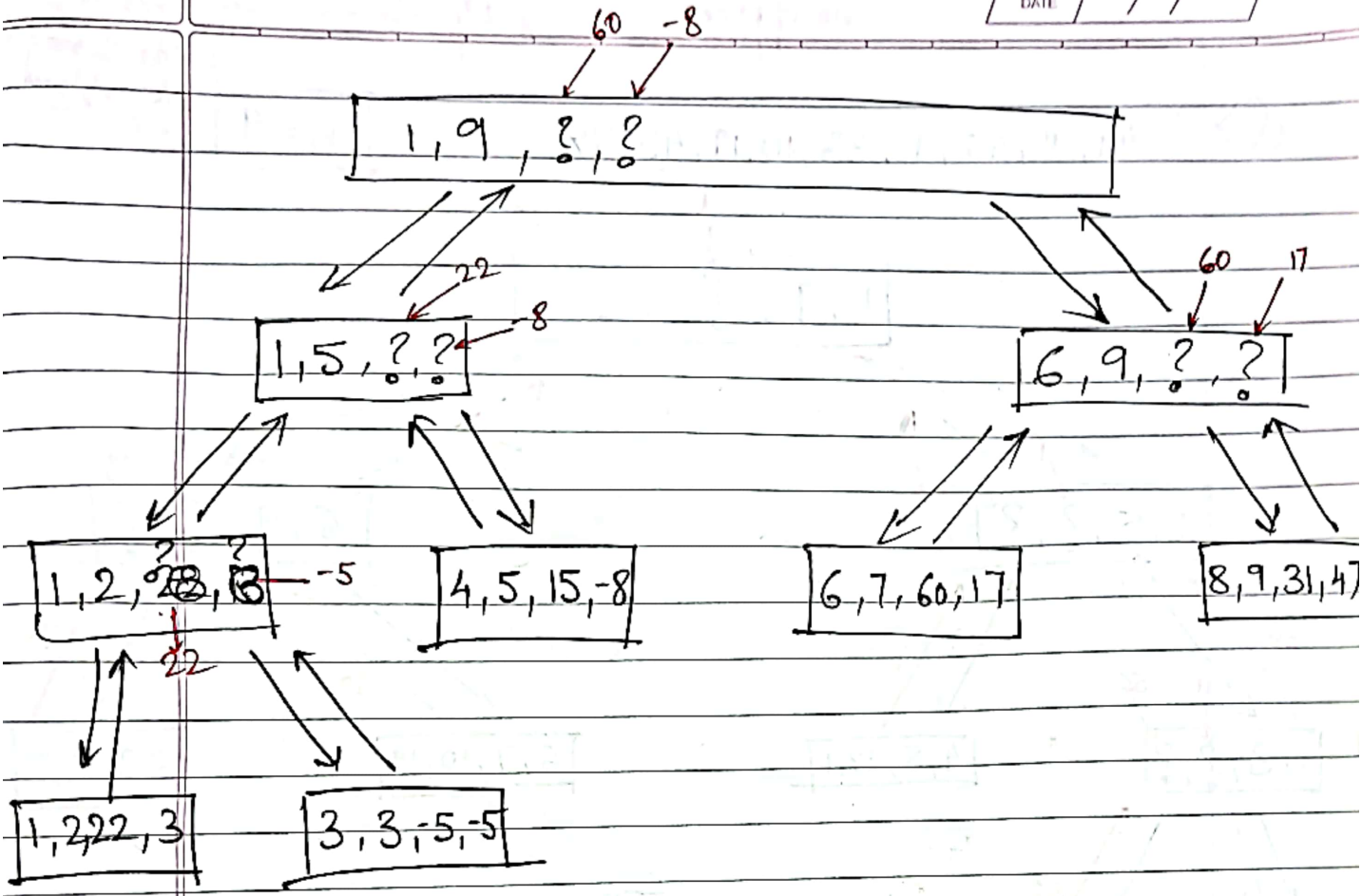
$k = \log_2 n$

Recursion tree of maxmin prob.

Q.) Find Max & Min in array :-

Ques on next page

22, 13, -5, -8, 15, 60, 17, 81, 47 (n=9)



$$\therefore T(n) = a\left(T(n/b)\right) + f(n)$$

$n=2 \Longrightarrow \therefore T(2) = 2(n-1)$

$= 2n-2$

$\therefore O(n)$

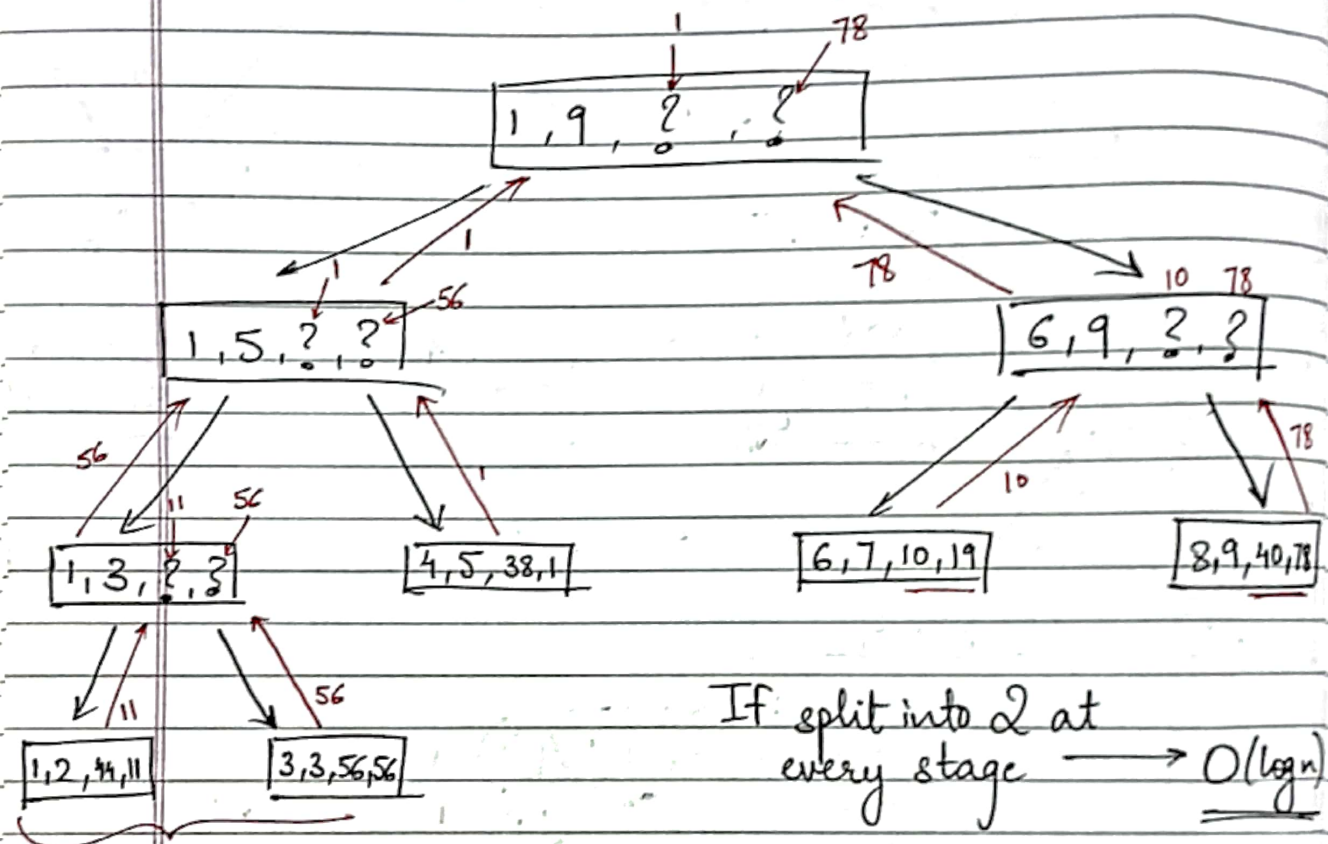
$n=1$

$n>2, \quad T(n) = 2 \cdot T(n/2) + 2$

$$T(n) = a \cdot T(n/b) + f(n)$$

- a → no. of probs
- n/b → size of each sub prob
- $f(n)$ → time req. to divide & combine sub problems
- no. of outputs at each stage

Q:) 44, 11, 56, 1, 38, 10, 19, 40, 78 — $n = 9$



If split into 2 at every stage → $O(\log n)$

$\therefore T(n) = 2\log(n/2) + 2$ — If not every where ↓

$\therefore n' > 2$ → $O(n)$

$\therefore O(n)$

$F(n) = n^{\log_b a} = n^1$

$F(n) = n$, $f(n) = 2$

$F(n) > f(n)$

$O(n^{\log_b a})$

$O(n)$

Q:) In max min,

Step 1 → Keep Dividing the entire array into 2 parts until either 2 or 1 element is left.

Step 2 → From the leaf, we send the max & min of each part upward till we find for the entire array

combine → see algo for it

PAGE No. / DATE / /

* Master Method :-

Also. $F(n) = n^{\log_b a}$

Case 1 :- $f(n) < n^{\log_b a}$

$\therefore \boxed{T(n) = O(n^{\log_b a})} \longrightarrow$ best

Case 2 :- $f(n) = n^{\log_b a}$

$\therefore \boxed{T(n) = O(n^{\log_b a} \cdot \log n)} \longrightarrow$ average.

Case 3 :- $f(n) > n^{\log_b a}$

$\therefore \boxed{T(n) = O(f(n))} \longrightarrow$ worst.

Q) $T(n) = 1 \cdot T(n/2) + 1$

$\therefore a=1, b=2, f(n)=1$

$n^{\log_b a} \longrightarrow n^{\log_2 1} \longrightarrow n^0 = 1 \quad \therefore F(n)=1$

$\therefore f(n) = F(n)$

Thus case (ii) $\longrightarrow T(n) = O(\log n)$

Q) $T(n) = 2 \cdot T(n/2) + n^2$

$\therefore a=2, b=2, f(n)=n^2$

$F(n) = n^{\log_b a} = n^{\log_2 2} = n \quad \therefore f(n) > F(n)$

$f(n) = n^2$

Thus case (iii) $\longrightarrow T(n) = O(f(n))$

$= O(n^2)$

PAGE No.
DATE / /

Q) $T(n) = T(9n/10) + n$

$\therefore a=1,\ b=10/9,\ f(n)=n$

$F(n) = n^{\log_b a} = n^{\log_{10/9} 1} = n^0 = 1$

$f(n)=n$

$\therefore f(n) > F(n)$

$\therefore$ case (iii) $\longrightarrow T(n) = O(f(n))$

$= O(n)$

Q) $T(n) = 4 \cdot T(n/2) + n^3$

$a=4,\ b=2,\ f(n)=n^3$

$F(n) = n^{\log_b a} = n^{\log_2 4} = n^2$

$f(n)=n^3$

$f(n) > F(n)$

$\therefore$ case (iii) $\longrightarrow T(n) = O(f(n)) = O(n^3)$

Q) $T(n) = 8 \cdot T(n/4) + n^2$

$a=8,\ b=4,\ f(n)=n^2$

$F(n) = n^{\log_b a} = n^{\log_4 8} = n^{3/2}$

$f(n)=n^2$

$f(n) > F(n)$

$\therefore$ case (iii) $\longrightarrow T(n) = O(n^2)$

Q) $T(n) = 3T(2n/6) + n$

$a=3,\ b=3,\ f(n)=n$

$F(n) = n^{\log_b a} = n^{\log_3 3} = n$

$f(n)=n.$

$\therefore F(n) = f(n)$

$\therefore$ case (ii) $\longrightarrow T(n) = O(n^{\log_b a} \cdot \log n)$

$= O(n \cdot \log n)$

PAGE No.
DATE / /

* Substitution Method.

Step 1 :- Identify a, b & f(n)
Step 2 :- Guess the solution
Step 3 :- Verify the guessed solution using mathematical induction.

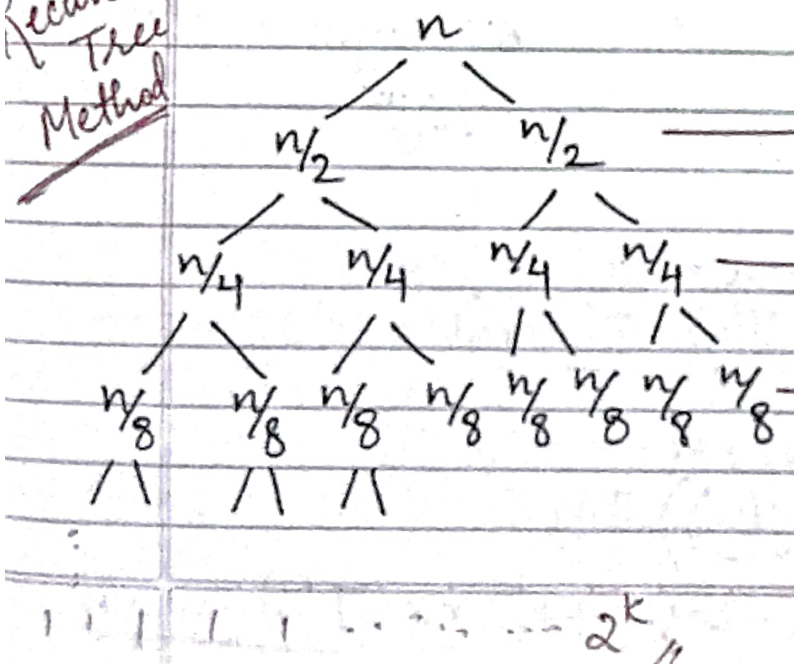
Example

$\therefore T(n) = \begin{cases} 1, & T(1) \\ T(n-1)+n, & \text{if } (n>1) \end{cases}$ — Insertion Sort

$\therefore T(1) = 1$
$T(2) = T(2-1)+2 = 1+2$
$T(3) = T(3-1)+3 = 1+2+3$
⋮
$T(n-1) = T(n-2)+(n-1)$
$= \Sigma(1+2+3\ldots.(n-1)) + (n-1)$
$= \frac{n(n-1)}{2} + (n-1) = \frac{n^2+n-2}{2} = O(n^2)$

Q:) $T(n) = 2 \cdot T(n/2) + n$. ∴ Merge Sort

Recurrence Tree Method



Since cost of 1 → 1
∴ cost of $2^k \to 2^k$
$n = 2^k$

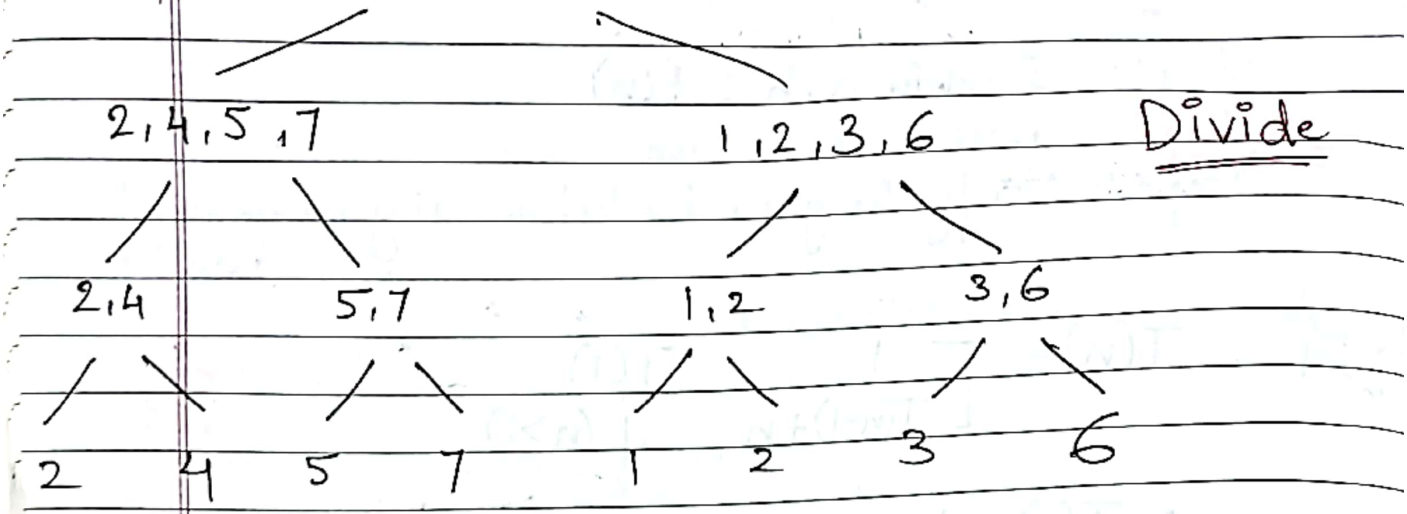
∴ n log(n)

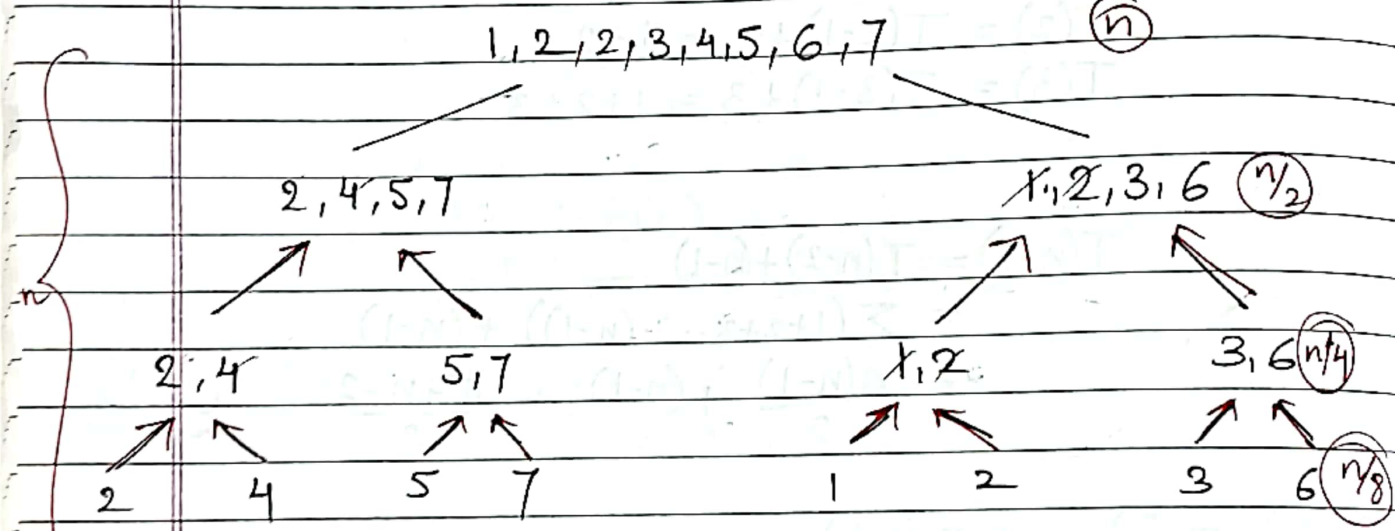
Example of Merge Sort

PAGE No. / DATE / /

1 2 3 4 5 6 7 8

Q) 2, 4, 5, 7, 1, 2, 3, 6





$T(n) = 2T(n/2) + n$ ∴ $2^k = n$

$k = \log_2 n$

∴ $n \log n$

Method 1

Master

$a = 2, b = 2, f(n) = n$

$F(n) = n^{\log_b a} = n^{\log_2 2} = n = f(n)$

∴ case (ii) → $T(n) = O(n \log n)$

Method 2

Method 3

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

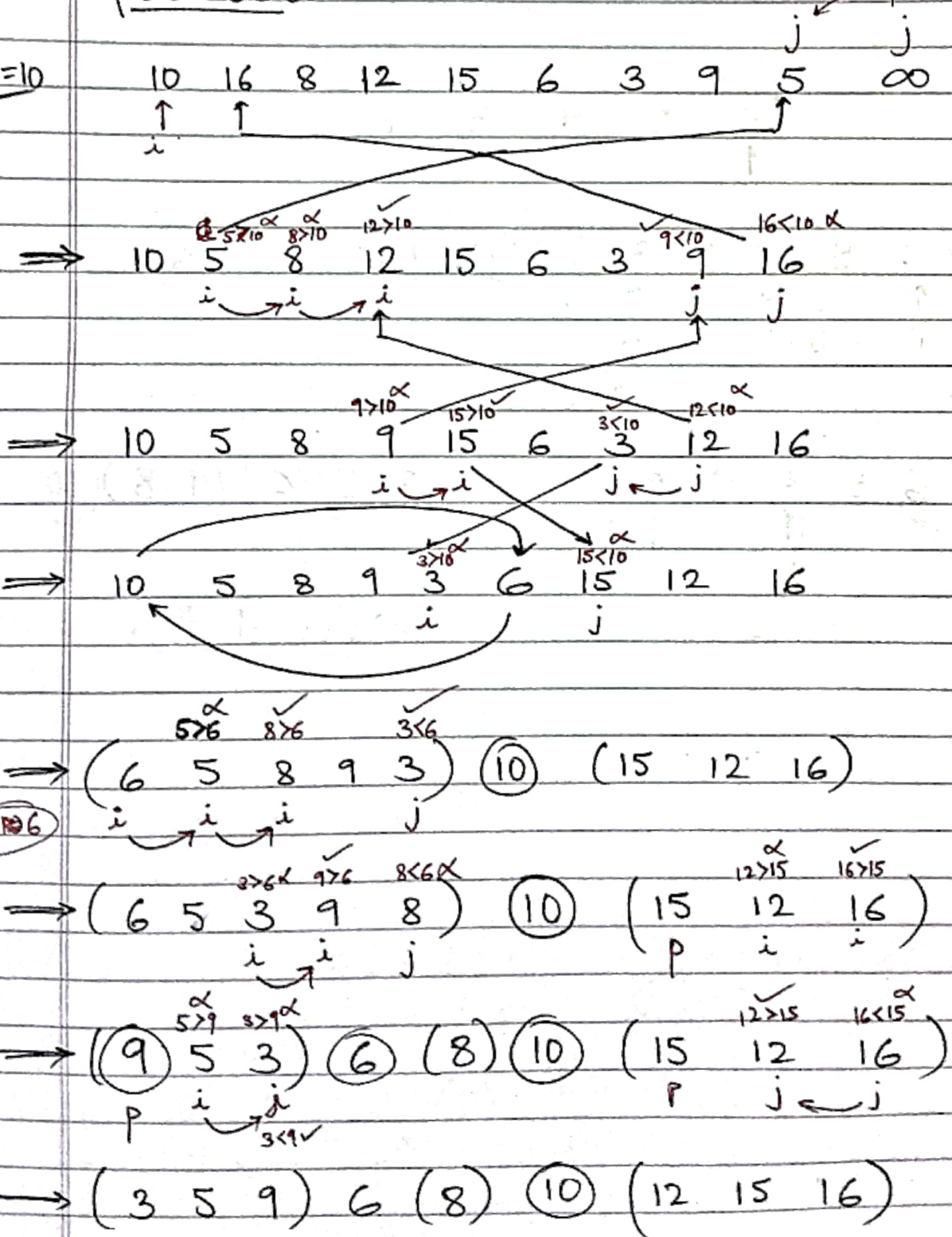
$= 2\left(\frac{n}{2}\log\left(\frac{n}{2}\right)\right) + n$

$= n[\log n - \log 2] + n \Rightarrow T(n) = n \log n$

Quick Sort & Merge Sort

Ques. { explain algo
most apt to solve quicksort
determine time complexity.

* Quick Sort :-



PAGE No.
DATE / /

$$T(n) = a\,T(n/b) + f(n)$$

$\therefore T(n) = 2T$ ← Worst case

$T(n) = 2T(n/2) + n$ ← Best case

$\therefore$ 6 5 9 3 8

↑

(6 5 8 9 3) (10) (12 15 16)

↑ ~ ↑ j

P a[i]=8 j-1

3<6

↓

6 5 3 9 8 ⟶ (3 5) 6 (9 8) 10 ---

// ↑ ↑

j j

⇓

Solved further

Quick Sort Algo (Divide & Conquer)

A	10	15	1	2	9	16	11

pivot

key

Partition	Pivot	Partition
< Pivot		> Pivot

< Pivot → left
> Pivot → Right

Q.

0	1	2	3	4	5	6	7	8
7	6	10	5	9	2	1	15	7

pivot, start, start, end, end

(start = left to right)
(end = right to left)
When to stop → If ele > 7 ⟹ stop
→ If ele > 7 → contain

Swap start to end

7	6	7	5	9	2	1	15	10

start, start, end, end

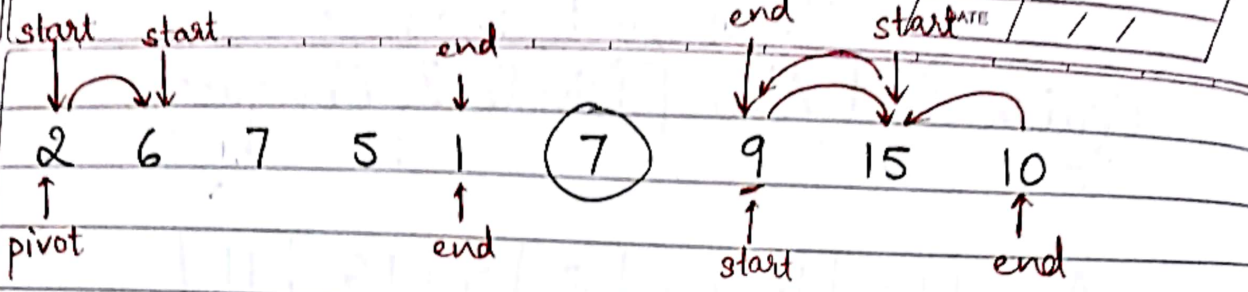
Swap end

7	6	7	5	1	2	9	15	10

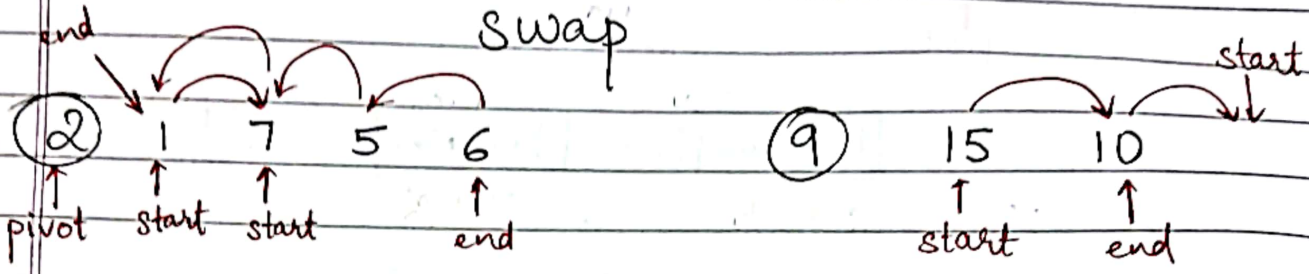
start, end, start

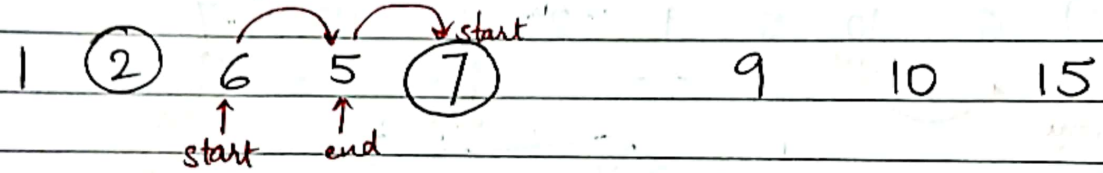
Now start > end, → no swap
∴ swap pivot & end

2	6	7	5	1	7	9	15	10



Swap



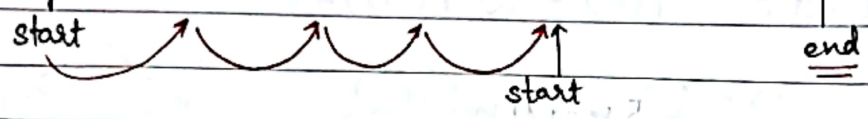


1 2 5 6 7

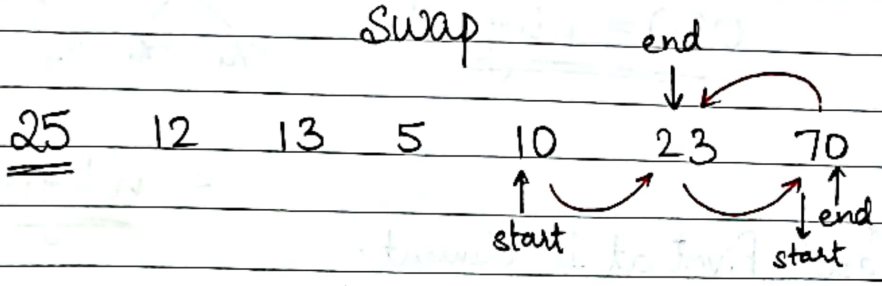
1 2 5 6 7 7 9 10 15

Quick Sort Example

Q.) 25 12 13 5 70 23 10

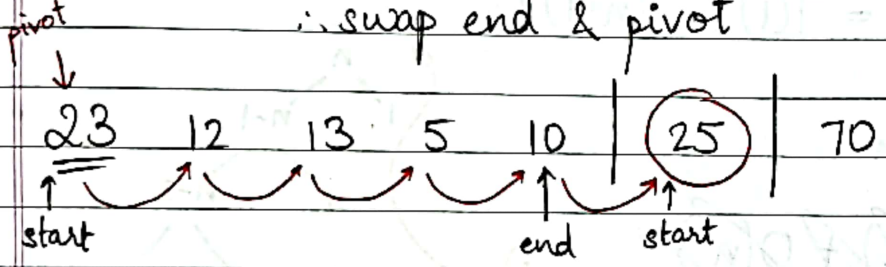


Swap



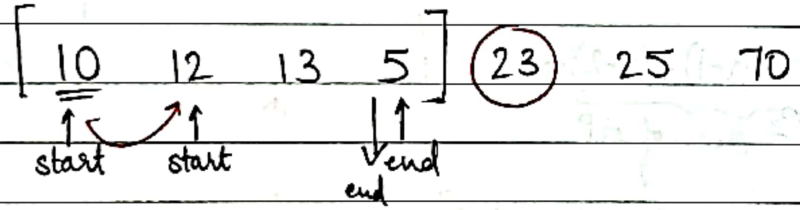
start > end

∴ swap end & pivot

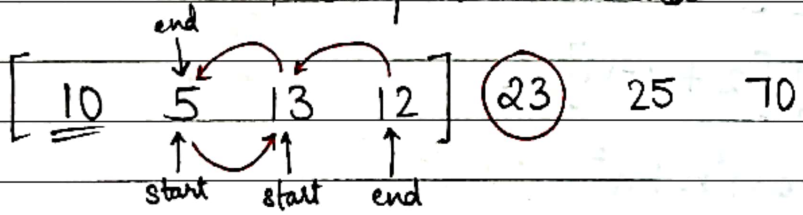


∴ start > end

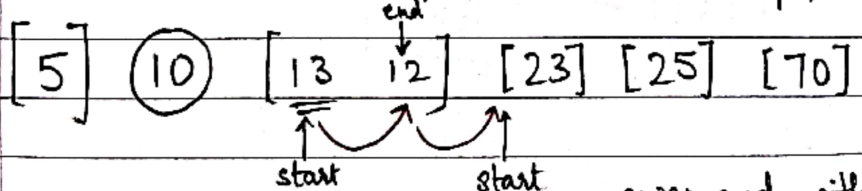
∴ swap end with pivot



∴ swap start with end



∴ ~~swap~~ start > end, swap pivot with end



∴ swap end with pivot.

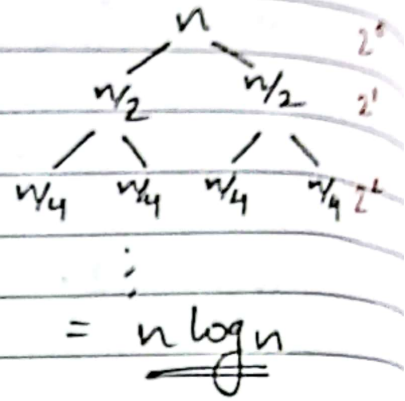
5 10 12 13 23 25 70.

Complexity for Quick Sort

→ Best Case: $T(n) = 2 \cdot T(n/2) + n$

~~$T(n) = O(n \log n)$~~

$O(n) = n \log n$

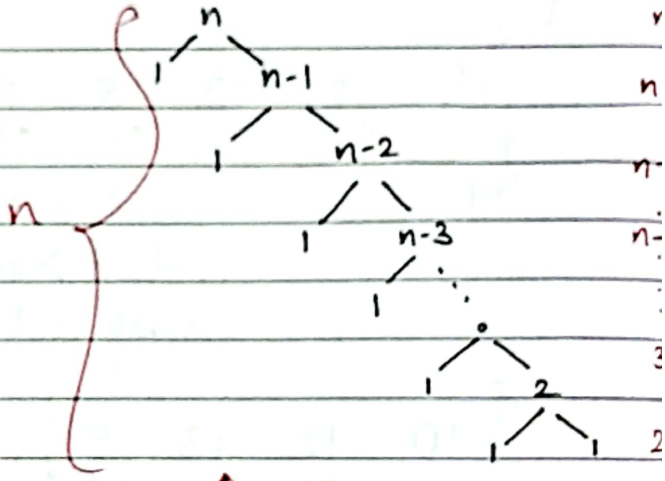


→ Worst Case: Pivot at 1st element.

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

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

~~$= O(n^2)$~~



Total Cost

↳ $n + n + (n-1) + (n-2) \cdots 1$

→ $n + \underbrace{\text{Sum of AP}}$

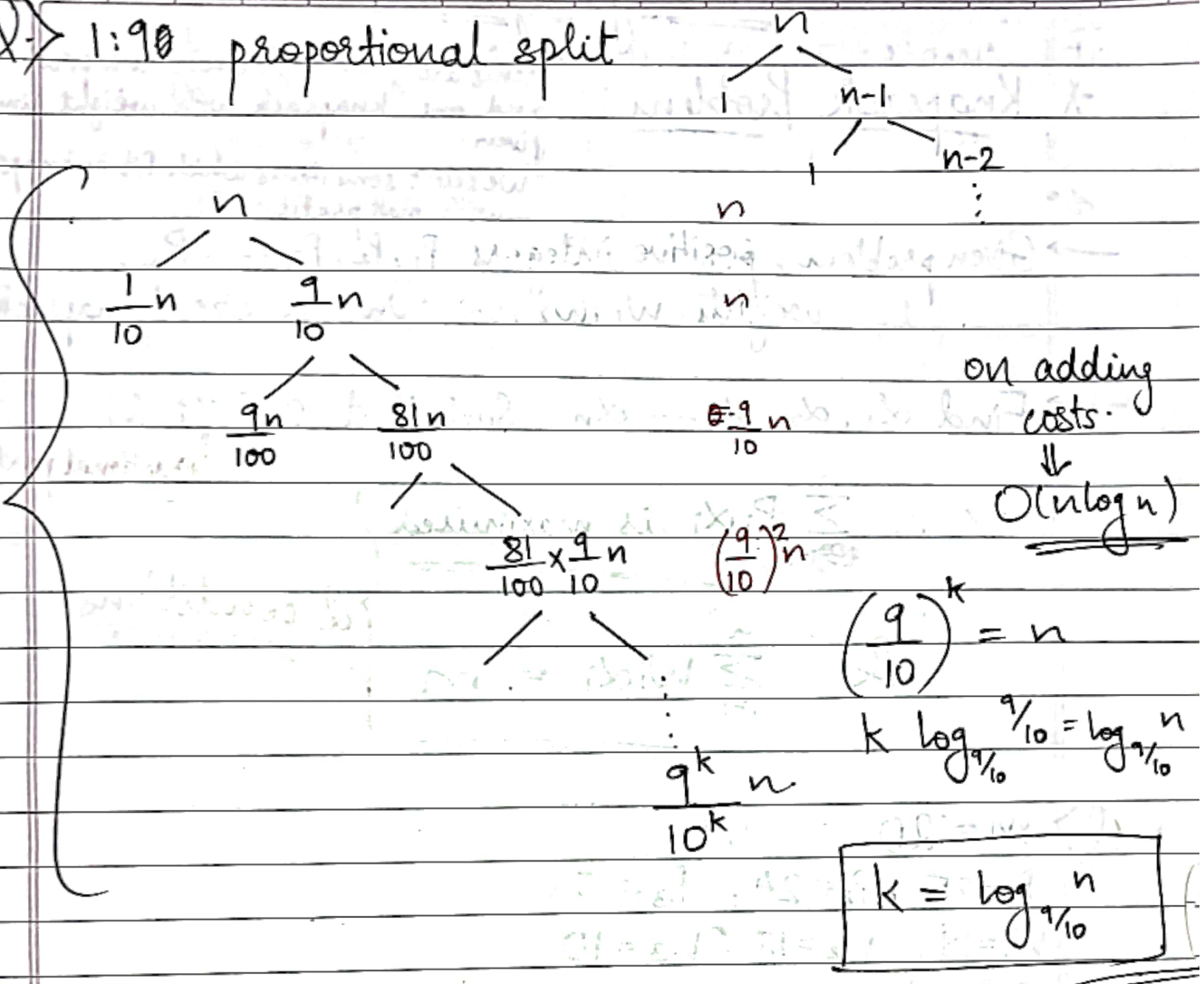
→ $n + \frac{n}{2}[2 + ((n-1)-1)(1)]$

→ $n + \frac{n}{2}[2 + 2 - n]$

→ ~~$n + n^2$~~ $n + n^2 \leq cn^2$

$\therefore T(n) \Rightarrow O(n^2)$

Q:> 1:9 proportional split



α —— α —— α —— α ——

Binary → a=1, b=2, f(n)=1

Max-Min → a=2, b=2, f(n)=2 (max & min)

Merge → a=2, b=2, f(n)=n

Quick Sort → Best: a=2, b=2, f(n)=n

↳ Worst: T(1) + T(n-1) + n

Module 2 :-

1st & 2nd

Divide & Conquer = Write algorithm

Master, Rec, Substitution = Analysis

Greedy Algorithm

PAGE No.
DATE / /

* **Knapsack Problem :-** "items are given with weights & profits and one knapsack with weight limit is given.
∴ We select some items which fit in knapsack with max profit.

→ Given problem, positive integers $P_1, P_2, P_3 \dots P_n$, weights $W_1, W_2, \dots W_n$ & size of knapsack m

→ Find $\alpha_1, \alpha_2, \alpha_3 \dots \alpha_n$ Such that $0 \leq \alpha_i \leq 1$ ← fractional part of object

$\therefore \sum_{i=1}^{n} P_i \alpha_i$ is maximised

& $\sum_{i=1}^{n} W_i \alpha_i = m$

} 2 conditions

Q) $m = 20$
$P_1 = 25$, $P_2 = 24$, $P_3 = 15$
$W_1 = 18$, $W_2 = 15$, $W_3 = 10$
3 feasible, 1 optimal

Strategies :-
1] Largest profit
2] Smallest weight
3] Ratio of P_i & W_i

} Very Crucial to solve fractional knapsack

Solution :-

1] largest profit

$18 \longrightarrow 25 \quad = 25$

$15 \longrightarrow 24 \times \frac{2}{15} = 3.2$

$\therefore$ Total Profit = 28.2 $\left(1, \frac{2}{15}, 0\right)$

2] Smallest

3] Rati

$\frac{P_i}{W_i}$

∴

2] Smallest weight

$10 \longrightarrow 15 \quad = 15$

$15 \longrightarrow 24 \times \frac{10}{15} \quad = 16$

$\therefore$ Total profit = 31 $\left(0, \frac{10}{15}, 1\right)$

3] Ratio of P_i & W_i :-

$$\frac{P_1}{W_1} = \frac{25}{18}, \quad \frac{P_2}{W_2} = \frac{24}{15}, \quad \frac{P_3}{W_3} = \frac{15}{10}$$

$\therefore P_1/W_1 = 1.39$ — (3)

$P_2/W_2 = 1.6$ — (1)

$P_3/W_3 = 1.5$ — (2)

$\therefore (15) \longrightarrow 24 \quad = 24$

$(10) \longrightarrow 15 \times \frac{5}{10} \quad = 7.5$

$\therefore$ Total profit = 31.5 $\left(0, \frac{5}{10}, \frac{1}{2}\right)$

	$(\alpha_1, \alpha_2, \alpha_3)$	$\Sigma W_i x_i$	$\Sigma P_i \alpha_i$
1]	$(1, 2/15, 0)$	20	28.2
2]	$(0, 10/15, 1)$	20	31
3]	$(0, 1, 1/2)$	20	31.5

Q) Find the fractional knapsack soln. of
$m = 100$

	Wt.	Profit
A	90	30
B	50	50
C	20	70
D	35	20

Soln.

1] Largest Profit

	Wt.	Profit
C	20	70
B	50	50
A	$90\left(\frac{30}{90}\right)$	$30 \times \frac{30}{90} = 10$

$\left(\frac{1}{3}, 1, 1, 0\right)$

$\therefore$ Total Profit $= 70 + 50 + 10 = 130$

2] Smallest Weight

	Wt.	Profit
C	20	70
D	35	20
B	$50\left(\frac{45}{50}\right)$	$50 \times \frac{45}{50}$

$\left(0, \frac{45}{50}, 1, 1\right)$

Total Profit $= 70 + 20 + 45 = 135$

3] Ratio of P

$P_1/$
$P_2/$
$P_3/$
$P_4/$

C
B
D

Who

PAGE No.
DATE / /

3] Ratio of Pi & Wi :-

$P_1/W_1 = 30/90 = 0.33$ (4)
$P_2/W_2 = 50/50 = 1$ (2)
$P_3/W_3 = 70/20 = 3.5$ (1)
$P_4/W_4 = 20/35 = 0.57$ (3)

	Wt.	Profit.
C	20	70
B	50	50
D	35	$20 \times \frac{30}{35}$

$(0, 1, 1, \frac{30}{35})$

∴ Total Profit. = 70 + 50 + 17.14
= 137.14

When solving fractional knapsack, remember 3 steps.

1. Largest Profit
2. Smallest Weight
3. Profit to Weight Ratio

} Showing all 3 steps is very imp.

Spanning tree of a graph is a method to travel across all vertices

& Minimum spanning tree is a spanning tree with minimum combo of weights of edges.

PAGE No.
DATE / /

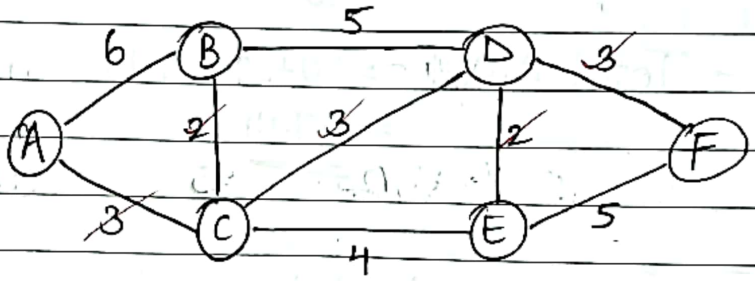
Minimum Spanning Tree (MST) using Prims Algorith

1. $x \in V$, let $A = \{x\}$, $B = V - \{x\}$
2. Select $(u,v) \in E$, $u \in A$, $v \in B$ such that (u,v) has smallest weight betn A & B.
3. Put (u,v) in the tree. $A = A \cup \{v\}$, $B = B - \{v\}$
4. If $B = \phi$, stop; otherwise go to step 2.

Remove all loops initially along with parallel edges.

Time Complexity: $O(n^2)$; $n = |V|$

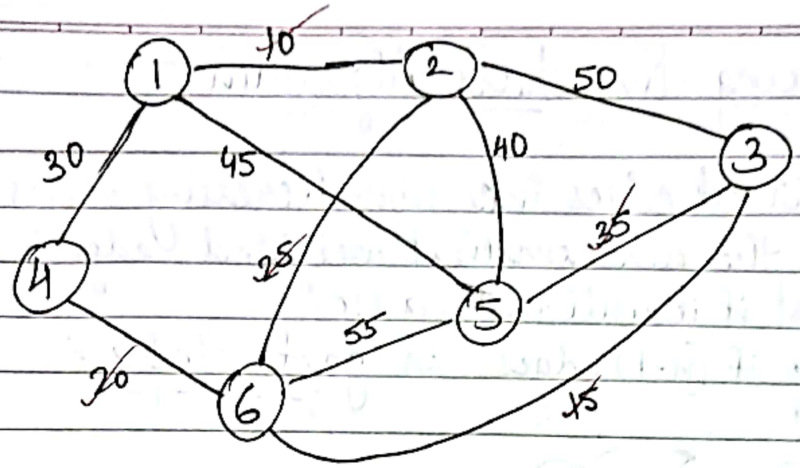
Q)



Soln:

Vertices Covered	Edge taken	Cost	Updated MST
(A) →	(A,B), (A,C)	3	A—C
(A,C) →	(A,B), (C,B), (C,D) (C,E)	2	A—C—B
(A,C,B) →	(A,B), (C,D), (C,E), (B,D)	3	A—C—D, C—B
(A,C,B,D) →	(A,B), (C,E), (B,D), (D,E), (D,F)	2	A—C—D—E, C—B
(A,C,B,D,E) →	(A,B), (C,E), (B,D), (D,F), (E,F)	3	A—C—D—E, C—B, D—E

Q.)



Solution :-

Vertices Covered	Edges taken	Cost	Updated MST
(1) →	~~(1,2)~~, (1,4), (1,5).	10	1—2
(1,2) →	(1,4), (1,5), (2,3), (2,5), ~~(2,6)~~	25	1—2—6
(1,2,6) →	(1,4), (1,5), (2,3), (2,5), (6,4), (6,5), ~~(6,3)~~	15	1—2—6—3
(1,2,6,3) →	(1,4), (1,5), (2,3), (2,5), ~~(6,4)~~, (6,5), (3,5)	20	1—2—6—3, 6—4
(1,2,6,3,4) →	(1,4), (1,5), (2,3), (2,5), (6,5), ~~(3,5)~~, ↳ chose this in order to avoid looping	35	1—2—6—3, 6—4, 3—5

∴ Total cost => 105

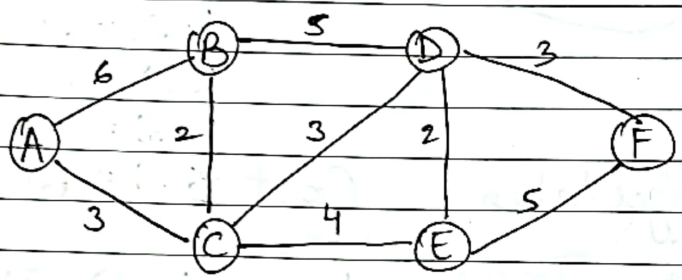
Again initially we remove all loops & parallel edges.

PAGE No.
DATE / /

* MST using Kruskal's Algorithm :-

☆ Step 1 :- Sort all edges into non-decreasing order
Step 2 :- Add the next smallest weighted edge to the first if it will not cause
Step 3 :- Stop if (n-1) edges, or go to step 2.

Q.)

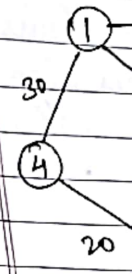


Sorting :- DE = 2, BC = 2, CD = 3, AC = 3, DF = 3, CE = 4, BD = 5, EF = 5, AB = 6

Vertices Covered	Edges Taken	Cost	Updated MST
(D) →	(D, E)	2	D — E
(D, E, B, C) →	(B, C)	2	B C D E
	(C, D)	3	B C D E
(D, E, B, C, A) →	(A, C)	3	A B C D E
(D, E, B, C, A, F) →	(D, F)	3	A B C D E F

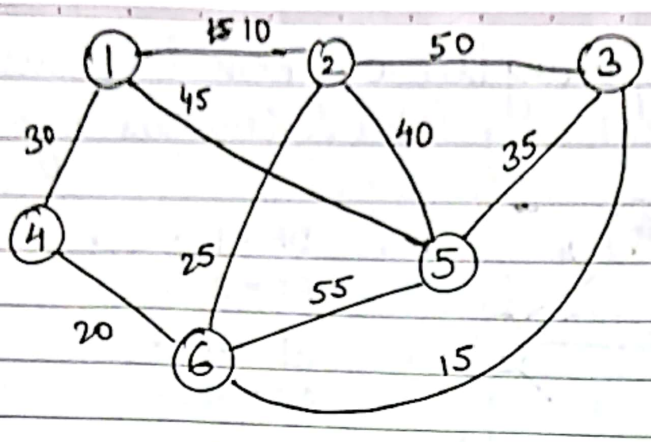
Final ans as we have (n-1) edges

Q.)



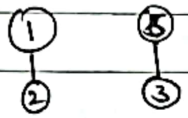
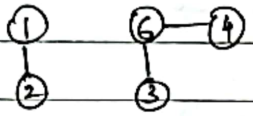
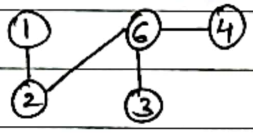
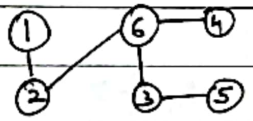
Vertices Covered
(1) →
(1, 2, 6, 3) →
(1, 2, 6, 3, 4) →
(1, 2, 6, 3, 4, 5)

Q.)



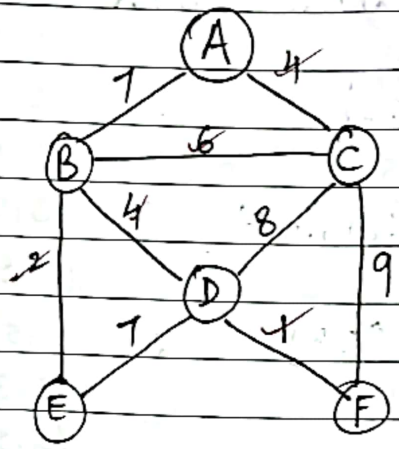
Sort:

(1,2) = 10
(6,3) = 15
(4,6) = 20
(2,6) = 25
(1,4) = 30
(3,5) = 35
(2,5) = 40
(1,5) = 45
(2,3) = 50
(6,5) = 55

Vertices Covered	Edges taken	Cost	Updated MST.
(1) →	(1,2)	10	
(1,2,6,3) →	(6,3)	15	
(1,2,6,3,4) →	(4,6)	20	
	(2,6)	25	
(1,2,6,3,4,5) →	(3,5)	35	

Final ans. (n-1) edges

Q.) Explain what is a spanning tree with example. Consider the following graph & find MST using Prims & Kruskal's algo.



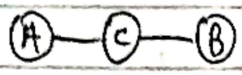
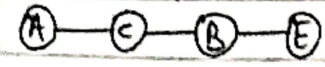
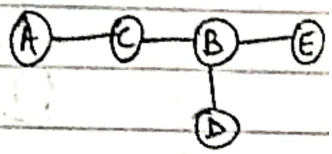
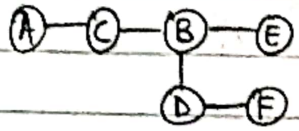
DF = 1 ✓
BE = 2 ✓
BD = 4 ✓
AC = 4 ✓
BC = 6
AD = 7
ED = 7
CD = 8
CF = 9

∴ Kruskal's

Vertices Covered	Edge taken	Cost	Updated MST
(D) →	(D, F)	1	D — F
(D, F) →	(B, E)	2	D F B E
(D, F, B, E) →	(B, D)	4	D F B E
	(A, C)	4	A C D F B E
(D, F, B, E, A, C) →	(B, C)	6	D F B E C A

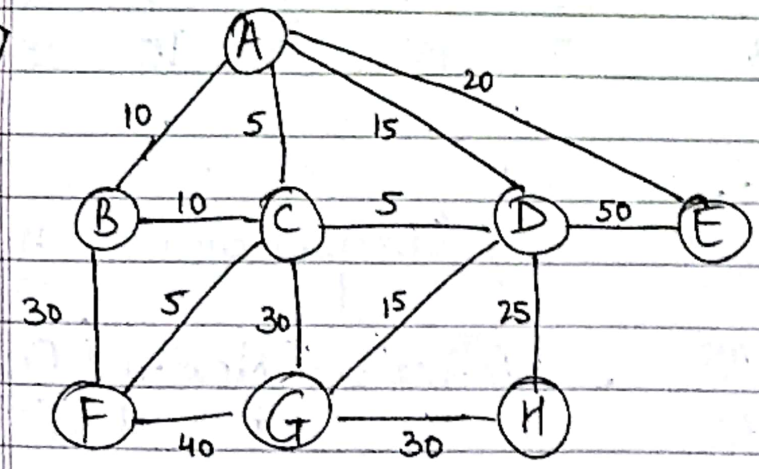
∴ Total Cost = 17

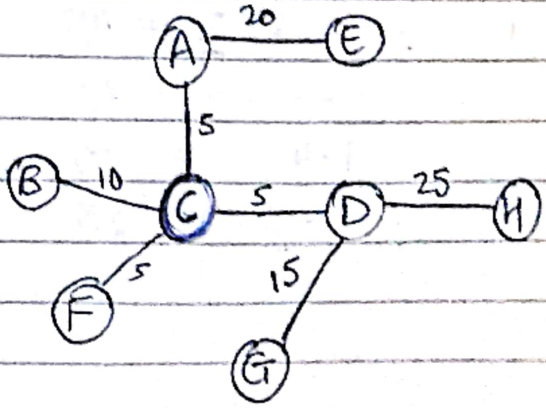
Prims

Vertices Covered	Edges Taken	Cost	Updated MST
(A) →	(A,B), (A,C)	4	
(A,C) →	(A,B), (C,B), (C,D), (C,F)	6	
(A,C,B) →	(A,B), (C,D), (C,F) (B,D), (B,E)	2	
(A,C,B,E) →	(A,B), (C,D), (C,F) (B,D), (E,D)	4	
(A,C,B,E,D) →	(A,B), (C,D), (C,F) (E,D), (D,F)	1	

∴ Total Cost = 17

Q.)





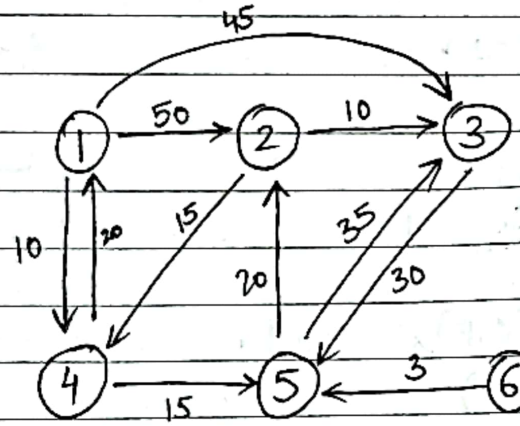
By Kruskals

PAGE No.
DATE / /

* Single Source Shortest Path :-

Assumption :- All weights are positive
Shortest path betn v_0 & some other node v is an ordering among a subset of edges.

Eg]



We always select the vertex of least weighted edge and then proceed.

initial vertex

Iteration		Vertex Selected	(1)	(2)	(3)	(4)	(5)	(6)
		1	0	∞	∞	∞	∞	∞
1	(1)	4	0	50	45	(10)	∞	∞
2	(1,4)	5	0	50	45	10	(25)	∞
3	(1,4,5)	2	0	(45) (10+15+20)	45	10	25	∞
4	(1,4,5,2)	3	0	45	(45)	10	25	∞
5	(1,4,5,2,3)							

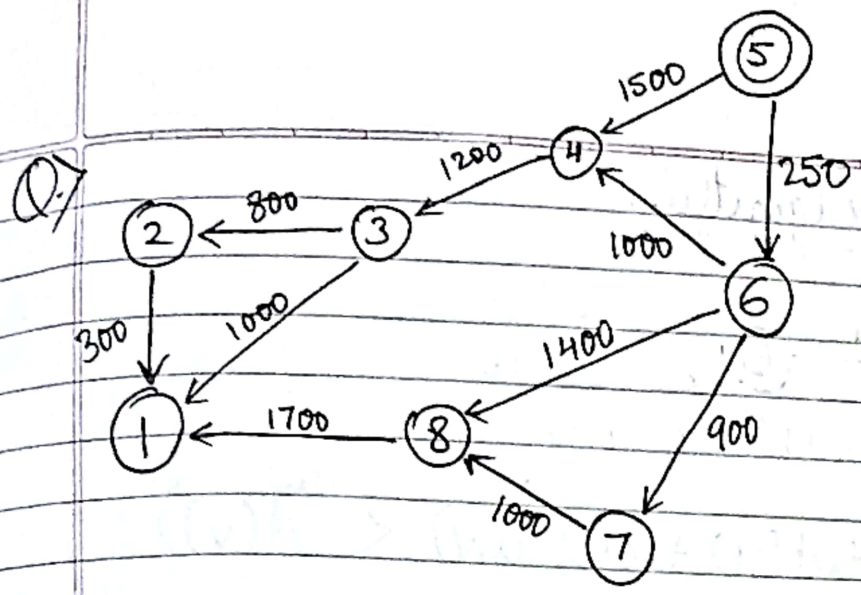
Shortest Path

1-4-5-2 = 45
1-3 = 45
1-4 = 10
1-4-5 = 25

∴ Comparisons → $n(n-1)$

∴ Time complexity : $O(n^2)$

Q:



Make one adjacency matrix initially

vertex with shortest dist. is selected.

Soln.

Iteration		Vertex Selected	1	2	3	4	(5)	6	7	8
			∞	∞	∞	∞	0	∞	∞	∞
1	(5)	6	"	"	"	1500	0	(250)	"	"
2	(5,6)	7	"	"	"	1250	0	"	(1150)	1650
3	(5,6,7)	4	"	"	"	(1250)	0	"	"	1650
4	(5,6,7,4)	8	"	"	2450	"	0	"	"	(1650)
5	(5,6,7,4,8)	3	3350	"	(2450)	"	"	"	"	"
6	(5,6,7,4,8,3)	2	3350	3250						
7	(5,6,7,4,8,3,2)	1	3350	3250	2450	1250	0	250	1150	1650

∴ In each iteration, ignore the already chosen vertex and then select the one having the least dist.

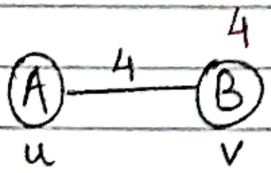
∴ Step 1 :- Begin with source vertex & select it.

Step 2 :- Update dist. of all nearby vertices ~~keeping~~ (see dijkstra's formula)

Step 3 :- Select the one vertex with least dist. ~~& repeat go to~~

Step 4 :- Go to step 2, until all vertices are selected once.

* Dijkstra's Algorithm :



if (dist(u) + cost(u,v) < dist(v))

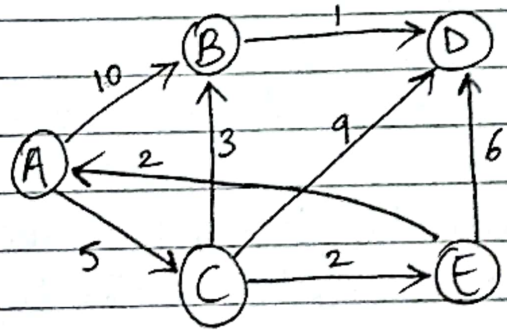
0 + 4 < ∞

4 < ∞.

∴ update ∞ to 4 → only of those nodes which are not visited

```
{
  d[v] = dist[u] + cost[u,v];
}
```

Q.)



Source Vertex = A

	Vertex Selected	A	B	C	D	E
	A	0	∞	∞	∞	∞
(A)	C	0	10	5	∞	∞
(A,C)	E	0	8	5	14	7
(A,C,E)	B	0	8	5	13	7
(A,C,E,B)	D	0	8	5	9	7

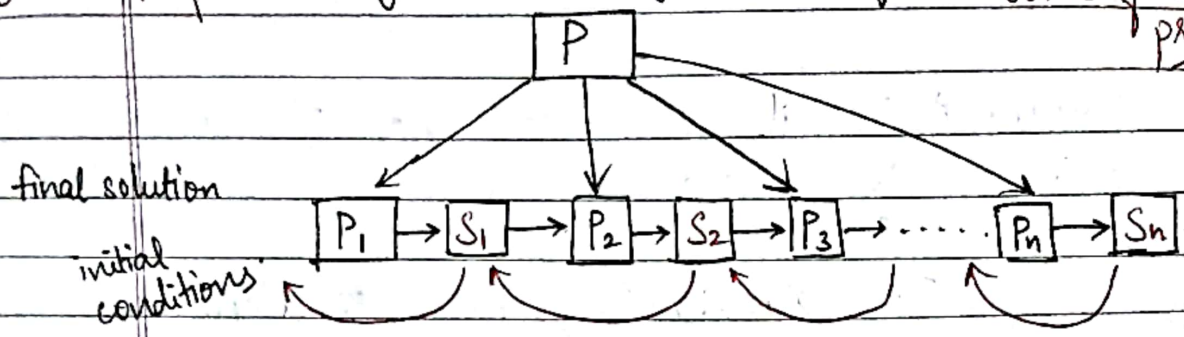
→ only vertices not in the adjacent set are selected

sample ques on last page

Dynamic Programming

Used to solve optimisation probs.

Uses Principle of Optimality → optimal soln to a prob is combination of optimal solns to some of its sub problems.



* Steps of Dynamic Programming :-

1] Characterize optimal ~~solution~~ substructure
2] Recursively define the value of an optimal soln.
3] Compute the value bottom up.
4] (optional) Construct an optimal soln.

* Divide & Conquer	Dynamic Programming
→ Problems broken into sub-problems and solved individually	→ Problems are broken into subproble of which some are common.
→ Subproblems are independent.	→ Subproblems are dependent on ou another.
→ Simpler	→ Complex
→ Any use case	→ Optimization

DP Approach

- General Method
- Multistage Graphs
- All pair shortest path
- Single source shortest path
- 0/1 knapsack
- Travelling Salesman Problem
- Matrix Chain Multiplication

* 0/1 Knapsack.

It means that an item is either fully accepted or rejected.
A portion of item is NOT accepted

∴ 0 → fully rejected , 1 → fully accepted.

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max[B[k-1,w], B[k-1,w-w_k] + b_k] & \text{for others} \end{cases}$$

very imp formula

∴ k = current item, w = weight. max capacity = 5

Eg]

i/w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	(7)

till max. capacity of bag

no. of items

change when second is 0 added.

// Initialise the best cases.

Initialise First row & First Col. to zero

```
for w=0 to w
    B[0,w] = 0

for i=1 to n
    B[i,0] = 0
```

Profit = 7

Items (w, p)

weight, profit.

1 (2, 3)
2 (3, 4)
3 (4, 5)
4 (5, 6)

Optimal Substructure

- Problem → several sub probs.
- Opt soln. to each subprob combined to form soln. of original prob.
- If opt. soln. contains opt. sub soln., then optimal substructure is exhibited.

Overlapping Subprobs.

- When recursive algo. revisits same sub prob over and over, then problem is said to have overlapping subprobs.

rejected.

ted.

] for others

ty = 5

Eg]

i/W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	(7)

no. of items

change when second is added.

// Initialise the best cases.

Initialise First row & First Col to zero

```
for w=0 to w
    B[0,w] = 0
for i=1 to n
```

Profit = 7

Items (w, v)

weight

profit.

1 (2, 3)

	1	2	3	4
Vi	3	4	5	6
Wi	2	3	4	5

2nd object → (W=5-3) = 2

(2,-2) = 0

Soln. → (1,1,0,0) = 7

(2,1) = 7

Complexity :. $T(n) = O(n \times w)$

If $w_i \leq w$ // Item i can be in the soln.

if $v_i + B[i-1, w-w_i] > B[i-1, w]$

$B[i,w] = v_i + B[i-1, w-w_i]$

else

$B[i,w] = B[i-1, w]$

else $B[i,w] = B[i-1,w]$ // $w_i > w$

Q.] W = 10

∴ Items:

	1	2	3	4
v_i	10	40	30	50
w_i	5	4	6	3

i \ W	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	10	10	10	10	10
2	0	0	0	0	40	40	40	(40)	40	50
3	0	0	0	0	40	40	40	40	40	50
4	0	0	0	50	50	50	50	90	90	90

Profit = 90

(10-3) = 7

7-4 = 3

∴ (0,1,0,1) = 40+50 = 90

Imp Note:

Let $i = n$ & $k = w$

if $B[i,k] \neq B[i-1,k]$ then

mark the i^{th} item as in the knapsack

$i = i-1$; $k = k-w_i$;

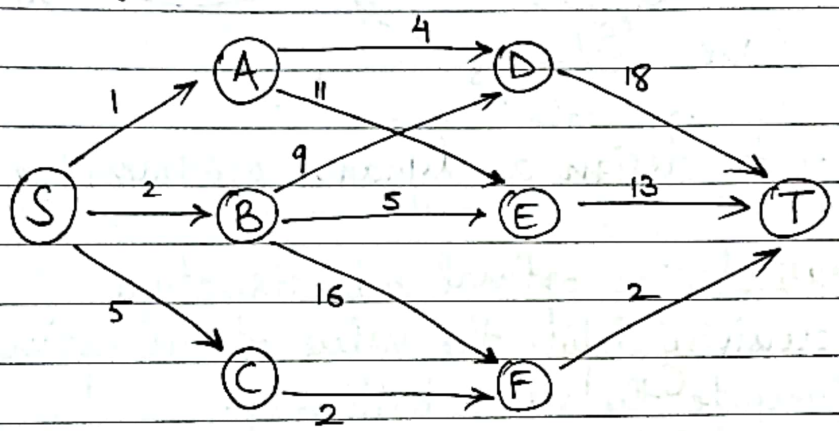
else

$i = i-1$ // Assume the i^{th} item is not in knapsack

---

* Multistage Graphs :-

Eg]

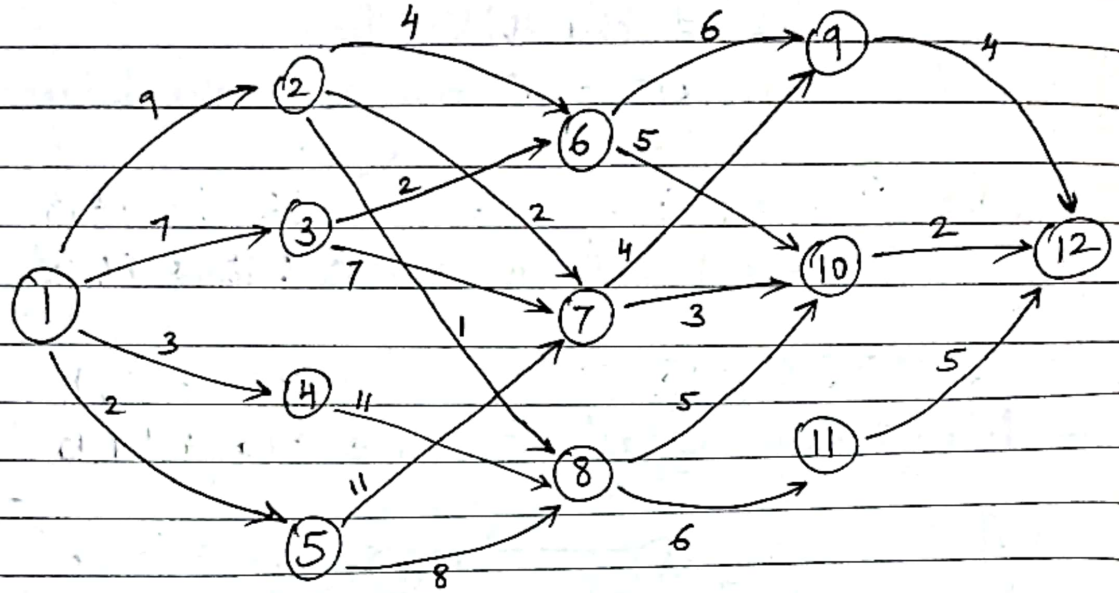


∴ Greedy gives ⟶ 1+4+18 = 23

Meanwhile, actual shortest path is 5+2+2 = 9

∴ Ques. P.T.O

Q.) V_1 V_2 V_3 V_4 V_5



Steps to design a dynamic programming algorithm :-

1] Characterize optimal sub-structure
2] Recursively define the value of an optimal soln.
3] Compute the value bottom up.
4] Construct an optimal soln.

Recursive Formula :-

a] Forward :-

$cost(i,j)$ = cost of path $p(i,j)$

$$cost(i,j) = \min_{l \in V_{i+1}} \{ c(j,l) + cost(i+1,l) \}$$

$c(j,l)$ → weight of path betn j & l.

stage → i ; vertex in stage → j

$l \in V_{i+1}$ → $i+1^{th}$ vertex.

$\langle j,l \rangle \in E$ → edges ka set.

b] Backward :-

$$bcost(i,j) = \min_{l \in V_{i-1},\ \langle j,l \rangle \in E} \{ bcost(i-1,l) + c(l,j) \}$$

Stage 4:

$cost(i,j) = cost(4,9) = 4$

$cost(4,10) = 2$

$cost(4,11) = 5$

stage 3:

$cost(i,j) = cost(3,6) = \min \begin{cases} c(6,9) + (4,9) \\ c(6,10) + (4,10) \end{cases} = 10, 7$ ✓

$cost(i,j) = cost(3,7) = \min \begin{cases} c(7,9) + (4,9) \\ c(7,10) + (4,10) \end{cases} = 8, 5$ ✓

$cost(i,j) = cost(3,8) = \min \begin{cases} c(8,10) + (4,10) \\ c(8,11) + (4,11) \end{cases} = 7, 11$ ✓

stage 2:

$cost(i,j) = cost(2,2) = \min \begin{cases} c(2,6) + (3,6) \\ c(2,7) + (3,7) \\ c(2,8) + (3,8) \end{cases} = \begin{cases} 14 \text{ or } 11 \\ 10 \text{ or } 7 \\ 8 \text{ or } 12 \end{cases}$ ✓

$cost(i,j) = cost(2,3) = \min \begin{cases} c(3,6) + (3,6) \\ c(3,7) + (3,7) \end{cases} = \begin{cases} 12 \text{ or } 9 \\ 15 \text{ or } 12 \end{cases}$ ✓

$cost(i,j) = cost(2,4) = \min \{ c(4,8) + (3,8) \} = 18 \text{ or } 22$

$cost(i,j) = cost(2,5) = \min \begin{cases} c(5,7) + (3,7) \\ c(5,8) + (3,8) \end{cases} = \begin{cases} 19 \text{ or } 16 \\ 15 \text{ or } 19 \end{cases}$

$cost(i,j) = cost(1,2) = \min \begin{cases} c(1,2) + (2,2) \\ c(1,3) + (2,3) \\ c(1,4) + (2,4) \\ c(1,5) + (2,5) \end{cases} = \begin{cases} 9+7 \\ 7+9 \\ 3+18 \\ 2+15 \end{cases}$

∴ Least Cost = 16 & optimal paths

PAGE No.
DATE / /

∴ Path $(1) \xrightarrow{9} (2) \xrightarrow{2} (7) \xrightarrow{3} (10) \xrightarrow{2} (12)$

$d(3,6) = 10$
$d(3,7) = 10$
$d(3,8) = 10$
$d(2,2) = 7$
$d(2,3) = 6$
$d(2,4) = 8$
$d(2,5) = 8$
$d(1,1) = 2, 3$

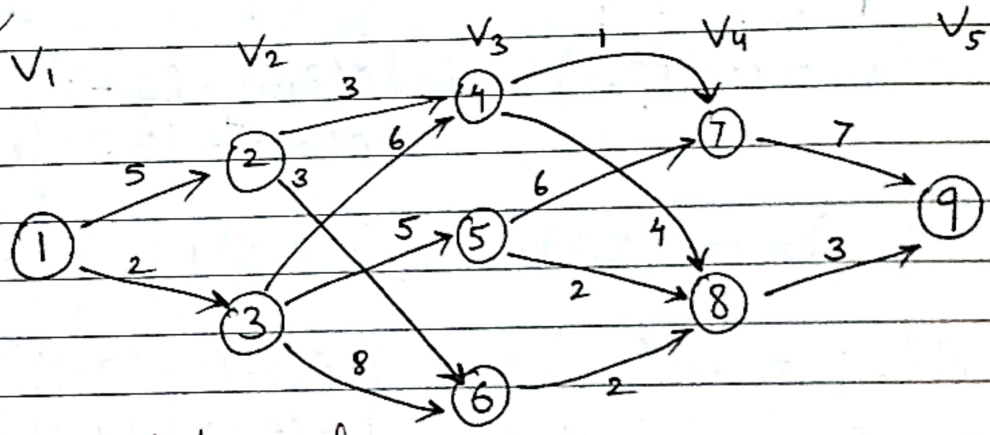
$V_2 = d(1,1) = 2$
$V_3 = d(2, d(1,1)) = d(2,2) = 7$
$V_4 = d(3, d(2,2)) = d(3,7) = 10$

$(1) \xrightarrow{9} (2) \xrightarrow{2} (7) \xrightarrow{3} (10) \xrightarrow{2} (12)$

* Multistage :- backward approach

$$\text{bcost}(i,j) = \min \{ \text{bcost}((i-1), l) + c(l,j) \}$$
$$l \in V_{i-1} \quad \langle j, l \rangle \in E$$

Q.)



Soln: Forward Approach

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

$$\text{cost}(3,5) = \min \begin{bmatrix} \text{cost}(5,7) + \text{cost}(4,7) \\ \text{cost}(5,8) + \text{cost}(4,8) \end{bmatrix} = 13 = 5$$

$$\text{cost}(3,6) = \min [\text{cost}(6,8) + \text{cost}(4,8)] = 5$$

$$cost(2,2) = \min \begin{bmatrix} cost(2,4) + cost(3,4) \\ cost(2,6) + cost(3,6) \end{bmatrix} = \begin{matrix} 10 \\ (8) \end{matrix}$$

$$cost(2,3) = \min \begin{bmatrix} cost(3,4) + cost(3,4) \\ cost(3,5) + cost(3,5) \\ cost(3,6) + cost(3,6) \end{bmatrix} = \begin{matrix} 13 \\ (10) \\ 13 \end{matrix}$$

$$cost(1,1) = \min \begin{bmatrix} cost(1,2) + cost(2,2) \\ cost(1,3) + cost(2,3) \end{bmatrix} = \begin{matrix} 13 \\ (12) \end{matrix}$$

$\therefore$ **Least Cost = 12**

Backtracking :-

Path (1) – (3) – (5) – (8) – (9)

$d(4,7) = 9$

$d(3,4) = 8$

$d(3,5) = 8$

$d(3,6) = 8$

$d(2,2) = 6$

$d(2,3) = 5$

$d(1,1) = 3$

$V_2 = d(1,1) = 3$

$V_3 = d(2,3) = 5$

$V_4 = d(3,5) = 8$

$V_5 = d(4,8) = 9$

Backward Approach.

$$bcost(i,j) = \min_{l \in V_{i-1},\ \langle j,l \rangle \in E} \{ bcost(i-1,l) + c(l,j) \}$$

$\therefore bcost(1,1) = \phi$

$bcost(2,2) = 5$

$bcost(2,3) = 2$

PAGE No.
DATE / /

$$bcost(3,4) = \min \begin{bmatrix} bcost(2,2) + c(4,2) \\ bcost(2,3) + c(4,3) \end{bmatrix} = 8 \\ = (8)$$

$$bcost(3,5) = \min [bcost(3,5) + c(5,3)] = (7)$$

$$bcost(3,6) = \min \begin{bmatrix} cost(2,3) + c(3,6) \\ cost(2,2) + c(2,6) \end{bmatrix} = 10 \\ = (8)$$

$$bcost(4,7) = \min \begin{bmatrix} cost(3,4) + c(4,7) \\ cost(3,5) + c(5,7) \end{bmatrix} = 9 \\ = (9)$$

$$bcost(4,8) = \min \begin{bmatrix} cost(3,4) + c(4,8) \\ cost(3,5) + c(5,8) \\ cost(3,6) + c(6,8) \end{bmatrix} = 12 \\ = (9) \\ = 10$$

$$bcost(5,9) = \min \begin{bmatrix} cost(4,7) + c(7,9) \\ cost(4,8) + c(8,9) \end{bmatrix} = 16 \\ = (12)$$

Time Complexity → $O(N^3)$

Through memoization

* Matrix Chain Multiplication :-

Recursive Formula :-

$$m[i,j] = 0 \qquad i = j$$

$$m[i,j]_{min} = (m[i,k] + m[k+1,j] + P_{i-1} P_k P_j) \qquad i \leq k \leq j$$

P → no. of columns (based on).

Q.) Apply matrix chain multiplication

I/P : <5, 8, 10, 3, 6, 11, 12>

$A_1 = 5 \times 8$ $\quad P_0 = 5$

$A_2 = 8 \times 10$ $\quad P_1 = 8$

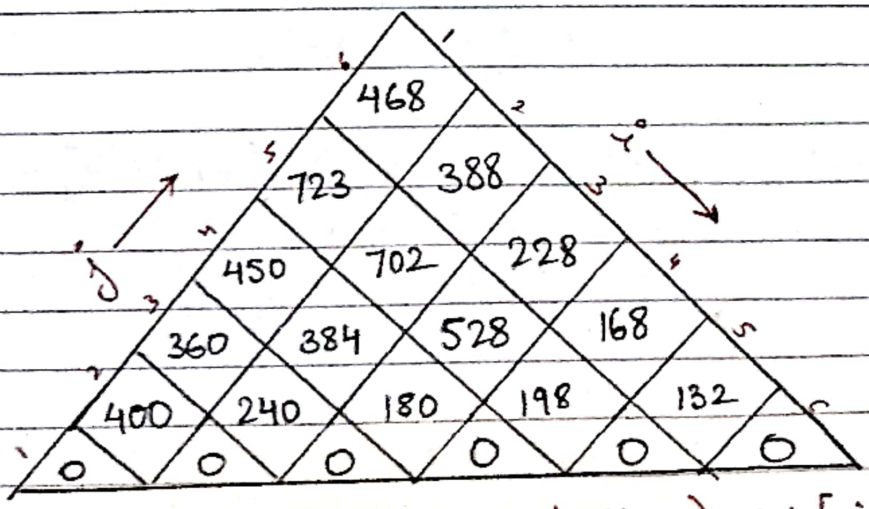
$A_3 = 10 \times 3$ $\quad P_2 = 10$

$A_4 = 3 \times 6$ $\quad P_3 = 3$

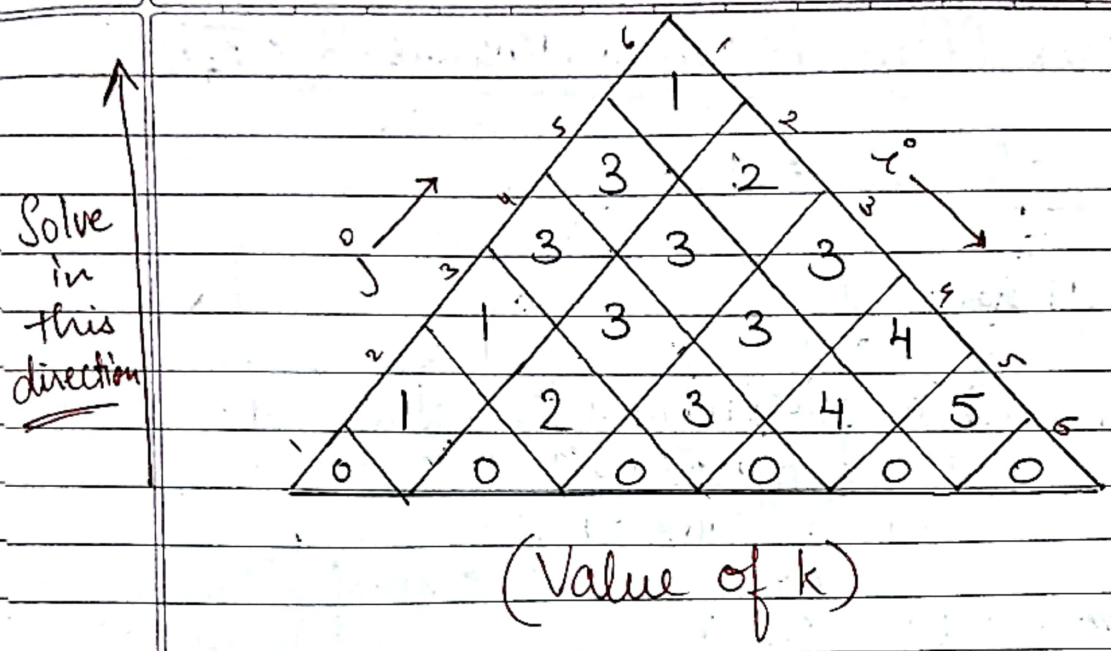
$A_5 = 6 \times 11$ $\quad P_4 = 6$

$A_6 = 11 \times 12$ $\quad P_5 = 11$

$P_6 = 12$



(No. of computations) M[i,j]



Iteration I :-

$i=1, j=2;$

$$m[1,2] = \min_{i \le k < j} \{ m[1,1] + m[2,2] + P_0 P_1 P_2 \}$$

$k=1$

$$= [0 + 0 + 5 \times 8 \times 10]$$

$$= 400 \quad (k=1)$$

$i=2, j=3$ $\quad m[2,3] = \min \{ m[2,2] + m[3,3] + P_1 P_2 P_3 \}$

$i \le k < j \quad \therefore k=2$

$$= [0 + 0 + 8 \times 10 \times 3]$$

$$= 240 \quad (k=2)$$

$i=3, j=4$ $\quad m[3,4] = \min \{ m[3,3] + m[4,4] + P_2 P_3 P_4 \}$

$i \le k < j \quad \therefore k=3$

$$= [0 + 0 + 10 \times 3 \times 6]$$

$$= 180 \quad (k=3)$$

$i=4, j=5$ $\quad m[4,5] = \min \{ m[4,4] + m[5,5] + P_3 P_4 P_5 \}$

$\therefore k=4 \quad = [0 + 0 + 3 \times 6 \times 11]$

$$= 198 \quad (k=4)$$

$i=5, j=6 \quad m[5,6] = \min\{m[5,5] + m[6,6] + P_4 P_5 P_6\}$

$k=5 \hookrightarrow = [0 + 0 + 6 \times 11 \times 2]$

$= \underline{132} \quad (k=5)$

Iteration II :-

$k=1,2$

$i=1, j=3 \quad m[1,3] = \min\begin{Bmatrix} m[1,1] + m[2,3] + P_0 P_1 P_3 \\ m[1,2] + m[3,3] + P_0 P_2 P_3 \end{Bmatrix}$

$= \min\begin{Bmatrix} [0 + 240 + 5 \times 8 \times 3] = 360 \\ [400 + 0 + 5 \times 10 \times 3] = 550 \end{Bmatrix}$

$= \underline{360} \quad (k=1)$

$k=2,3$

$i=2, j=4 \quad m[2,4] = \min\begin{Bmatrix} m[2,2] + m[3,4] + P_1 \cdot P_2 P_4 \\ m[2,3] + m[4,4] + P_1 P_3 P_4 \end{Bmatrix}$

$= \min\begin{Bmatrix} [0 + 180 + 8 \cdot 10 \cdot 6] = 660 \\ [240 + 0 + 8 \cdot 3 \cdot 6] = 384 \end{Bmatrix}$

$= \underline{384} \quad (K=3)$

$k=3,4$

$i=3, j=5 \quad m[3,5] = \min\begin{Bmatrix} m[3,3] + m[4,5] + P_2 P_3 P_5 \\ m[3,4] + m[5,5] + P_2 P_4 P_5 \end{Bmatrix}$

$= \min\begin{Bmatrix} [0 + 198 + 10 \cdot 3 \cdot 11] = 528 \\ [180 + 0 + 10 \cdot 6 \cdot 11] = 840 \end{Bmatrix}$

$= \underline{528} \quad (K=3)$

$k=4,5$

$i=4, j=6 \quad m[4,6] = \min\begin{Bmatrix} m[4,4] + m[5,6] + P_3 \cdot P_4 \cdot P_6 \\ m[4,5] + m[6,6] + P_3 P_5 P_6 \end{Bmatrix}$

$= \min\begin{Bmatrix} [0 + 132 + 36] = 168 \\ [198 + 0 + 66] = 264 \end{Bmatrix}$

$= \underline{168} \quad (K=4)$

PAGE No.
DATE / /

Iteration III :-

$i=1, j=4$ $\quad m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + P_0 P_1 P_4 \\ m[1,2] + m[3,4] + P_0 P_2 P_4 \\ m[1,3] + m[4,4] + P_0 P_3 P_4 \end{cases}$

$K = 1, 2, 3$

$= \min \begin{cases} [0 + 384 + 5 \cdot 8 \cdot 6] = 624 \\ [400 + 180 + 5 \cdot 10 \cdot 6] = 880 \\ [360 + 0 + 5 \cdot 3 \cdot 6] = 450 \end{cases}$

$= 450 \quad (k=3)$

$i=2, j=5$ $\quad m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + P_1 P_2 P_5 \\ m[2,3] + m[4,5] + P_1 P_3 P_5 \\ m[2,4] + m[5,5] + P_1 P_4 P_5 \end{cases}$

$K = 2, 3, 4$

$= \min \begin{cases} [0 + 528 + 8 \cdot 10 \cdot 11] = 1408 \\ [240 + 198 + 8 \cdot 3 \cdot 11] = 702 \\ [384 + 0 + 8 \cdot 6 \cdot 11] = 912 \end{cases}$

$= 702 \quad (k=3)$

$i=3, j=6$ $\quad m[3,6] = \min \begin{cases} m[3,3] + m[4,6] + P_2 P_3 P_6 \\ m[3,4] + m[5,6] + P_2 P_4 P_6 \\ m[3,5] + m[6,6] + P_2 P_5 P_6 \end{cases}$

$K = 3, 4, 5$

$= \min \begin{cases} [0 + 168 + 10 \cdot 3 \cdot 2] = 228 \\ [180 + 132 + 10 \cdot 6 \cdot 2] = 432 \\ [528 + 0 + 10 \cdot 11 \cdot 2] = 748 \end{cases}$

$= 228 \quad (k=3)$

Iteration IV :-

$i=1, j=5$ $\quad m[1,5] = \min_{k=1,2,3,4} \begin{cases} m[1,1] + m[2,5] + P_0 P_1 P_5 \\ m[1,2] + m[3,5] + P_0 P_2 P_5 \\ m[1,3] + m[4,5] + P_0 P_3 P_5 \\ m[1,4] + m[5,5] + P_0 P_4 P_5 \end{cases}$

$= \min \begin{cases} [0 + 702 + 5 \cdot 8 \cdot 11] = 1142 \\ [400 + 528 + 5 \cdot 10 \cdot 11] = 1478 \\ [360 + 198 + 5 \cdot 3 \cdot 11] = 723 \\ [450 + 0 + 5 \cdot 6 \cdot 11] = 780 \end{cases}$

$= \underline{723} \quad (k=3)$

$i=2, j=6$ $\quad m[2,6] = \min \begin{cases} m[2,2] + m[3,6] + P_1 P_2 P_6 \\ m[2,3] + m[4,6] + P_1 P_3 P_6 \\ m[2,4] + m[5,6] + P_1 P_4 P_6 \\ m[2,5] + m[6,6] + P_1 P_5 P_6 \end{cases}$

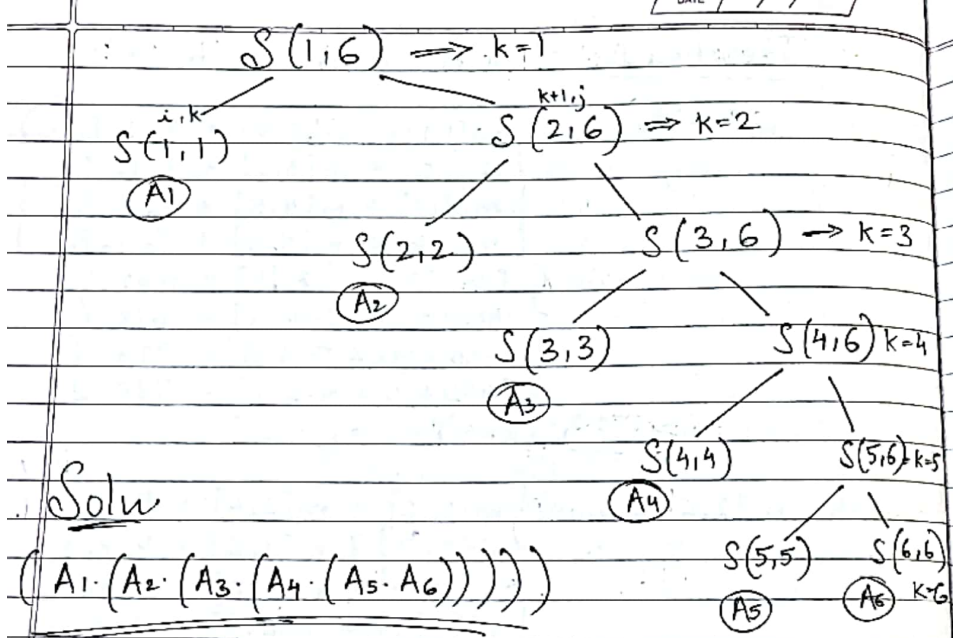
$= \min \begin{cases} [0 + 228 + 8 \cdot 10 \cdot 2] = 388 \\ [240 + 168 + 8 \cdot 3 \cdot 2] = \\ [384 + 132 + 8 \cdot 6 \cdot 2] = \\ [702 + 0 + 8 \cdot 11 \cdot 2] = \end{cases}$

$= \underline{388} \quad (k=2)$

Iteration V :-

$i=1, j=6$ $\quad m[1,6] = \min \begin{cases} m[1,1] + m[2,6] + P_0 P_1 P_6 \\ m[1,2] + m[3,6] + P_0 P_2 P_6 \\ m[1,3] + m[4,6] + P_0 P_3 P_6 \\ m[1,4] + m[5,6] + P_0 P_4 P_6 \\ m[1,5] + m[6,6] + P_0 P_5 P_6 \end{cases}$

$= \underline{468} \quad (k=1)$



Solu

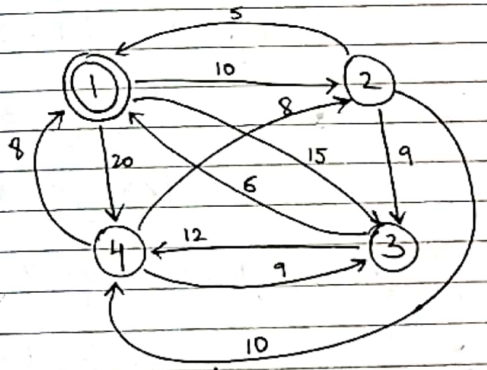
$(A_1 \cdot (A_2 \cdot (A_3 \cdot (A_4 \cdot (A_5 \cdot A_6)))))$

Start from any vertex, go through all vertices once and return to starting index & COST should be minimum

* Travelling Salesman Problem (TSP) :-

(C) M =

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0



S → Adjacent vertic

Represents set of vertices the path goes thro

Step 1: $g(i, \phi) = C_{i1}$ $\quad 1 \le i \le n$

Step 2: $g(i, S) = \min_{j \in S} \{ C_{ij} + g(j, S-$

(i: starting, S: going through)

Step 1: $g(2, \phi) = C_{21} = 5$ $\quad |S|=1$

$g(3, \phi) = C_{31} = 6$

$g(4, \phi) = C_{41} = 8$

Step 2: $g(2, \{3\}) = \min_{j \in S} \{ C_{23} + g(3, \phi) \}$ $\quad |S|=1$

$g(2, \{4\}) = \min_{j \in S} \{ C_{24} + g(4, \phi) \}$

$g(3, \{2\}) = \min_{j \in S} \{ C_{32} + g(2, \phi) \}$

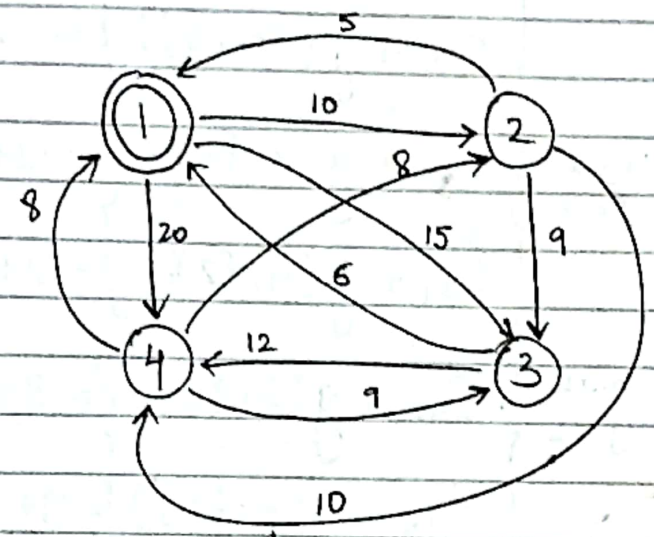
$g(3, \{4\}) = \min_{j \in S} \{ C_{34} + g(4, \phi) \}$

Start from any vertex, go through all vertices once and return to starting index & COST should be minimum

* Travelling Salesman Problem (TSP) :-

(C) M =

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0



S → Adjacent vertices.

↓

Represents set of vertices the path can goes through.

Step 1: $g(i, \phi) = C_{i1}$ $\quad 1 \le i \le n$

Step 2: $g(i, S) = \min_{j \in S} \{ C_{ij} + g(j, S - \{j\}) \}$

i → starting, S → Going through

Step 1: $g(2, \phi) = C_{21} = 5$

$g(3, \phi) = C_{31} = 6$

$g(4, \phi) = C_{41} = 8$

$|S| = 1$ Every |S| has their own step 2

Step 2: $g(2, \{3\}) = \min_{j \in S} \{ C_{23} + g(3, \phi) \} = 6 + 9 = 15$

$g(2, \{4\}) = \min_{j \in S} \{ C_{24} + g(4, \phi) \} = 10 + 8 = 18$

$g(3, \{2\}) = \min_{j \in S} \{ C_{32} + g(2, \phi) \} = 13 + 5 = 18$

$g(3, \{4\}) = \min_{j \in S} \{ C_{34} + g(4, \phi) \} = 12 + 8 = 20$

PAGE No.
DATE / /

$|S|=1$

$g(4,\{2\}) = \min_{j \in S}\{C_{42} + g\{2,\phi\}\} = 8+5 = 13$

$g(4,\{3\}) = \min_{j \in S}\{C_{43} + g\{3,\phi\}\} = 9+6 = 15$

$|S|=2$

$g(2,\{3,4\}) = \min_{j \in S}\{C_{23} + g(3,\{4\}),\ C_{24} + g(4,\{3\})\}$
$= 9+20 = 29$
$= 10+15 = 25$

$g(3,\{2,4\}) = \min_{j \in S}\{C_{32} + g(2,\{4\}),\ C_{34} + g(4,\{2\})\}$
$= 13+18 = 31$
$= 12+13 = 25$ ✓

$g(4,\{2,3\}) = \min_{j \in S}\{C_{42} + g(2,\{3\}),\ C_{43} + g(3,\{2\})\}$
$= 8+15 = 23$
$= 9+18 = 27$

$|S|=3$

$g(1,\{2,3,4\}) = \min\{C_{12} + g(2,\{3,4\}),\ C_{13} + g(3,\{2,4\}),\ C_{14} + g(4,\{2,3\})\}$
$= 10+25 = 35$
$= 15+25 = 40$
$= 20+23 = 43$

$\therefore$ (1) → (2) → (4) → (3) → (1)

$g(1,\{2,3,4\}) = 2 \quad (35)$

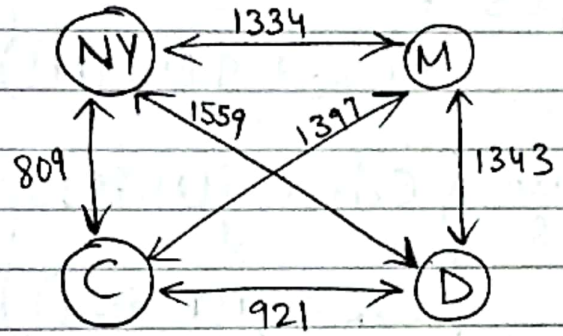
$g(2,\{3,4\}) = 4 \quad (25)$

$g(4,\{3\}) = 3 \quad (15)$

Recursive Formula → $g(i, S) = \min_{j \in S} \{ C_{ij} + g(j, \{S\} - \{j\}) \}$

Q.) Cities = { New York, Miami, Dallas, Chicago }

$$C = \begin{array}{c} \\ NY \\ M \\ D \\ C \end{array} \begin{array}{c} \begin{array}{cccc} NY & M & D & C \end{array} \\ \begin{bmatrix} 0 & 1334 & 1559 & 809 \\ 1334 & 0 & 1343 & 1397 \\ 1559 & 1343 & 0 & 921 \\ 809 & 1397 & 921 & 0 \end{bmatrix} \end{array}$$



Step 1: $g(M, \phi) = C_{MNY} = 1334$

$g(D, \phi) = C_{DNY} = 1559$

$g(C, \phi) = C_{CNY} = 809$

Step 2: $|S| = 1$

$g(M, \{D\}) = \min_{j \in S} \{ C_{MD} + g(D, \phi) \} = 1343 + 1559 = 2902$

$g(M, \{C\}) = \min_{j \in S} \{ C_{MC} + g(C, \phi) \} = 1397 + 809 = 2206$

$g(D, \{M\}) = \min_{j \in S} \{ C_{DM} + g(M, \phi) \} = 1343 + 1334 = 2677$

$g(D, \{C\}) = \min_{j \in S} \{ C_{DC} + g(C, \phi) \} = 921 + 809 = 1730$

$g(C, \{M\}) = \min_{j \in S} \{ C_{CM} + g(M, \phi) \} = 1397 + 1334 = 2731$

$g(C, \{D\}) = \min_{j \in S} \{ C_{CD} + g(D, \phi) \} = 921 + 1559 = 2480$

PAGE No.
DATE / /

$|S| = 2$

$$g(M, \{D, C\}) = \min_{j \in S} \begin{cases} C_{MD} + g(D, \{C\}) \\ C_{MC} + g(C, \{D\}) \end{cases} = \begin{cases} 1343 + 1730 = 3073 \\ 1397 + 2480 = 3877 \end{cases}$$

$$g(D, \{M, C\}) = \min_{j \in S} \begin{cases} C_{DM} + g(M, \{C\}) \\ C_{DC} + g(C, \{M\}) \end{cases} = \begin{cases} 1343 + 2206 = 3549 \\ 921 + 2731 = 3652 \end{cases}$$

$$g(C, \{M, D\}) = \min_{j \in S} \begin{cases} C_{CM} + g(M, \{D\}) \\ C_{CD} + g(D, \{M\}) \end{cases} = \begin{cases} 1397 + 2902 = 4299 \\ 921 + 2477 = 3398 \end{cases}$$

$|S| = 3$

$$g(NY, \{M, D, C\}) = \min_{j \in S} \begin{cases} C_{NYM} + g(M, \{D, C\}) \\ C_{NYD} + g(D, \{M, C\}) \\ C_{NYC} + g(C, \{M, D\}) \end{cases} = \begin{cases} = 4407 \\ = 5108 \\ = 4407 \end{cases}$$

Soln:1 NY → M → D → C → NY

$g(NY, \{M, D, C\}) = M$ or C

$g(M, \{D, C\}) = D$
$g(D, \{C\}) = C$

$g(C, \{M, D\}) = D$
$g(D, \{M\}) = M$

Soln:2 NY → C → D → M → NY

Dijkstras → Greedy → +ve weights only

Bellman Ford → Dynamic

PAGE No. / DATE / /

* Bellman Ford Single Source Shortest Path :-

(Dynamic Prog.)

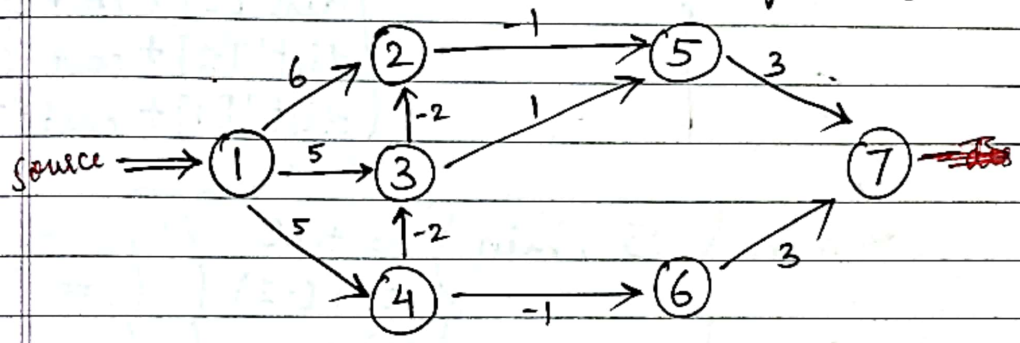
Recursive Formula

$$dist^{k}[u] = \min\left\{dist^{k-1}[u], \min\left\{dist^{k-1}[i] + cost[i,u]\right\}\right\}$$

(path length → k; current vertex → u)

$u < i \leq n$ $\quad i \in \{n\} - [\text{current} + \text{source}]$

if working for ①, then $i = [2,7]$



k → indicates path ka length kitna hona chahiye → Path length 1 matlab 1→2 yeh path

∞ → Path of path length k does not exist.

k	1	2	3	4	5	6	7
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

For k=2,

$$dist^{2}[2] = \min\left\{dist'[2], \min\left\{\begin{array}{l} dist'[3] + cost[3,2] \\ dist'[4] + cost[4,2] \\ dist'[5] + cost[5,2] \\ dist'[6] + cost[6,2] \\ dist'[7] + cost[7,2] \end{array}\right\}\right\}$$

$$= \min\left\{6, \begin{Bmatrix} 5+(-2) \\ 5+\infty \\ \infty+\infty \\ \infty+\infty \\ \infty+\infty \end{Bmatrix}\right\} = 3$$

$$dist^2[3] = \min\left\{5, \min\begin{Bmatrix} dist'[2]+cost[2,3] \\ dist'[4]+cost[4,3] \\ dist'[5]+cost[5,3] \\ dist'[6]+cost[6,3] \\ dist'[7]+cost[7,3] \end{Bmatrix}\right\}$$

$$= \min\left\{5, \min\begin{Bmatrix} 6+\infty \\ 5+(-2) \\ \infty+ \\ \infty+ \\ \infty+ \end{Bmatrix}\right\} = 3$$

$$dist^2[4] = \min\left\{5, \min\begin{Bmatrix} dist'[2]+[2,4] \\ dist'[3]+[3,4] \\ dist'[5]+[5,4] \\ dist'[6]+[6,4] \\ dist'[7]+[7,4] \end{Bmatrix}\right\}$$

$$= \left\{5, \min\begin{Bmatrix} 6+\infty \\ 5+\infty \\ \infty+ \\ \infty+ \\ \infty \end{Bmatrix}\right\} = 5$$

PAGE No.
DATE / /

$$dist^2[5] = \min\left\{\infty, \min\left\{\begin{matrix} dist'[2]+[2,5] \\ dist'[3]+[3,5] \\ dist'[4]+[4,5] \\ dist'[6]+[6,5] \\ dist'[7]+[7,5] \end{matrix}\right\}\right\}$$

$$= \min\left\{\infty, \min\left\{\begin{matrix} 6+(-1) \\ 5+1 \\ 5+\infty \\ \infty+\infty \\ \infty+\infty \end{matrix}\right\}\right\} = 5$$

$$dist^2[6] = \min\left\{\infty, \min\left\{\begin{matrix} dist'[2]+[2,6] \\ \text{"}\ [3]+[3,6] \\ \text{"}\ [4]+[4,6] \\ \text{"}\ [5]+[5,6] \\ \text{"}\ [7]+[7,6] \end{matrix}\right\}\right\}$$

$$= \min\left\{\infty, \min\left\{\begin{matrix} 6+\infty \\ 5+\infty \\ 5+(-1) \\ \infty \\ \infty \end{matrix}\right\}\right\} = 4$$

$$dist^2[7] = \min\left\{\infty, \min\left\{\begin{matrix} dist'[2]+[2,7] \\ \text{"}\ [3]+[3,7] \\ \text{"}\ [4]+[4,7] \\ \text{"}\ [5]+[5,7] \\ \text{"}\ [6]+[6,7] \end{matrix}\right\}\right\}$$

Here we feed all these values to the row with $i=2$

$$= \min\left\{\infty, \min\left\{\begin{matrix} 6+\infty \\ 5+\infty \\ 5+\infty \\ \infty \\ \infty \end{matrix}\right\}\right\} = \infty$$

PAGE No.
DATE / /

k = 3

$$dist^3[2] = \min\{dist^2[2], \min\{dist^2[3]+cost[3,2], dist^2[4]+cost[4,2], dist^2[5]+cost[5,2], dist^2[6]+cost[6,2], dist^2[7]+cost[7,2]\}\}$$

$$= \min\{3, \min\{3+(-2), 5+\infty, 5+\infty, 4+\infty, \infty+\infty\}\} = 1$$

$$dist^3[3] = \min\{dist^2 3 \downarrow 3, \min\{dist^2 \downarrow \; cost: 3\;[2]+\infty,\ 5\;[4]+(-2),\ 5\;[5]+\infty,\ 4\;[6]+\infty,\ \infty\;[7]+\infty\}\} = 3$$

$$dist^3[4] = \min\{dist^2[4] \downarrow 5, \min\{dist^2\ cost: [2]3+\infty,\ [3]3+\infty,\ [5]5+\infty,\ [6]4+\infty,\ [7]\infty+\infty\}\} = 5$$

$$dist^3[5] = \min\{5, \min\{dist^2\ cost: [2]3+(-1),\ [3]3+(1),\ [4]5+\infty,\ [6]4+\infty,\ [7]\infty+\infty\}\} = 2$$

$$dist^3[6] = \min\left\{4,\ \min\left\{\begin{array}{lll} & dist^2 & cost \\ [2] & 3 & +\infty \\ [3] & 3 & +\infty \\ [4] & 5 & +(-1) \\ [5] & 5 & +\infty \\ [7] & \infty & +\infty \end{array}\right\}\right\} = 4$$

$$dist^3[7] = \min\left\{dist^2[7],\ \min\left\{\begin{array}{lll} & dist^2 & cost \\ [2] & 3 & +\infty \\ [3] & 3 & +\infty \\ [4] & 5 & +\infty \\ [5] & 5 & +3 \\ [6] & 4 & +3 \end{array}\right\}\right\} = 7$$

$\therefore$ Fill the row with $k=3$ after this & we solve till $k=6$

Solution.

$(1,2) \Rightarrow k=3 = (1-4-3-2) = 1$
$(1,3) \Rightarrow k=2 = (1-4-3) = 3$
$(1,4) \Rightarrow k=1 = (1-4) = 5$
$(1,5) \Rightarrow k=4 = (1-4-3-2-5) = 0$
$(1,6) \Rightarrow k=2 = (1-4-6) = 4$
$(1,7) \Rightarrow k=5 = (1-4-3-2-5-7) = 3$

PAGE No.
DATE / /

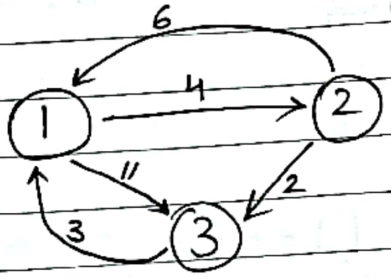
* All pair Shortest Path (Dynamic Programming) :-

Recursive Formula $A^k[i,j] = \min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\}, k \geq 1$

Algorithm

```
for k: 1 to n do
    for i: 1 to n do
        for j: 1 to n do
            A[i,j] = min
```

Q.)



$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array} \quad k=0$$

For A^1 :- $(k=1)$

$$A^1[3,2] = \min\{A^0[3,2], A^0[3,1] + A^0[1,2]\} = 7$$

(∞, 7)

$$A_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array} \qquad A_2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

$1 \to 3$ $3 \to$

$4+2 < 11$ $7+6$

$$A_3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

$1 \to 2$ $\quad 2 \to 1$

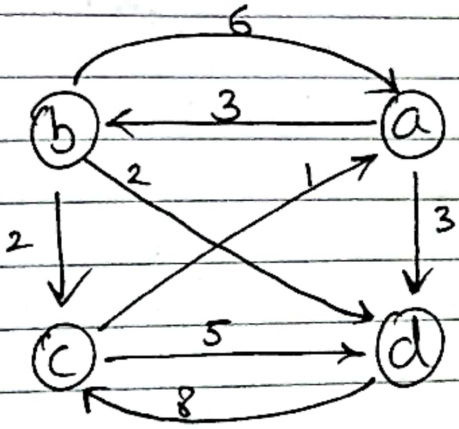
$6+7 > 4$ $\quad 6 > 2+3$

Solution Set

(1,2) $1 \to 2 = 4$ (2,1) $2 \to 3 \to 1 = 5$ (3,1) $3 \to 1 = 3$

(1,3) $1 \to 2 \to 3 = 6$ (2,3) $2 \to 3 = 2$ (3,2) $3 \to 1 \to 2 = 7$

Q.)



$$A^0 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 3 \\ 6 & 0 & 2 & 2 \\ 1 & \infty & 0 & 5 \\ \infty & \infty & 8 & 0 \end{bmatrix} \end{array}$$

$$A^a = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 3 \\ 6 & 0 & 2 & 2 \\ 1 & 4 & 0 & 4 \\ \infty & \infty & 8 & 0 \end{bmatrix} \end{array}$$

$b \to a \to d$ $\quad$ $b \to a \to c$

$\underline{2} < (6+3)$ $\quad$ $6+\infty > \underline{2}$

$c \to a \to d$ $\quad$ $c \to a \to b$

$\underline{1+3} < 5$ $\quad$ $\underline{1+3} < \infty$

$$A^b = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 3 \\ 6 & 0 & 2 & 2 \\ 1 & 4 & 0 & 4 \\ & & 8 & 0 \end{bmatrix} \end{array}$$

$a \to b \to c$ $\quad$ $a \to b \to d$

$\underline{3+2} < \infty$ $\quad$ $3+2 > \underline{3}$

$c \to b \to d$ $\quad$ $c \to b \to a$

$\underline{4} \not< 4+2$

Binary Search Algo

PAGE NO.
DATE / /

```
Algorithm BinSrch (A, n, x, j)
// Given an array A[1:n] of elements in ascending order,
// n>=0, determine if x is present, and if so set j= A[j]=x
// else j=0.
{
   integer high, low, mid, j, n;
   low := 1 , high := n ;
   while (low <= high)
   do {
         mid := (low + high)/2 ;
         case
               x < A(mid) : high := mid - 1;
               x > A(mid) : low := mid + 1;
               else
               j := mid , return A[j]
         end case
   repeat
   j = 0 ;
}
```

Space Complexity :- → Array of n eles. → Excess variables } O(n)

Time Complexity. ⟶ O(log n)

Recurrence ⟶ $T(n) = T(n/2) + c$ ⟶ Solve this further making $n/2^k = 1$

Min Max Algo

```
Algorithm MaxMin (i,j,max,min)
// a[1:n] is a global array, i & j are integers
// 1 ≤ i,j ≤ n. The goal is to set max & min to the
// largest and smallest values in a[i:j] resp.
{
  if (i==j) then max=min=a[i]; //Small P

  if (i==j-1) then
  {
    if (a[i]>a[j]) then
    {
      max=a[i]; min=a[j];
    }
    else
    {
      max=a[j]; min=a[i];
    }
  }

  else
  {  // If P is not small, divide P into subprobs.
     // Find where to split the set
     mid=[(i+j)/2];                    — (1)
     // Solve the subprobs.
     MaxMin(i,mid,max,min);
     MaxMin(mid+1,j,max1,min1);
     // Combine the solutions
     if (max1>max) then max=max1;}— (1)
     if (min1<min) then min=min1;
  }
}
```

Recurrence Relation

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

∴ From master method

$T(n) = O(n)$

$\therefore f(n) = 2$

↓

Time taken to divide subprob & combine soln.

MergeSort Algo

```
Algorithm MergeSort (low, high)
// a[1:n] is a global array to be sorted
// Small P is true if only one element is present in array
// to sort. In this case, list is already sorted.
{
    if (low < high) then
    {
        // Divide P into subproblems
        // Find where to split
        mid = [(low + high)/2];
        // Solve the subproblems
        MergeSort (low, mid);
        MergeSort (mid+1, high);
        // Combine the solutions
        Merge (low, mid, high);
    }
}

Algorithm Merge (low, mid, high)
// a[low:high] is global array containing 2 sorted subsets
// a[low:mid] & in a[mid+1, high].
// Goal is to merge these 2 subsets into a single set.
// b[] is an auxillary global array.
{
    h := low; i := low; j := mid+1;

    while ((h ≤ mid) and (j < high)) do
    {
        if (a[h] ≤ a[j]) then
        {
            b[i] = a[h]; h = h+1;
        }
```

```
        else
        {
            b[i] = a[j]; j = j+1;
        }
        i = i+1;
    }

    if (h > mid) then
    {
        for k := j to high do
        {
            b[i] = a[k]; i := i+1;
        }
    }
    else
    {
        for k := h to mid do
        {
            b[i] = a[k]; i := i+1;
        }
    }

    for k := low to high do a[k] := b[k];
}
```

(if any elements are left)

∴ Recurrence Eqn. : $T(n) = 2T(n/2) + n$;

By master method → $T(n) = O(n \log n)$

Space Comp → $O(n)$.

Quick Sort Algo

```
Algorithm QuickSort (p, q)        // p: low, q: high
// Sorts the elements a[p] .... a[q] which reside in the global
// array a[1:n] into asc. order; a[n+1] is considered to be
// defined & must be ≥ all elements in a[1:n];
{
    if (low < high) then
    {
        ~~p := Partion~~ // divide P into subproblems
        p := Partition (a, low, high+1)
        // Solve subproblems
        QuickSort (low, p-1);
        QuickSort (p+1, high);
        // There is no need to combine solns.
    }
}

Algorithm Partition (a, low, high)
// Within a[low], a[low+1] ..... a[high-1] the elements are
// rearranged in such manner that if initially t = a[low], then aft
// completion a[p] for some p betn low & high-1,
// a[k] ≤ t for low ≤ k ≤ p, and a[k] ≥ t for p ≤ k ≤ hig
// p is returned, set a[high] = ∞.
{
    V := a[low]; i = low; j := high;
    repeat
    {
        repeat
            i := i+1;
        until (a[i] ≥ V);
```

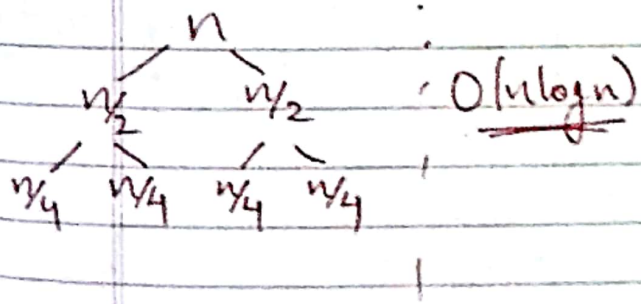
```
repeat
    j := j+1;
until (a[j] ≤ v);

if (i<j) then Swap(a,i,j);
} until (i≥j);

} a[low] := a[j]; a[j] := v; return j;
```

```
Algorithm Swap (a,i,j)  // Exchange a[i] & a[j]
{
} p := a[i]; a[i] := a[j]; a[j] := p;
```

Complexity

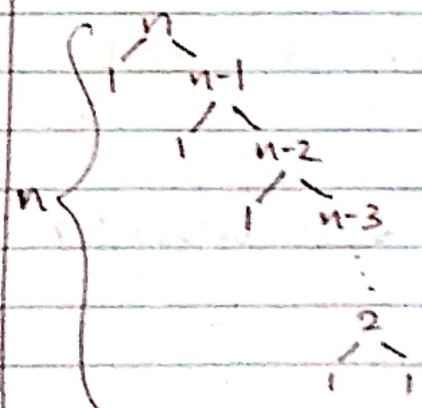
Best Case: Pivot always at centre

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



$\therefore O(n \log n)$

Worst Case: Pivot at edge



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

$T(n) = 0 + O(n^2) + O(n)$

$= O(n^2)$

GREEDY

Fractional Knapsack

```
Algorithm FracKnap (P,W,M,X,n)
// P[1:n] & W[1:n] contain profit & weights resp. of n objects
//ordered such that P[i]/W[i] >= P[i+1]/W[i+1].
// M is the knapsack capacity & X[1:n] is the soln. vector.
{
   Real P[1:n], W[1:n], X[1:n], M, Cu
   Integer i,n;
   X <- 0 // Initialize soln. vector to zero
   Cu <- M // Cu is remaining knapsack capacity
   for i <- 1 to n do
        if (W(i) > Cu) then exit for loop.
        X[i] <- 1;
        Cu <- Cu - W[i];
   Repeat

   if (i <= n) then X[i] <- Cu/W[i]
        . Calc total profit as well.
}
```

∴ Time Complexity

→ Sorting : O(n log n) [Using merge so

→ GreedyKnapsack : O(n).

∴ Total time → O(n log n)

Prims & Kruskals

Adjacency Matrix

* Kruskals :-

i] Sort all edges in ascending order of their weight
ii] Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If not, then include this edge or discard it.
iii] Repeat this step until there (V-1) edges in the spanning tree.

∴ Time Complexity → O(E log E) or O(E log V)

→ sorting takes O(E log E) time
→ find & union ops take O(log V) time
∴ overall complexity → O(E log E) or O(E log V)

Space → O(E+V)

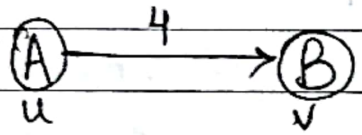
* Prim's :-

i] Determine arbitrary vertex as starting vertex of MST.
ii] Follow steps iii to v till there are vertices are not included in the MST. (fringe vertex)
iii] Find edges connecting any tree vertex with fringe vertices
iv] Find the minimum among these edges.
v] Add the chosen edge to the MST if it does not form any cycle.
vi] Return the MST & exit.

Time Complexity → $O(V^3)$ → adjacency matrix method
Time Complexity → $O(V^2)$ → normal method.

Space → O(E+V)

Dijkstras (Single Source Shortest Path)



FORMULA

if (dist(u) + cost(u,v) < dist(v))

0 + 4 ∞

4 < ∞

∴ update ∞ to 4

{

dist(v) = dist(u) + cost(u,v)

}

Time Complexity → $O(V^2)$ Since adjacency matrix

BFS

BACKTRACKING → Brute Force Approach

BFS ← BRANCH & BOUND

Terminologies.

i] State Space Tree → Soln. space organised as a tree.

ii] Explicit constraint → Rules that restrict each component of soln. vector to take values only from given set S. ∴ $x_i = 0$ or $x_i = 1$ in 0/1 knapsack.

soln. vector x_i ← $x[]$

$1 \le x_i \le n$, $\forall 1 \le i \le n$ in n-queen

iii] Implicit constraint → Rules that describe the way in which x_i's must relate to each other or which components satisfy the criteria func.

↳ No. queens can attack in same row, column or diagnol in N Queen

→ knapsack capacity in 0/1 knapsack.

iv] Solution space → set of all tuples that satisfy explicit constraints.

v] Live node → Node that has been generated but none of its children are generated.

*vi] Bounding func./Criteria :- Func. created to kill a live node without generating all its children.

always kept in mind while making state space tree.

vii] Extended node :- Live node whose children are being generated.

viii] Dead node :- Node that is not be extended further or all of whose children are generated.

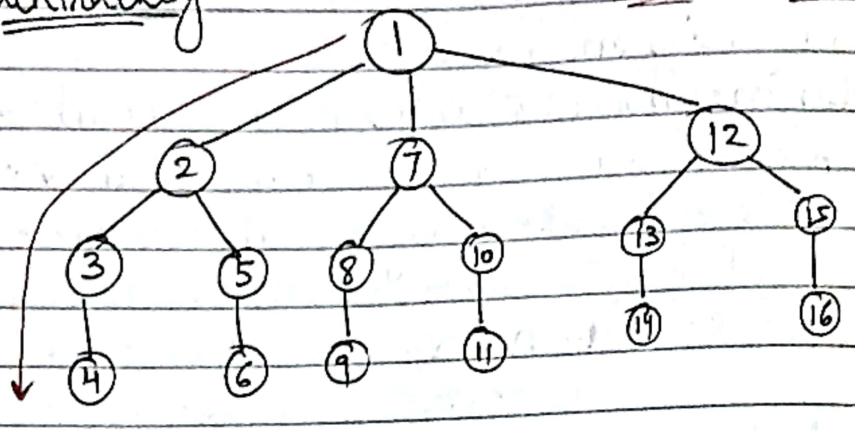
ix] Answer node :- Node that represents answer of the prob. i.e. node at which bounding func. are satisfied.

x] Solution node :- Node that has possibility to become answer node

PAGE No.
DATE / /

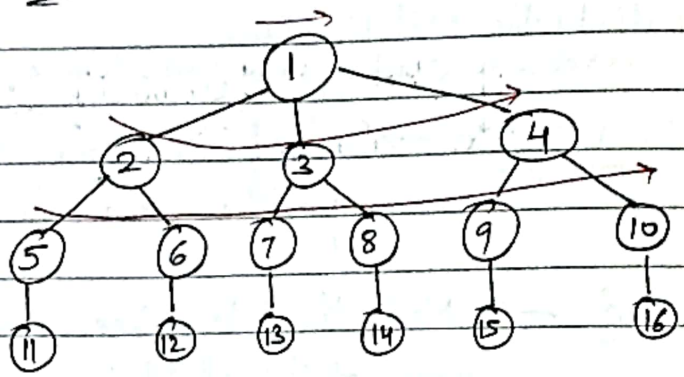
Backtracking

Follow Numbering



DFS

Branch & Bound



BFS

Backtracking	Branch & Bound.
→ State Space tree traversed by DFS.	→ State Space tree traversed by BFS. & DFS both.
→ Used to find all possible solutions to a problem.	→ Used to find the most optimal solution to a problem.
→ Solves decision problems	→ Solves optimisation problems (only minimization ones)
→ ~~Backtrack~~ More efficient	→ Less efficient.
→ Solves N-Queen, Graph Coloring, Sum of subset problems	→ Solves Knapsack Problem, Travelling Salesman Prob.

Explicit cond. → $1 \le x_i \le n$ [x_i column nos. of the board] where $1 \le i \le n$.

N-Queen Problem

Implicit cond. → No 2 queens can attack same row, column, diagnol.

```
Algorithm NQueens (k, n)
// Using backtracking, this procedure prints all
// possible placements of n-queeens on an nxn
// chessboard so that they are non-attacking.
{
    for i:=1 to n do
    {
        if Place(k,i) then
        {
            x[k] := i;
            if (k==n) then write (x[1:n]);
            else NQueens (k+1, n);
        }
    }
}
```

i → column.
k → row

```
Algorithm Place (k, i)
// Returns true if a queen can be placed in kth row and
// ith column. Otherwise returns false. x[] is a global
// array whose first (k-1) values have been set.
// Abs(r) returns the absolute value of r.
{
    for j := 1 to k-1 do
        if ((x[j]==i) or (Abs(x[j]-i)=Abs(j-k)))
        {
            then return false;
        }
        return true;
}
```

2 lie in same column

2 lie in same diagonal.

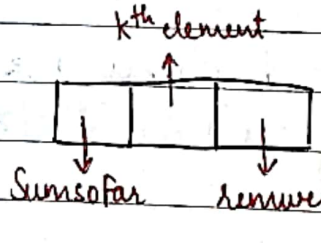
Time Complexity → O(N!) | Space Comp → O(N)

weight array given & find appropriate subset whose sum matches the required sum.

PAGE No.
DATE / /

Sum of Subsets

```
Algorithm SumOfSubsets (Sumsofar, k, remweight)
// Used to find all solutions of sum of subsets problem.
// X[] is a solution vector.  M is the req. sum.
{
    Set X[k] = 1 ;
    if (Sumsofar + W[k] = M) then
        print X[1:k] ;    // soln. is found.

    else
    {
        if (Sumsofar + W[k] + W[k+1] <= M) then
        { //Generate left child
            SumOfSubsets (Sumsofar + W[k], k+1, remweight - W[k]
        }
    }
    if((Sumsofar + remweight - W[k] >= M) and
                (Sumsofar + W[k+1] >= M)) then
    {   //Generate Right Child.
        X[k] := 0 ;
        SumOfSubsets (Sumsofar, k+1, remweight - W[k]);
    }
}
```



Graph Coloring Problem

```
Algorithm mColoring (k)  node
// Graph is represented by a boolean adjacency matrix
// G[1:n, 1:n].
{
    repeat
    {
        NextValue(k);
        if (x[k]==0) return ~~return true~~ then return;
        if (k==n) then write x[1:n];
        else mColoring (k+1);
    } until false;
}

Algorithm NextValue (k)
{
    repeat
    {
        x[k]= (x[k]+1) mod (m+1); // Next Colour.
        if (x[k]=0) then return;

        for j:=1 to n do
        {
            if ((G[k,j]≠0) and (x[k]=x[j]))
                then break;
        }
        if (j==n+1) then return; // new colour found
    } until false;
}
```

Time Complexity → O(m^(v))

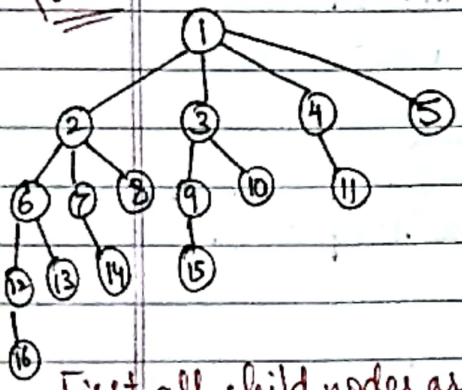
Space Complexity → O(V)

Branch & Bound.

State Space Tree is constructed using 3 methods.

FIFO

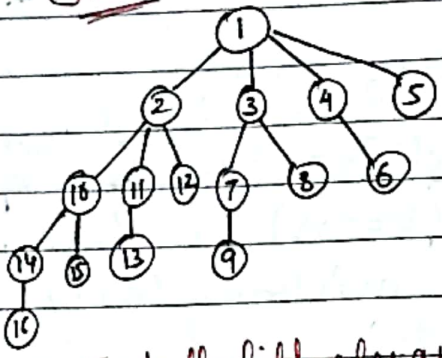
Queue



First all child nodes are generated, then the first element of queue is removed & split into its child nodes

LIFO

Stack



First all child nodes are generated, then the last element of the stack is popped & split into its child nodes.

Least Cost BB

A cost func. is designed in such way, that it tells which node is to be split by knowing the chance of generating an outcome

---

0/1 Knapsack.

→ maximisation prob converted to minimisation prob.

↳ We use LC BB technique

where we calc. cost of each node & go ahead with least one.

∴ 2 funcs are there

$$\text{Upperbound}(u) = \sum_{i=1}^{n} p_i x_i \quad \text{(Without fraction)}$$

$$\text{Cost} \quad (c) = \sum_{i=1}^{n} p_i x_i \quad \text{(With fraction)}$$

→ Taken to be -ve, as we convert it to minimise

We begin with including all objects & then go ahead

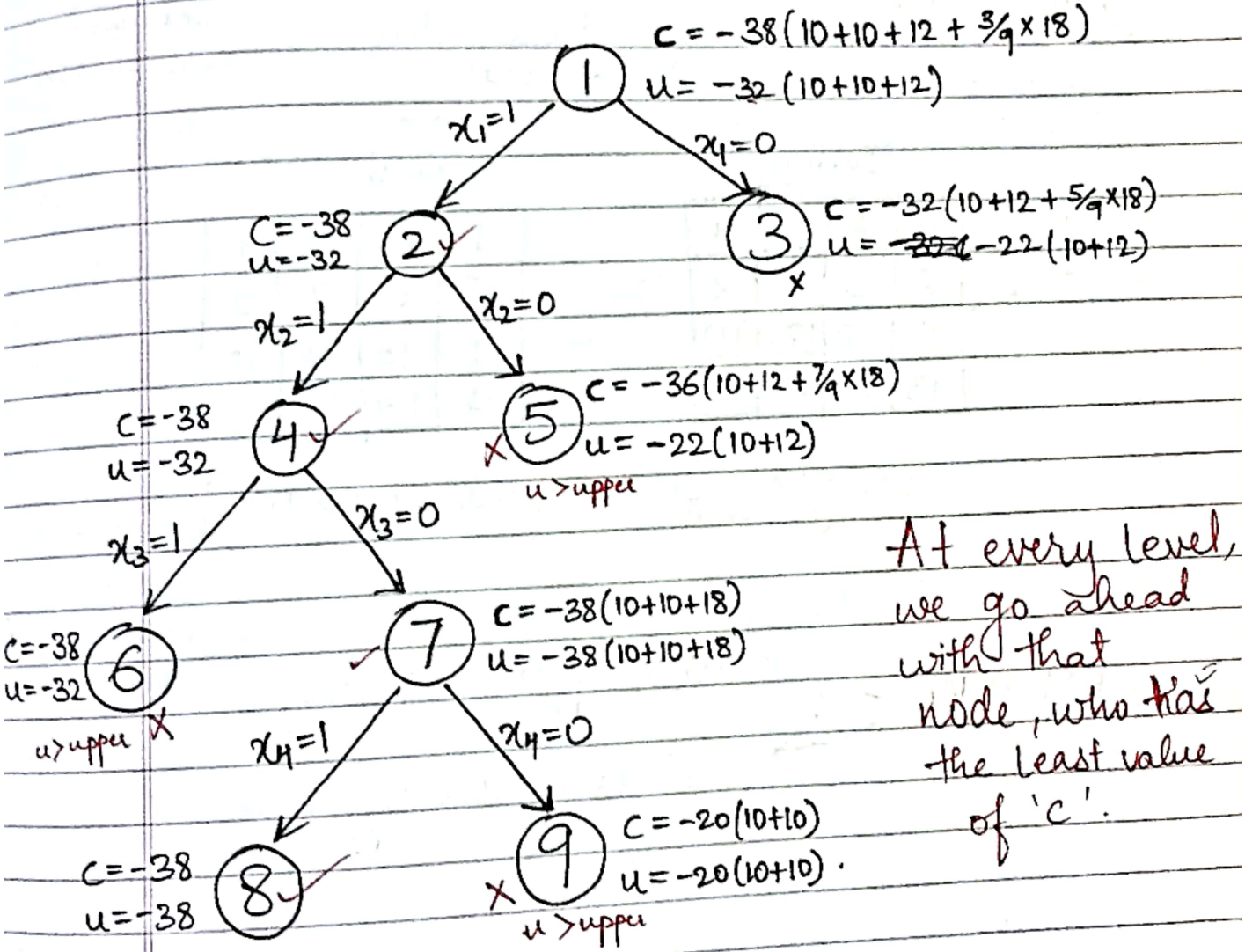
capacity

$n = 4,\ m = 15$

	1	2	3	4
P	10	10	12	18
W	2	4	6	9

$x[]$ → Solution vector

∴ Generate State Space Tree: upper = ~~0~~ → ~~-32~~ → -38



At every level, we go ahead with that node, who has the least value of 'c'.

∴ $x[] = \{1, 1, 0, 1\}$.

Max Profit → 38

15 puzzle problem → Check in ppts.

Very imp cost func: $c(x) = f(x) + g(x)$

- $f(x)$: no. of moves from initial state
- $g(x)$: no. of non-blank tiles not in goal position

Solved by LCBB

[g]

Initial

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

Goal

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

∴ $g(x) = 3$ [no of misplaced tiles]

Cost func → Gives approximation
→ Used as bounding func.

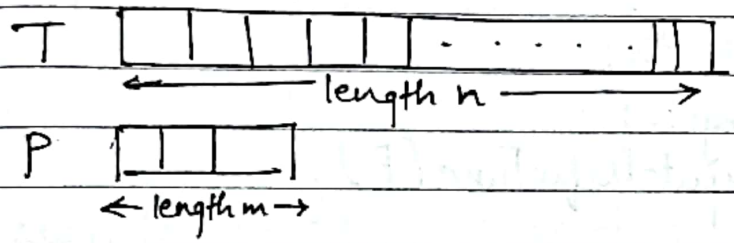
locating pattern P in Text T.

DYNAMIC PROGRAMMING

STRING MATCHING

INTRO FROM PPT

Involves locating a pattern P in the Text T.



$n \geq m \geq 1$

NAIVE

If pattern P occurs in text after S shifts then

$P[1 \ldots m] = T[s+1 \ldots s+m]$ where $n-m \geq s \geq 0$

∴ if S is finite, then we can say s is valid shift.

Naive String Matching.

```
Algorithm Naive String Matcher (T, P)
{
    n = T.length;
    m = P.length;

    for s = 0 to n-m do
    {
        if P[1...m] == T[s+1 ... s+m] then
        print "Pattern occurs after shift 's'.
    }
}
```

Preprocessing Time → 0

Matching Time → $O((n-m+1)m)$

DYNAMIC PROGRAMMING

Longest Common Subsequence

Eg] String 1 :- a b c d e f g h i
String 2 :- e c d g i

∴ subsequences → e g i
→ c d g i ✓ (Cannot overlap)
→ d g i
→ g i

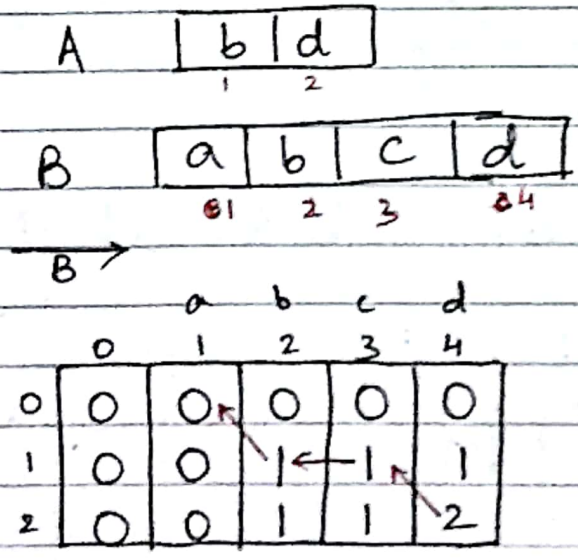
{ To find subsequences in these 2

Eg] String 1 : a b d a c e
String 2 : b a b c e

Subs → b a c e → cannot overlap.
a b c e

Using DP:

bottom up approach




```
if (A[i] == B[j])
{
  LCS[i,j] = 1 + LCS[i-1,j-1]
}
else
{
  LCS[i,j] = max(LCS[i-1,j], LCS[i,j-1])
}
```

Step 1:- Fill row 0 & column 0 with zeros
Step 2:- Use the adjacent logic to fill the table row wise
Step 3:- Once the table is filled, trace backward, the common letter is found wherever there is a diagonal & its length is bottom right

if match → 1 + diagonally left upar vaala value.

if not match → max of (left value, upar value).

Time Complexity → O(m x n) m & n are lengths of strings

Space Complexity → O(n) → 2 arrays to store strings.

Eg]

str1 → s t o n e

str2 → l o n g e s t.

		l	o	n	g	e	s	t
	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	1	1
t	2	0	0	0	0	0	1	2
o	3	0	0	1	1	1	1	2
n	4	0	0	1	2	2	2	2
e	5	0	0	1	2	2	3	(3)

o n e

∴ ~~toto~~ LCS → one (3)

∴ Recursive formula.

$$LCS[i,j] = \begin{cases} 0 & , i=j=0 \\ 1 + LCS[i-1,j-1] & , i,j>0 \;\&\; x_i = y_i \\ \max(LCS[i-1,j], LCS[i,j-1]) & , i,j>0 \;\&\; x_i \neq y_i \end{cases}$$

We generally return only length of LCS while coding.

0/1 Knapsack Problem

PAGE No.
DATE / /

~~Algorithm Knapsack (n, w, v, x)~~

☆ Logic

$$B[k,w] = \begin{cases} 0 & k=0 \text{ or } w=0 \\ B[k-1,w] & w < w_k \\ \max(B[k-1,w], B[k-1,w-w_k] + P[k]) & w \geq w_k \end{cases}$$

k → item no.
w → weight in knapsack
w_k → weight of item
P[k] → profit of k^{th} item.

Time Complexity → O(k*w)
k → no. of items
w → capacity of knapsack

Eg

(k) Item	1	2	3	4
(w_k) Weight	2	3	4	5
(P_k) Value	4	8	9	11

W=5

Weight in knapsack (w)

Items (k)	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	4	4	4	4
2	0	0	4	8	8	12
3	0	0	4	8	9	12
4	0	0	4	8	9	(12) → optimal profit

To find items :- Bottom up approach,

i] Start from B[k,w] bottom right element. If it matches Till doesn't it's upper element, then do k = k-1.

ii] If it doesn't match, then k^{th} item is selected ∴ print k & set $w = w - w_k$.

Works for directed & undir
& also for negative edges

All Pair Sh

Algorithm

```
for k:=1 to n do
{   for i:=1 to
    {   for j:=
        {
            A[i,j]
        }
    }
}
```

Recursive formula.

$A^k[i][j] = mi$

Distance matrices a

→ nxn matric
→ D^0 is initial m
source i &
→ D^k is matrix
i & j ,
→ D^n is final

See exa

Works for directed & undirected & also for negative edges

All Pair Shortest Path - [Floyd Warshall]

Algorithm

```
for k:=1 to n do
{
    for i:=1 to n do
    {
        for j:=1 to n do
        {
            A[i,j] = min(A[i,j], A[i,k] + A[k,j]);
        }
    }
}
```

Recursive formula.

$$A^k[i][j] = \min\left(A^{k-1}[i][j],\ A^{k-1}[i][k] + A^{k-1}[k][j]\right)$$

Distance matrices are computed.

- $n \times n$ matrices
- D^0 is initial matrix with no intermediate betn. source i & destination j.
- D^k is matrix with k as intermediate vertex betn i & j, ~~k is betn~~ $1 \leq k \leq n-1$
- D^n is final matrix with all shortest paths.

See example solved in notes before.

Single Source Shortest Path [Bellman Ford]

→ Handles -ve edges, slower than Dijkstras but more versatile.
→ Combo of Dijkstras algo & unweighted algos.

∴ Relaxation formula.

If (d[v] > d[u] + cost[u,v].)
  then d[v] = d[u] + cost[u,v]; ☆

∴ Max Path length (k) = |V| - 1

∴ Main Formula.

☆ $dist^k[u] = \min\left(dist^{k-1}[u], \min\left(dist^{k-1}[i] + cost[i,u]\right)\right)$

See example in notes to verify the same.

Algo

```
SSSP
{
  for each vertex k do.
  {
    D[k] = ∞;
  }
  for each vertex i=1 to |V| do
  {  for each edge (e1,e2) do
     {  Relax(e1,e2);
     }
  }
  for each edge (e1,e2) do
  {
    if D[e2] > D[e1]+W[e1,e2] then
       print "Graph has -ve weight cycle";
  }
}
```

```
Relax (e1,e2)
{
  for each edge (e1,e2) do
  {
    if D[e2] > D[e1]+W[e1,e2]
    then D[e2] = D[e1]+W[e1,e2];
  }
}
```

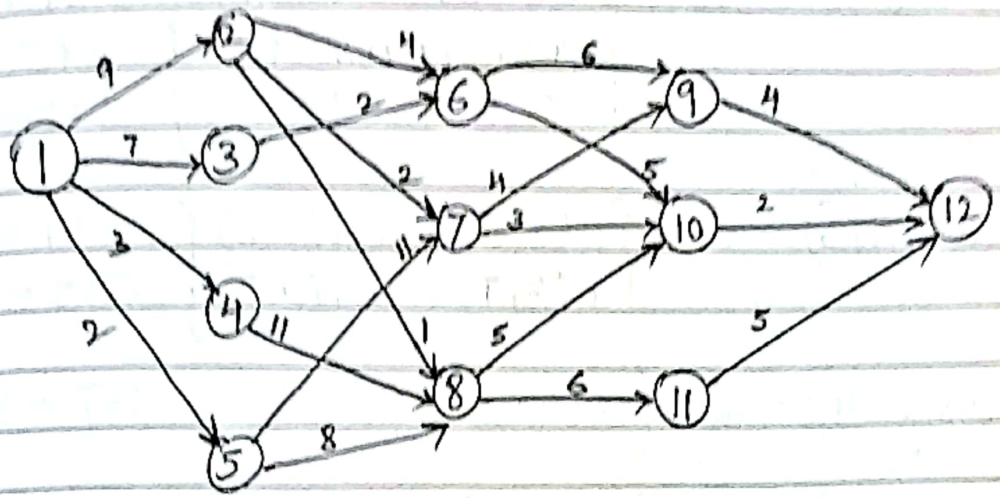
Multistage Graph

Forward Approach.

To find path with min. weight from source to sink.

src. ↓ v_1 — v_2 — v_3 — v_4 — V_5 ← sink

Stages



Principal of Optimality is applicable as we can solve this problem in a sequence of subprobs/decisions.

We find cost of each vertex starting from the sink (12).

V	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

d ← vertex which gives min. weighted path.

Stage 5 : $\because$ cost (5, 12) = 0 (stage, vertex)

Stage 4
- cost(4,9) = 4
- cost(4,10) = 2
- cost(4,11) = 5

Stage 3
- cost(3,6) = min { c(6,9) + cost(4,9), c(6,10) + cost(4,10) } = min { 6+4, 5+2 } = 7 ✓ (weight of edge betn 6 & 9)
- cost(3,7) = min { c(7,9) + cost(4,9), c(7,10) + cost(4,10) } = min { 4+4, 3+2 } = 5 ✓
- cost(3,8) = min { c(8,10) + cost(4,10), c(8,11) + cost(4,11) } = min { 5+2, 6+5 } = 7 ✓

Stage 2:

$$cost(2,2) = \min \begin{cases} c(2,6) + cost(3,6) \\ c(2,7) + cost(3,7) \\ c(2,8) + cost(3,8) \end{cases} = \begin{matrix} 4+7=11 \\ 2+5=7 \checkmark \\ 1+8=18 \end{matrix}$$

$$cost(2,3) = \min \begin{cases} c(3,6) + cost(3,6) \\ c(3,7) + cost(3,7) \end{cases} = \begin{matrix} 2+7=9 \checkmark \\ 7+5=12 \end{matrix}$$

$$cost(2,4) = \min \{ c(4,8) + cost(3,8) \} = 11+7=18 \checkmark$$

$$cost(2,5) = \min \begin{cases} c(5,7) + cost(3,7) \\ c(5,8) + cost(3,8) \end{cases} = \begin{matrix} 11+5=16 \\ 8+7=15 \checkmark \end{matrix}$$

Stage 1:

$$cost(1,1) = \min \begin{cases} c(1,2) + cost(2,2) \\ c(1,3) + cost(2,3) \\ c(1,4) + cost(2,4) \\ c(1,5) + cost(2,5) \end{cases} = \begin{matrix} 9+7=16 \\ 7+9=16 \\ 3+18=21 \\ 2+15=17 \end{matrix}$$

∴ Recursive Formula:

$$cost(i,j) = \min_{\langle j,l \rangle \in E \quad l \in V_{i+1}} \{ c(j,l) + cost(i+1,l) \}$$

(stage, vertex)

∴ After these computations & table filling, actual DP starts where we go FORWARD and take a sequence of decisions to determine minimum path length.

PAGE No.
DATE / /

∴ For determining paths, we use 'd' from the table.

stage ↙ vertex

$d(1,1) = 2$

$d(2,2) = 7$

$d(3,7) = 10$

$d(4,10) = 12$

∴ Path.

$1 \to 2 \to 7 \to 10 \to 12$

Length = 16.

Similarly,

$d(1,1) = 3$

$d(2,3) = 6$

$d(3,6) = 10$

$d(4,10) = 12$

Path.

$1 \to 3 \to 6 \to 10 \to 12$

Length = 16.

String Matching with Finite Automata

```
Finite Automaton Matches (T, d, m)
{
    n ← length[T]
    q ← 0
    for i ← 1 to n do
        q ← d(q, T[i])
        if q = m then
            print "Pattern occurs with shift" i-m
}
```

Matching time
↳ O(n)

```
Compute Transition Function (P, Σ)
{
    m ← length[P]
    for q = 0 to m do
        for each character a ∈ Σ do
        {   k = min(m+1, q+2);
            repeat
                k = k-1
            until Pk ⊇ Pqa
            δ(q, a) = k
        }
    return δ;
}
```

Transition Calculation
⇓
$O(m^3 \times |\Sigma|)$

String

KMP Solved Example

PAGE No.
DATE: / /

Pattern : a b a b a c a

String : b a c b a b a b a b a c a a b

First compute prefix of pattern :-

Initially, length(p) = m = 7

$\pi(1) = 0$, $k = 0$

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0	1	2	3	0	1

∴ Step1 : q=2, k=0 as b doesn't occur previous.

Step2 : q=3, k=0 as p[1] = p[3], → $\pi[3] = k++ (1)$,

Step3 : q=4, k=1 ∵ p[2] = p[4] → $\pi[4] = k++ (2)$,

Step4 : q=5, k=2 ∵ p[3] = p[5] → $\pi[5] = k++ (3)$,

Step5 : q=6, k=3 ∵ no match ∴ $\pi[6] = 0$, ∴ k=0

Step6 : q=7, k=0 p[1] = p[7] → $\pi[7] = k++ (1)$,

∴ Prefix func

q	1	2	3	4	5	6	7
P	a	b	a	b	a	c	a
π	0	0	1	2	3	0	1

∴ Matching Process

P[q+1] with T[i]

⟹ Step 1: i=1, q=0. ∴ Comparing P[1] with T[1]

T: b a c b a b a b a b a c a a b
   ↓
P: a b a b a c a
   q+1

∵ P[1] ≠ T[1], i++, q unchanged as q=0 ~~q = π[q] → π[0] = 0~~

⟹ Step 2: i=2, q=0 Comparing T[2] with P[1]

T: b a c b a b a b a b a c a a b
     ↑
P: a b a b a c a
   q+1

∵ P[1] = T[2], q++ & i++

⟹ Step 3: i=3, q=1 Comparing T[3] with P[2]

T: b a c b a b a b a b a c a a b
       ↑
P: a b a b a c a
     q+1

∵ P[2] ≠ T[3], q = π[q] → π[1] = 0

i++

PAGE No.
DATE / /

⇒ Step 4: $i=4, q=0$ Comparing $T[4] = P[1]$

```
T: b a c b a b a b a b a c a a b
       i
       ↑
P:   a b a b a c a
     q+1
```

$\because T[4] \neq P[1]$, q unchanged → as $q=0$; i++.

⇒ Step 5: $i=5, q=0$

```
T: b a c b a b a b a b a c a a b
         i
         ↑
P:   a b a b a b a
     q+1
```

$\because T[5] = q[0]$, i++ & q++

⇒ Step 6: $i=6, q=1$ $\because T[6] = q[2]$, i++ & q++

⇒ Step 7: $i=7, q=2$ $\because T[7] = q[3]$, i++ & q++

⇒ Step 8: $i=8, q=3$ $\because T[8] = q[4]$, i++ & q++

⇒ Step 9: $i=9, q=4$ $\because T[9] = q[5]$, i++ & q++

⇒ Step 10: $i=10, q=5$

```
T: b a c b a b a b a b a c a a b
               ↑
P:       a b a b a c a
```

$\because T[10] \neq q[6]$, i++.

$q = \pi[q] = \pi[5] = 3$

$\because T[i] = P[q+1]$ → q++.

Step 11 $i = 11$, $q = 4$ Comparing T[11] & P[5]

T: b a c b a b a b a b a l c a a b

P: a b a b a c a

$\because T[11] = P[5]$, q++ & i++

Step 12: $i = 12$, $q = 5$; $\because T[12] = P[6]$, q++ & i++

Step 13: $i = 13$, $q = 6$ $\because T[13] = P[7]$, q++ & ~~i++~~

Step 14 $i = 13$, $q = 7$ $\because q = 7 = m$

Pattern found at (i-m) after → 13 - 7 = 6 shifts

positioning of a NXN chessboard such that they don't attack each other.

N-Queen: [4 Queen]

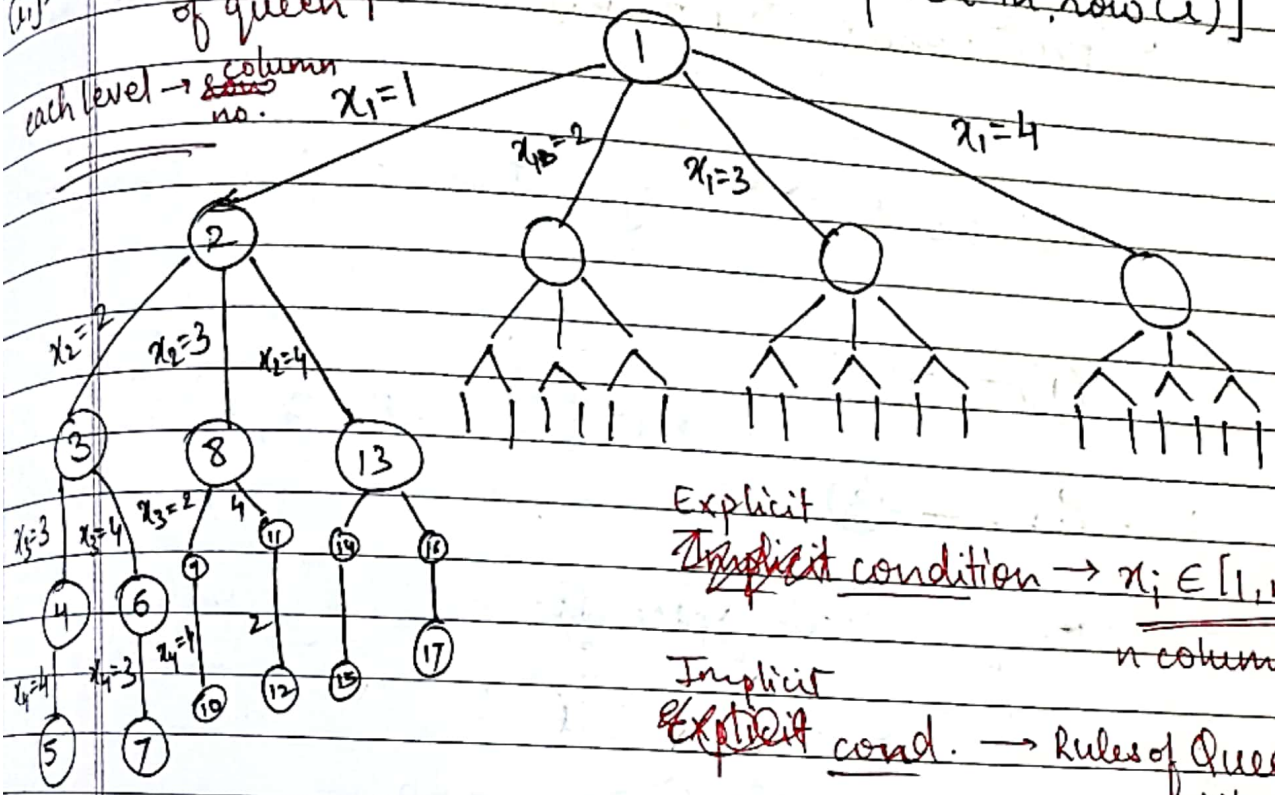
PAGE No.
DATE / /

Making the State Space Tree: [We initially place 1st queen in row (i)]

matrix be (i,j)

x_i → denotes row no. of queen

each level → column no.

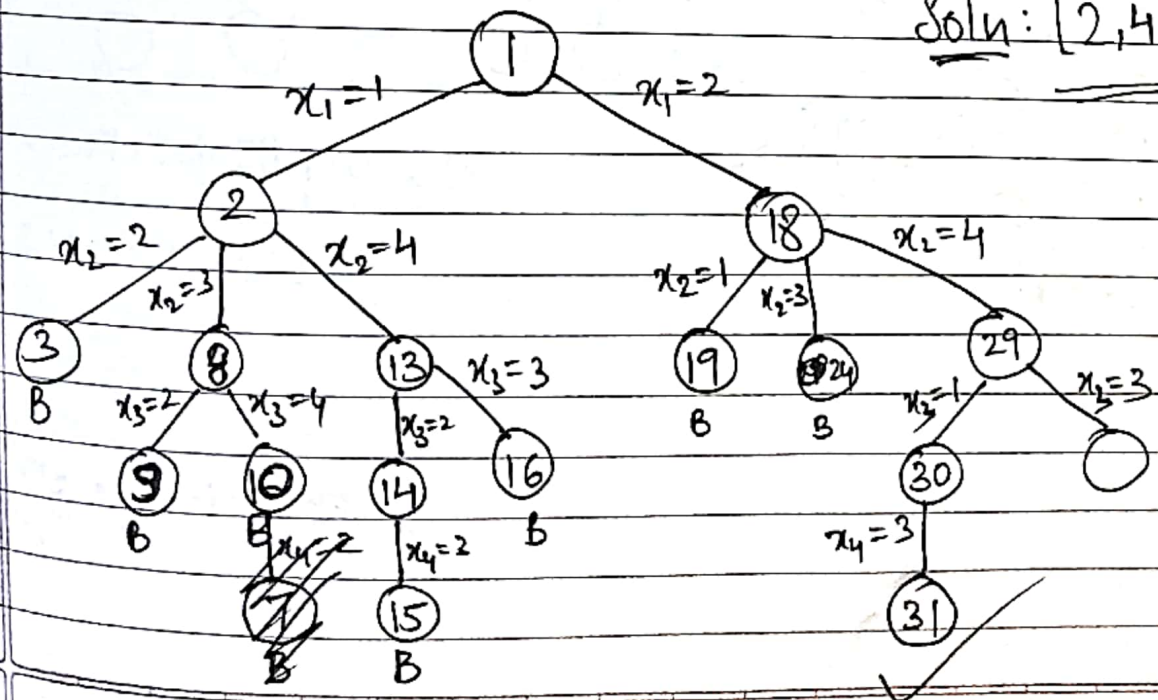


Explicit condition → $x_i \in [1, n]$ (n columns)

Implicit cond. → Rules of Queen Attack.

Partial Backtracked Tree

Soln: [2,4,1,3]



A coloring of graph $G(V,E)$ is a mapping of $f: V \to C$ where C is a finite set of colors, such that adjacent vertices are not assigned same color.

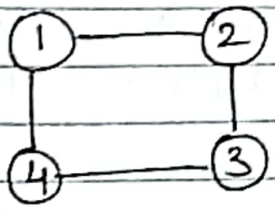
Graph Coloring Problem.

~~[illegible]~~ Chromatic No. → smallest no. of colors with which we can color a graph. (χ)

for all vertices

- Explicit cond. → Solution vector has all possible combos of colors.
- Implicit cond. → No adjacent vertices have same color.

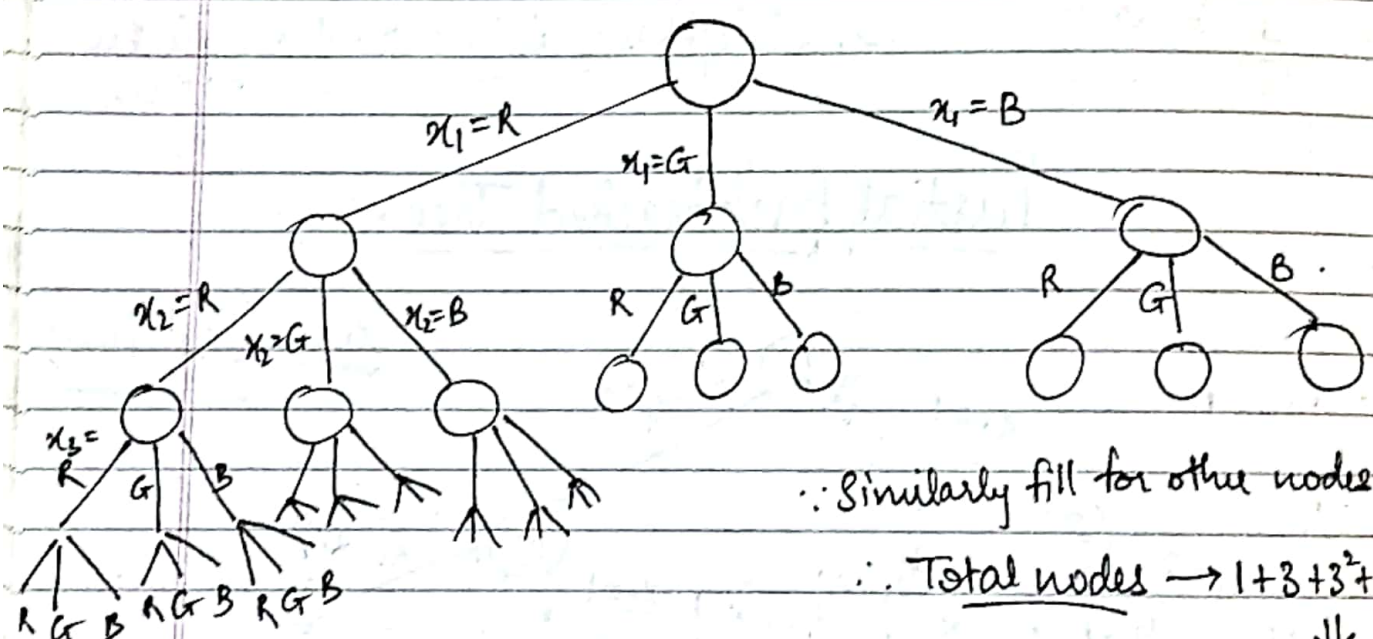
Eg]



$m = 3$ {R, G, B}.

State Space Tree

x_1 → Vertex



∴ Similarly fill for other nodes.

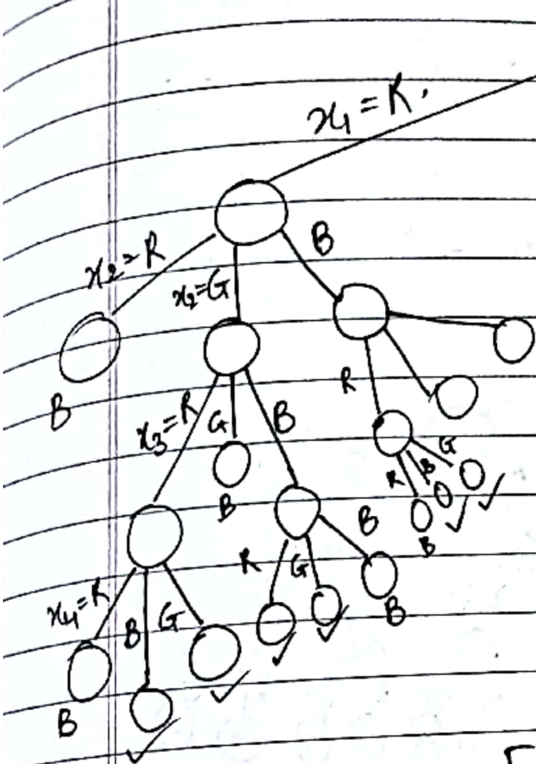
∴ Total nodes → $1+3+3^2+3^3+3^4$

$\Downarrow$

$\frac{3^{n+1}-1}{2}$.

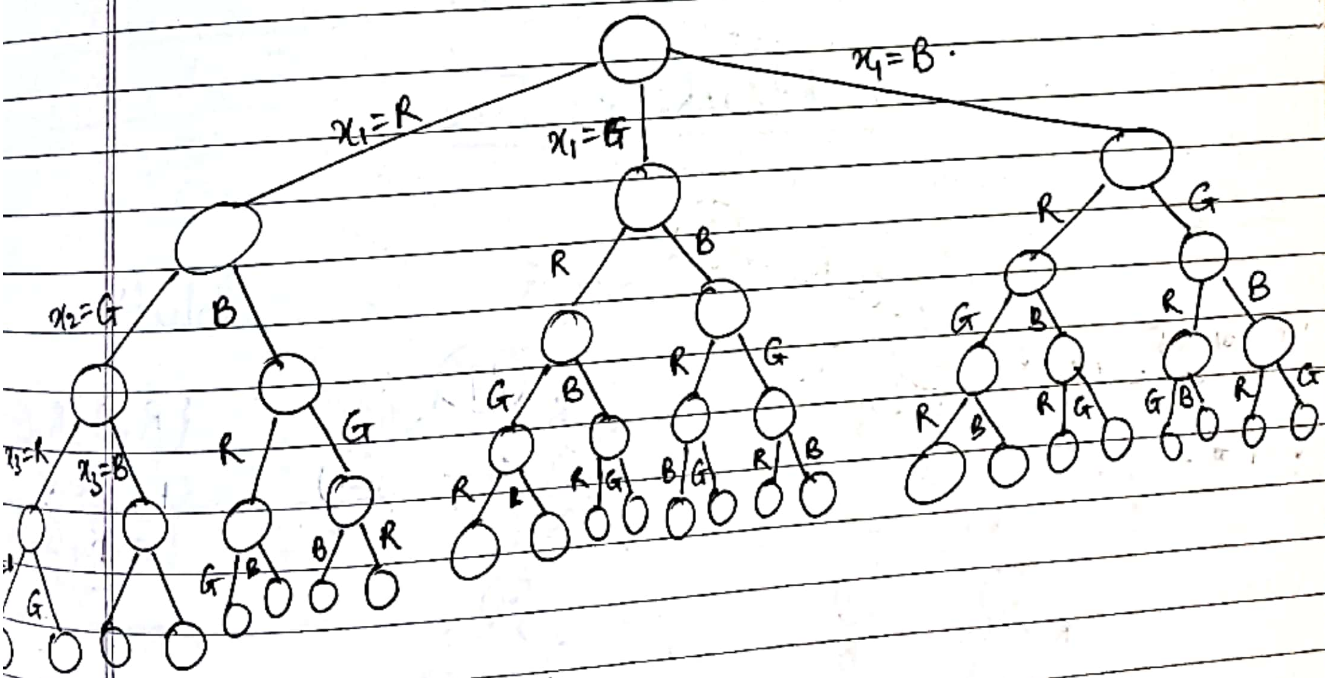
∴ Time Complexity → 3^n //

Partial Backtracked tree.

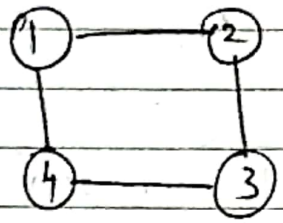


like this we check for all nodes & colors.

Final Tree. $n=4$, $m=3$ $\{R, G, B\}$.

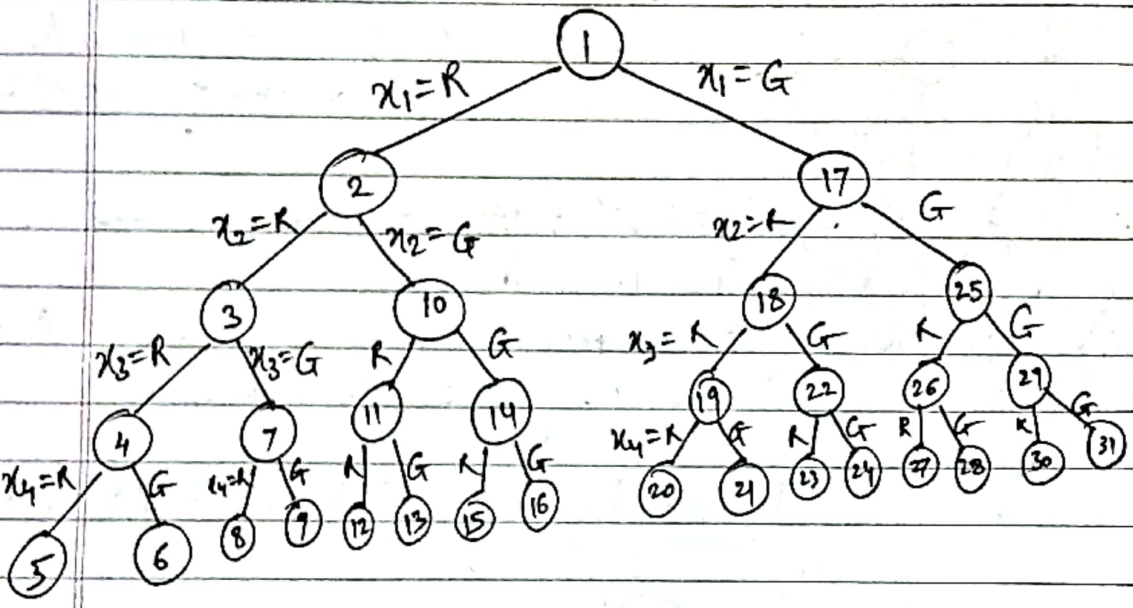


Similarly,

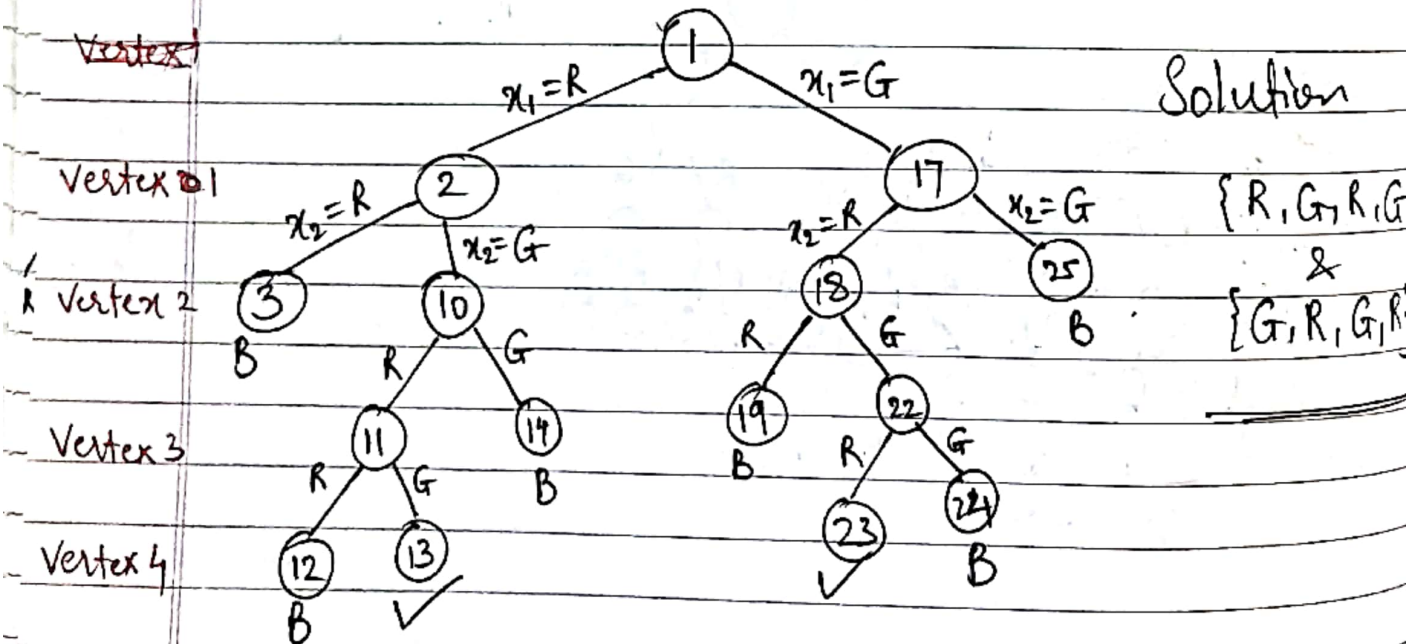


$n=2$ {R, G}

State Space Tree



Backtracking Tree



Solution

{R, G, R, G

&

{G, R, G, R

∴ Note: Solution tree has only the path with answer

Some values will be given along with a required sum such that we have to find set of those values which match the sum.

Sum of Subsets Problem

W[] → values

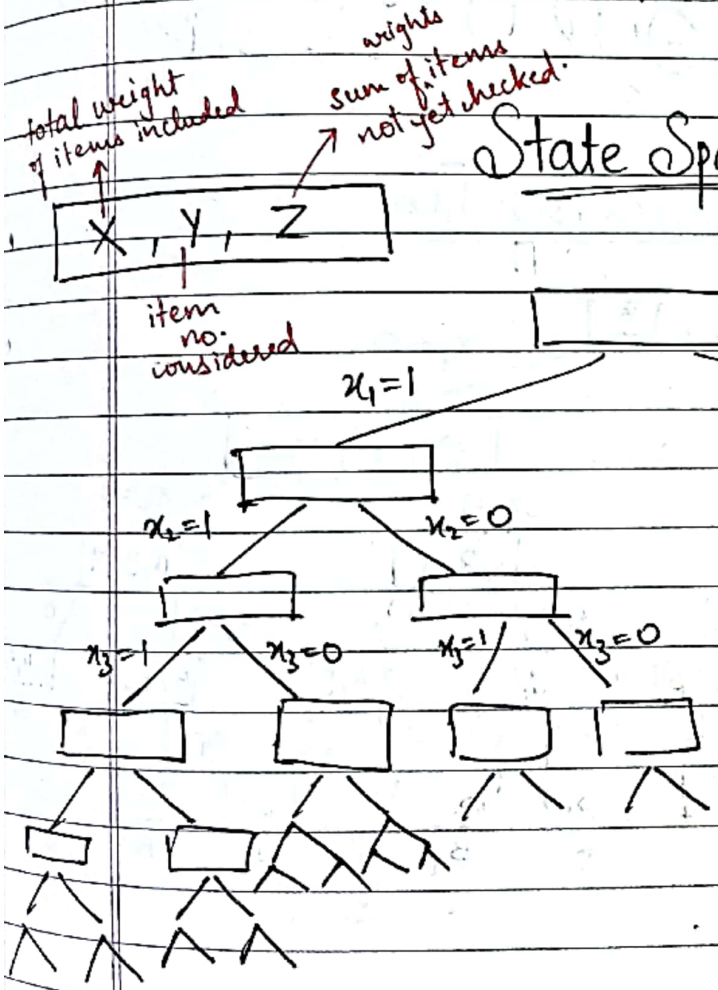
m → required sum

Explicit cond. → $x_i \geq 0$ & $x_i \in \{j \mid 1 \leq j \leq n\}$ } Items from the bag are taken

Implicit cond. → No two x_i can be same.
- ↳ $\sum W_{x_i} = m$ (Sum of all $W_{x_i} = m$)
- ↳ $x_i < x_{i+1}$, $1 \leq i \leq k$. (helps to avoid multiple generations of same subset)
  - ↳ sort W[] initially.

Bounding func. → Total value after adding item should be less than m.
- ↳ Sum of items present in list and those left out must be greater than m.

State Space Tree.

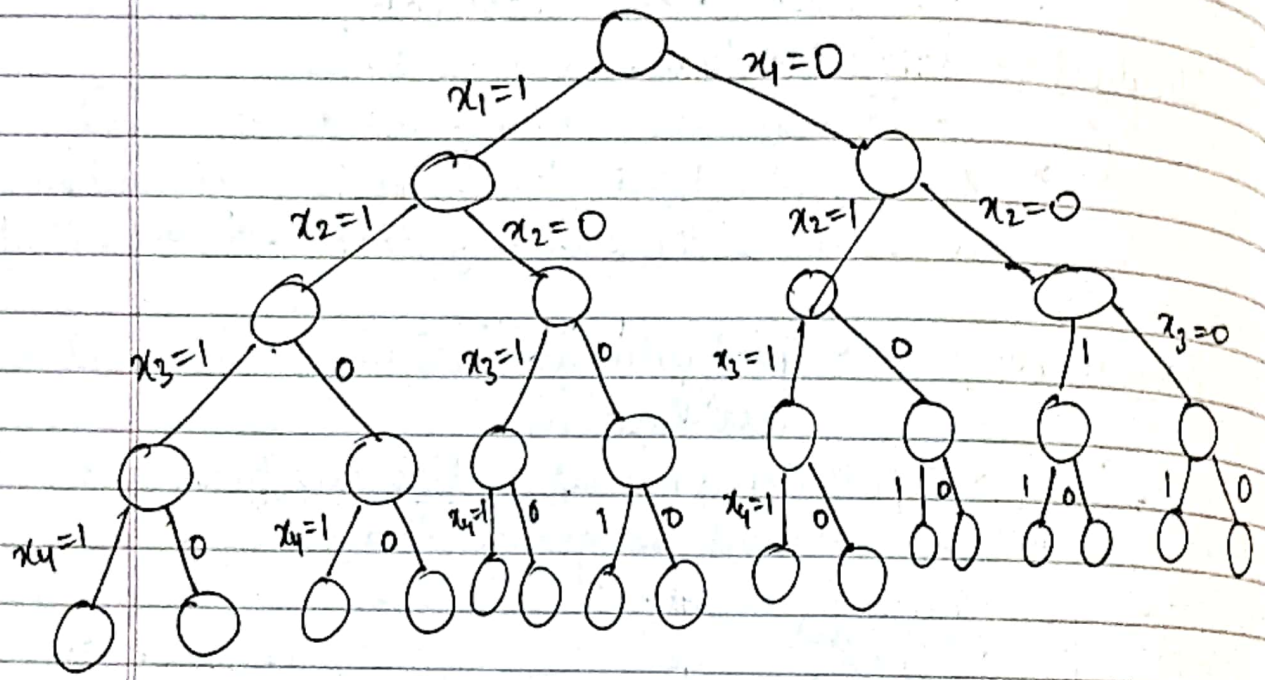


Has the possibilities of item, whose it included or not included.

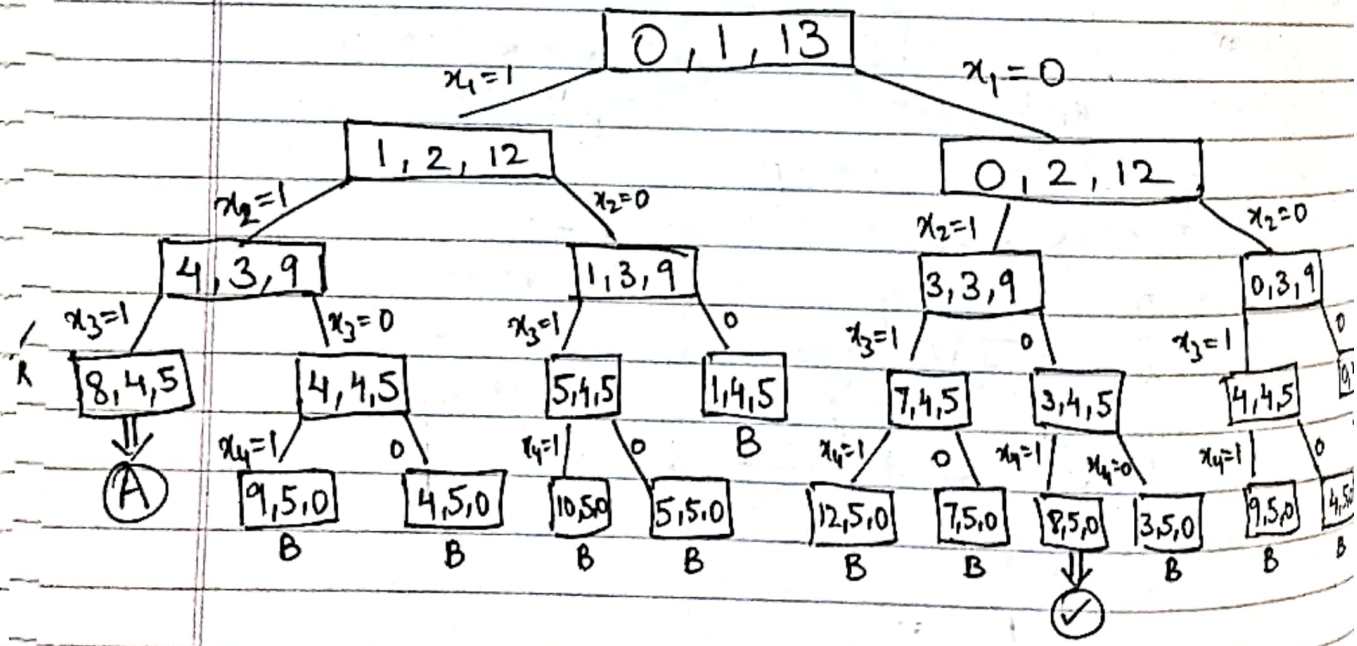
2^n nodes

$W = \{1, 3, 4, 5\}$ and $m = 8$.

State Space Tree



Backtracking Tree



∴ 2 solutions $\{1, 1, 1, 0\}$, $\{0, 1, 0, 1\}$.

Branch & Bound

solves only minimisation problems

0/1 Knapsack.

capacity = 15

Pi	10	30	40	50
Wi	5	4	6	3

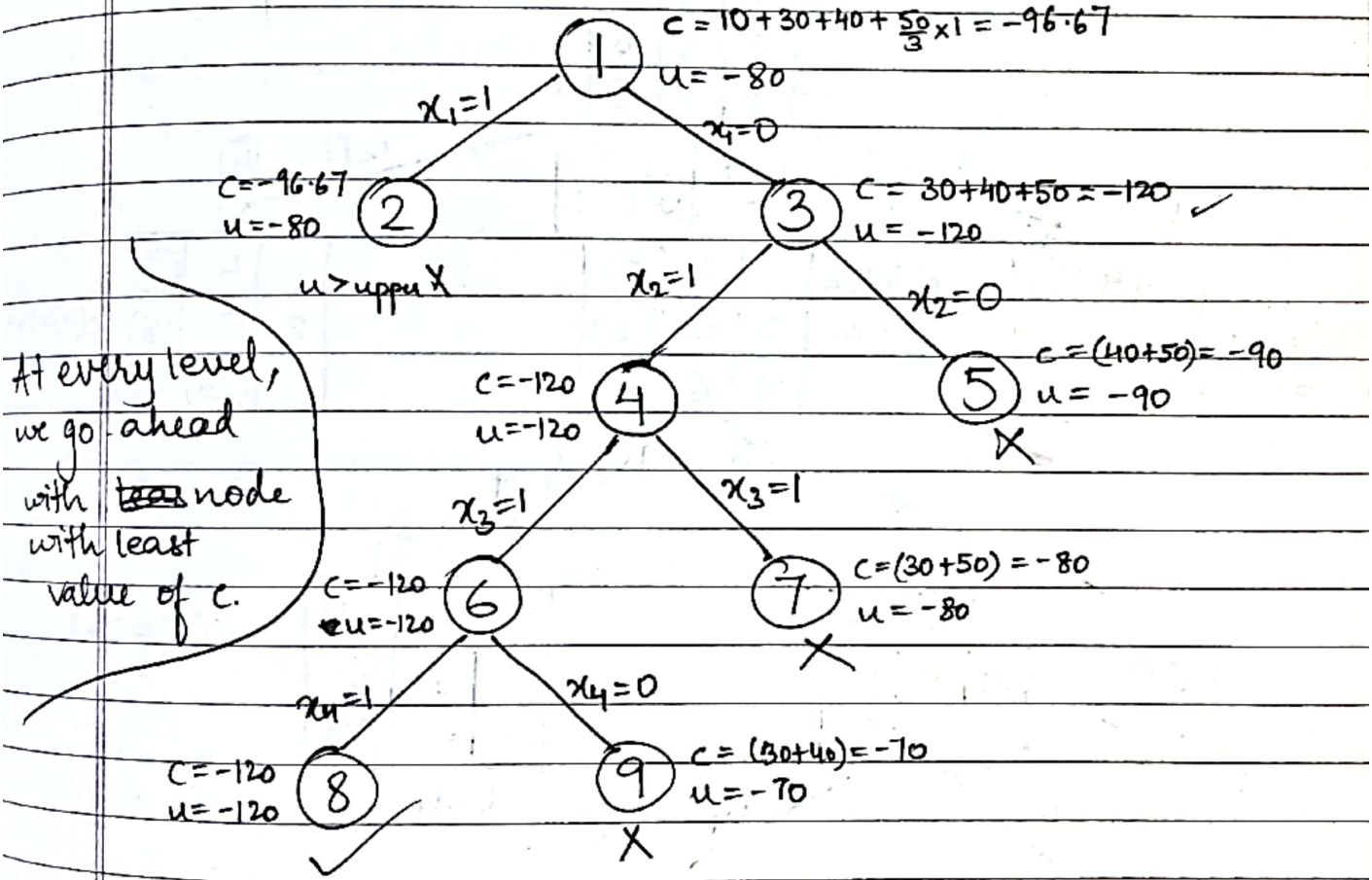
x[] → solution vector

$c = -\sum p_i x_i$ (includes fractional parts)

$u = -\sum p_i x_i$ (excludes " ")

upper = 0 -80 -120

Initially, lets take all objects and proceed,



Solution : {0, 1, 1, 1}

Max Profit → -120

8-Puzzle Problem.

Solved using Least Cost BB. $c(x) = f(x) + g(x)$.

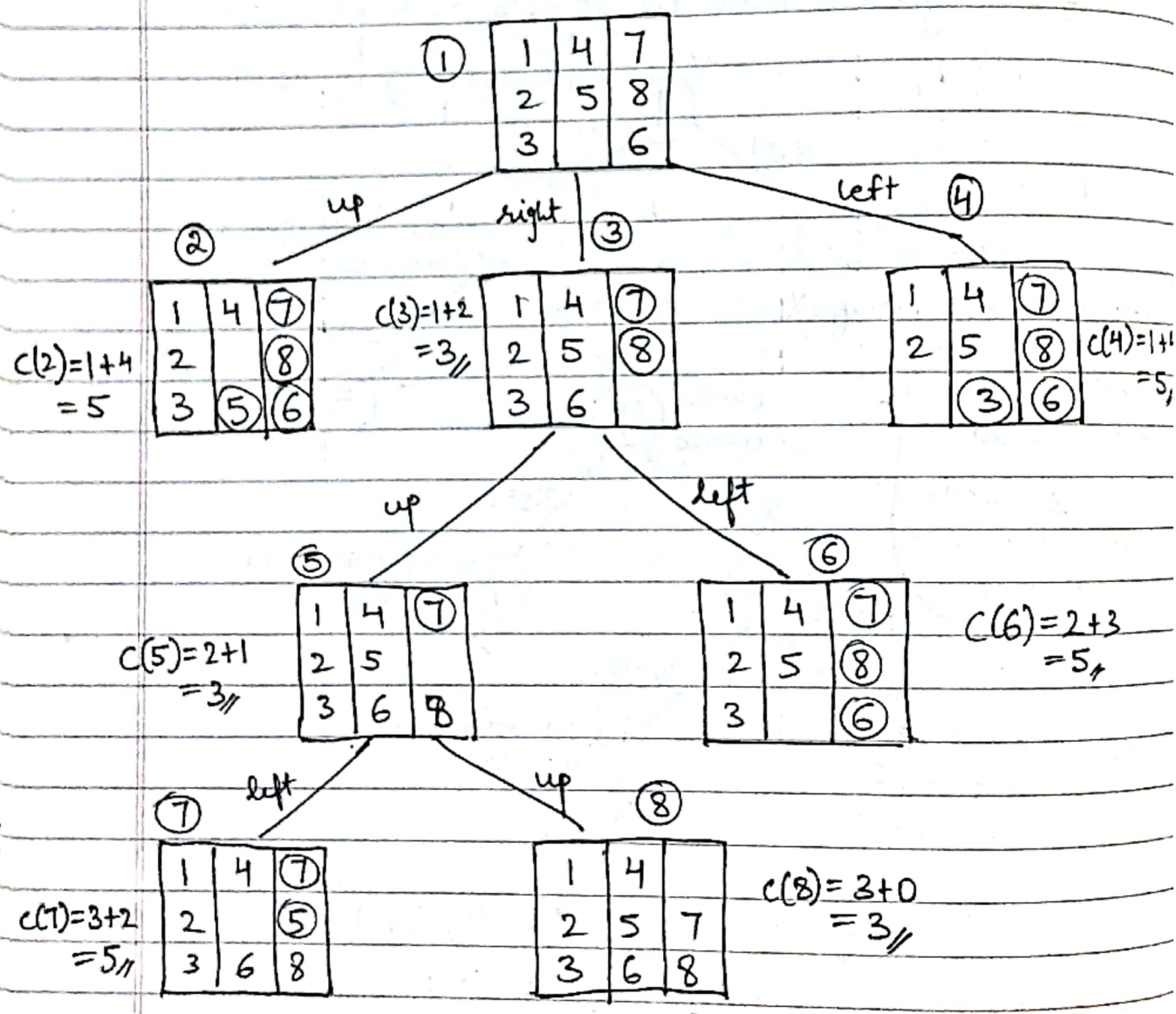
Initial

1	4	7
2	5	8
3		6

Final.

1	4	
2	5	7
3	6	8

State Space.



final state

Dynamic Programming

0/1 Knapsack.

P	3	4	5	6
W	2	3	4	5

$n = 4$.

capacity $(w) = 5$

$$B[k,w] = \begin{cases} B[k-1,w] & , w_k > w \\ \max(B(k-1,w),\ B(k-1,w-w_k)+P_k) & , w_k \leq w \end{cases}$$

no. of items ↓ \ capacity of bag → i \ W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	(7)

∴ Max Profit = (7)

∴ 2nd item in knapsack.

1st item in knapsack.

Soln. {1, 1, 0, 0}

Logic Let $i = n$, $k = w$

```
if (b(i,k) ≠ b(i-1,k))
    mark the ith element in knapsack;
    i = i-1, k = k-w_i
else
    i = i-1;
```

P	4	5	7	10
W	1	3	4	6

$n = 4$, $m = 9$.

Formula:

$$b[k,W] = \begin{cases} 0 & \text{if } i = 0 \text{ or } W = 0 \\ b[k-1,W] & \text{if } W_k > W \\ \max(b[k-1,W], b[k-1, W-W_k] + P_k) & \text{otherwise} \end{cases}$$

i \ W	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	4	4	4	4	4	4	4	4	4
2	0	4	4	5	9	9	9	9	9	9
3	0	4	4	5	9	9	9	12	16	16
4	0	4	4	5	9	9	10	14	16	(16)

Max Profit → (16)

∴ Starting at $b[4,9]$ → let $i = 4, k = 9$

∵ $b[3,9] = b[4,9]$ ⟶ $i = i-1$ (3).

$b[3,9] \neq b[2,9]$ ∴ include 3rd element.

$i = i-1$, $k = k - W_3$

$i = 2$, $k = 5$.

$b[2,5] \neq b[1,5]$ include 2nd element

$i = i-1$, $k = k - W_2$

$i = 1$, $k = 2$

$b[1,2] \neq b[0,2]$ ∴ include 1st element.

Longest Common Subsequence using DP.

Logic. $$LCS[i,j] = \begin{cases} 0 & i,j = 0 \\ 1 + LCS[i-1, j-1] & , A[i] = B[i] \\ \max(LCS[i-1,j], LCS[i,j-1]) & , A[i] \neq B[i] \end{cases}$$

A[] = MANTRALAYA
B[] = MALAYALAM

			M	A	L	A	Y	A	L	A	M
		0	1	2	3	4	5	6	7	8	9
	0	0	0	0	0	0	0	0	0	0	0
M	1	0	1	1	1	1	1	1	1	1	1
A	2	0	1	2	2	2	2	2	2	2	2
N	3	0	1	2	2	2	2	2	2	2	2
T	4	0	1	2	2	2	2	2	2	2	2
R	5	0	1	2	2	2	2	2	2	2	2
A	6	0	1	2	2	3	3	3	3	3	3
L	7	0	1	2	3	3	3	3	4	4	4
A	8	0	1	2	3	4	4	4	4	5	5
Y	9	0	1	2	3	4	5	5	5	5	5
A	10	0	1	2	3	4	5	6	6	6	6

Length of LCS = 6

To obtain LCS, we start Traceback → Retrace the original formula.
Go left till it's same
Diagonal appears then take char.

LCS → MALAYA

Handles -ve edges,
→ Slower than Dijkstras, combo of Dijkstras & -ve edges

Single Source Shortest Path (Bellman Ford)

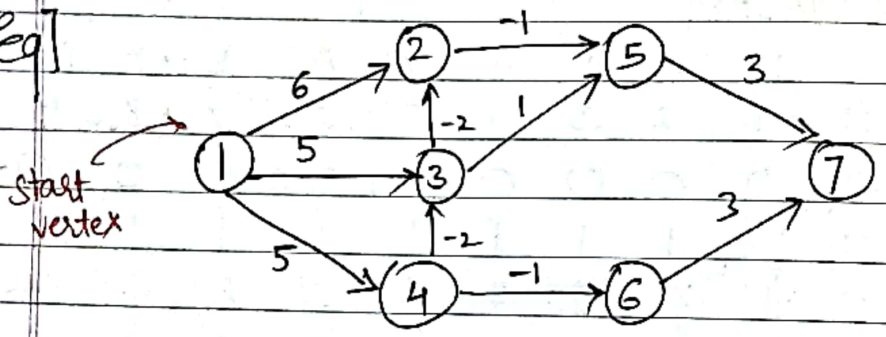
Relaxation formula is that of Dijkstras.

Formula,

$$dist^{k}[u] = \min\left\{dist^{k-1}[u], \min\left\{dist^{k-1}[i] + cost(i,u)\right\}\right\}$$

$dist^{k}[u]$: length of path; u: current vertex

Eg]



Table

k \ V	1	2	3	4	5	6	7
∅							
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	6	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

kitne length ka path hoga betn 2 vertices

For k=2,

$$dist^{2}[2] = \min\left\{dist^{1}[2], \min\left\{\begin{array}{l} dist^{1}[3] + cost(3,2) \\ dist^{1}[4] + cost(4,2) \\ dist^{1}[5] + cost(5,2) \\ dist^{1}[6] + cost(6,2) \\ dist^{1}[7] + cost(7,2) \end{array}\right\}\right\}$$

PAGE No.
DATE / /

$$dist^2[2] = \min\{6, \min\{5+(-2),\ 5+\infty,\ \infty+(-1),\ \infty+\infty,\ \infty+\infty\}\} = 3//$$

$$dist^2[3] = \min\{dist'[3], \min\{dist'[2]+cost(2,3),\ dist'[4]+cost(4,3),\ dist'[5]+cost(5,3),\ dist'[6]+cost(6,3),\ dist'[7]+cost(7,3)\}\}$$

$$= \min\{5, \min\{5+\infty,\ 5+(-2),\ \infty,\ \infty,\ \infty\}\} = 3//$$

$$dist^2[4] = \min\{dist'[4], \min\{dist'[2]+cost(2,4),\ dist'[3]+cost(3,4),\ dist'[5]+''(5,4),\ dist'[6]+''(6,4),\ dist'[7]+''(7,4)\}\}$$

$$= \min\{5, \min\{6+\infty,\ 5+\infty,\ \infty,\ \infty,\ \infty\}\} = 5//$$

$$dist^2[5] = \min\{dist'[5], \min\{dist'[2]+cost(2,5),\ ''[3]+''(3,5),\ ''[4]+''(4,5),\ ''[6]+''(6,5),\ ''[7]+''(7,5)\}\}$$

$$= \min\{\infty, \min\{6+\infty,\ 5+1,\ 5+\infty\}\} = 6//$$

$$dist^2[6] = \min\left\{ dist^1[6], \min\left\{ \begin{array}{l} dist^1[2] + cost(2,6) \\ \quad "\ [3] + "\ (3,6) \\ \quad "\ [4] + "\ (4,6) \\ \quad "\ [5] + "\ (5,6) \\ \quad "\ [7] + "\ (7,6) \end{array} \right. \right.$$

$$= \min\left\{ \infty, \min\left\{ \begin{array}{l} 6+\infty \\ 5+\infty \\ 5+(-1) \\ \infty \\ \infty \end{array} \right. \right\} = 4 //$$

$$dist^2[7] = \min\left\{ dist^1[7], \min\left\{ \begin{array}{l} dist^1[2] + cost(2,7) \\ \quad "\ [3] + "\ (3,7) \\ \quad "\ [4] + "\ (4,7) \\ \quad "\ [5] + "\ (5,7) \\ \quad "\ [6] + "\ (6,7) \end{array} \right. \right.$$

$$= \min\left\{ \infty, \min\left\{ \begin{array}{l} \infty \\ 8 \\ \infty \\ \infty \\ \infty \end{array} \right. \right\} = 8 //$$

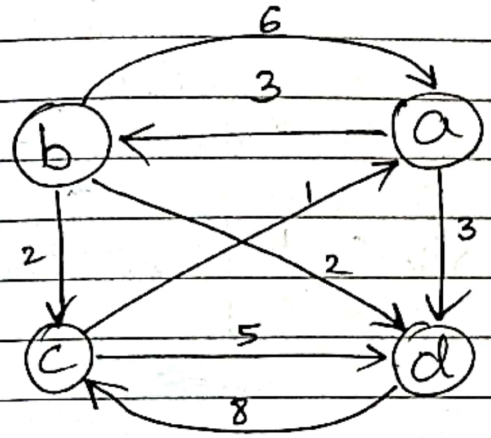
We solve similarly for all values.

& entries in the last row are our lengths of shortest paths. //

All Pair Shortest Path (Floyd Warshall)

Recursive Formula $A^k[i,j] = \min\left(A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j]\right)$

k is treated as intermediate matrix



$$A^0 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 3 \\ 6 & 0 & 2 & 2 \\ 1 & \infty & 0 & 5 \\ \infty & \infty & 8 & 0 \end{bmatrix} \end{array}$$

∴ ~~A~~ a For A^a.

$$A^a = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 3 \\ 6 & 0 & 2 & 2 \\ 1 & 4 & 0 & \cancel{\infty}\,4 \\ \infty & \infty & 8 & 0 \end{bmatrix} \end{array}$$

$A^a[a,b] = \min\left[A^0[a,b], A^0[a,a] + A^0[a,b]\right]$
$= 3$

$A^a[b,d] = \min\left[A^0[b,d], A^0[b,a] + A^0[a,d]\right]$
$= 2 \;\left(< (6+3)\right)$

$c \to a \to b$
$\infty > 1+3$

$$A^b = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 3 \\ 6 & 0 & 2 & 2 \\ 1 & \cancel{\infty}\,4 & 0 & \cancel{\infty}\,4 \\ \infty & \infty & 8 & 0 \end{bmatrix} \end{array}$$

$a \to b \to c$: $\infty > (3+2)$

$a \to b \to d$: $3 < (3+2)$

$$A^c = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 3 \\ 3 & 0 & 2 & 2 \\ 1 & 4 & 0 & 4 \\ 9 & 12 & 8 & 0 \end{bmatrix} \end{array}$$

$$A^d = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 3 \\ 3 & 0 & 2 & 2 \\ 1 & 4 & 0 & 4 \\ 9 & 12 & 8 & 0 \end{bmatrix} \end{array}$$

Always note to check only & only with previous matrix

Multistage Graphs — Forward Approach

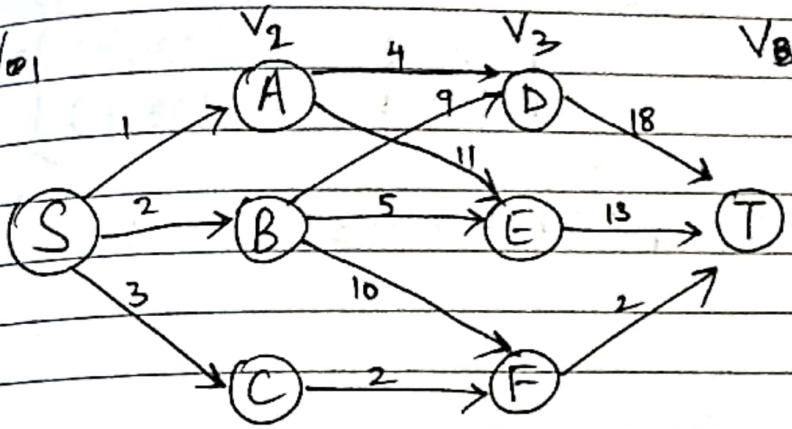
Recursive formula:-

$$cost(i,j) = \min \{ c(j,l) + cost(i+1,l) \}$$

(stage i; $c(j,l)$ = length of path between j & l)

$l \in V_{i+1}$, $\langle j,l \rangle \in E$.

Eg]



V_4 → sink

We start calculating cost of each vertex from the sink (V_4).

V	S	A	B	C	D	E	F	T
cost	9	22	18	4	18	13	2	0
d	C	D	E/F	F	T	T	T	T

(d ← vertex which gives min cost path)

$\therefore$ At V_4 → cost(4, T) = 0 (stage 4, vertex T)

At V_3, $cost(3,D) \rightarrow \min \{ c(D,T) + cost(4,T) \}$

$\rightarrow 18$

Similarly, $cost(3,E) \rightarrow c(E,T) + cost(4,T) = 13$

$cost(3,F) \rightarrow c(F,T) + cost(4,T) = 2$

PAGE No.
DATE / /

At V_2,

$$cost(2,A) = \min \begin{cases} c(A,D) + cost(3,D) \\ c(A,E) + cost(3,E) \end{cases} = \begin{matrix} 4+18 = 22 \checkmark \\ 11+13 = 24 \end{matrix}$$

$$cost(2,B) = \min \begin{cases} c(B,D) + cost(3,D) \\ c(B,E) + cost(3,E) \\ c(B,F) + cost(3,F) \end{cases} = \begin{matrix} 9+18 = 27 \\ 5+13 = 18 \\ 16+2 = 18 \checkmark \end{matrix}$$

$$cost(2,C) = c(C,F) + cost(3,F) = 2+2 = 4 //$$

At V_1

$$cost(1,S) = \min \begin{cases} c(S,A) + cost(2,A) \\ c(S,B) + cost(2,B) \\ c(S,C) + cost(2,C) \end{cases} = \begin{matrix} 1+22 = 23 \\ 2+18 = 20 \\ 5+4 = 9 \checkmark \end{matrix}$$

∴ Joining all the solutions in a forward approach

∴ Shortest Path $\longrightarrow$ S → C → F → T

Length = 9

d(1,S) = 9
d(2,C) = 4
d(3,F) = 2

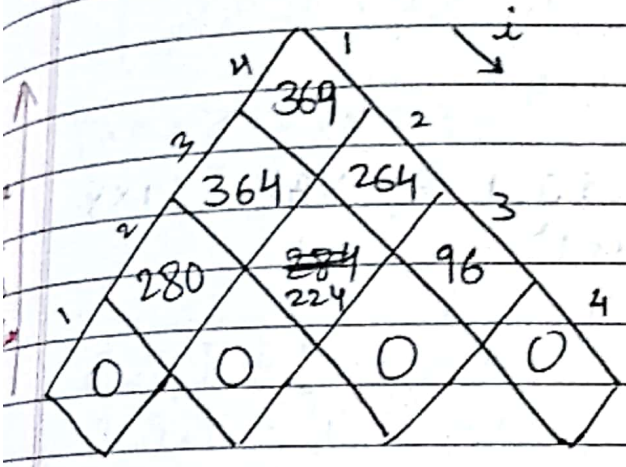
Matrix Chain Multiplication

Recursive Formula :-

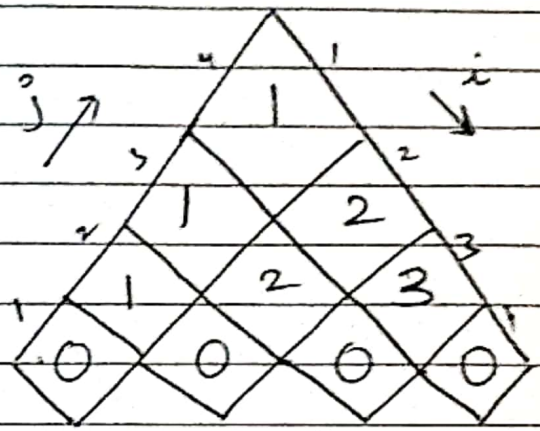
P_i is no. of columns (based on)

$$m[i,j] = \begin{cases} 0, & i = j \\ \min\left(m[i,k] + m[k+1,j] + P_{i-1} P_j P_k\right), & i \le k < j \end{cases}$$

Eg. Four matrices
↳ A_1, A_2, A_3, A_4 with $P_0 = 5, P_1 = 7, P_2 = 8, P_3 = 4, P_4 = 3$



Values



Value of k

Iteration 1 :-

$$m[1,2] = \min_{i \le k < j} \{ m[i,k] + m[k,j] + P_{i-1} P_j P_k \}$$

(k = 1)

$$= \min\{ m[1,1] + m[2,2] + P_0 P_2 P_1 \}$$

$$= 0 + 0 + 5 \times 7 \times 8 = 280 \quad (k = 1)$$

$i=2, j=3$ $\quad m[2,3] = \min_{i \le k < j} \{ m[2,2] + m[3,3] + P_1 P_3 P_2 \}$ $\quad k=2$

$= 0 + 0 + 7 \times 8 \times 4 = 224$ $\quad (k=2)$

$i=3, j=4$ $\quad m[3,4] = \min_{i \le k < j} \{ m[3,3] + m[4,4] + P_2 P_4 P_3 \}$ $\quad k=3$

$= 0 + 0 + 8 \times 4 \times 3 = 96$ $\quad (k=3)$

Iteration 2 :-

$i=1, j=3$ $\quad m[1,3] = \min_{i \le k < j} \begin{cases} m[1,1] + m[2,3] + P_0 P_3 P_1 \\ m[1,2] + m[3,3] + P_0 P_3 P_2 \end{cases}$ $\quad k=1,2$

$= \min \begin{cases} 0 + 224 + 5 \times 4 \times 7 \\ 280 + 0 + 5 \times 4 \times 8 \end{cases}$

$= 364$ $\quad (k=1)$

$i=2, j=4$ $\quad m[2,4] = \min_{i \le k < j} \begin{cases} m[2,2] + m[3,4] + P_1 P_4 P_2 \\ m[2,3] + m[4,4] + P_1 P_4 P_3 \end{cases}$ $\quad k=2,3$

$= \min \begin{cases} 0 + 96 + 7 \times 3 \times 8 \\ 224 + 0 + 7 \times 3 \times 4 \end{cases}$

$= 264$ $\quad (k=2)$

Iteration 3 :-

$i=1, j=4$ $\quad m[1,4] = \min_{i \le k < j} \begin{cases} m[1,1] + m[2,4] + P_0 P_4 P_1 \\ m[1,2] + m[3,4] + P_0 P_4 P_2 \\ m[1,3] + m[4,4] + P_0 P_4 P_3 \end{cases}$ $\quad k=1,2,3$

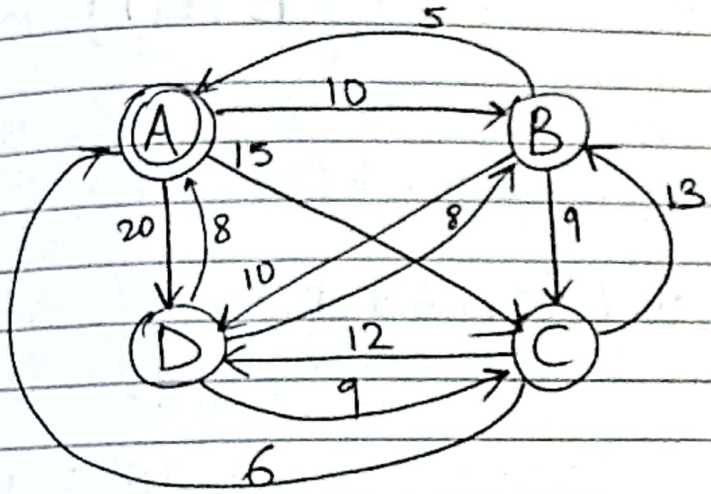
$= 364$ $\quad (k=1)$

Travelling Salesman Problem:

$C(M) =$

	A	B	C	D
A	0	10	15	20
B	5	0	9	10
C	6	13	0	12
D	8	8	9	0

Starting & Ending at A



S → Set of Adjacent vertices.

Formula ☆

$$g[i, \phi] = C_{i1} \quad [\text{cost of edge from } i \text{ to } 1]$$

$$g[i, S] = \min_{j \in S} \{ C_{ij} + g(j, \{S\} - j) \}$$

∴ Initially, $g[B, \phi] = 5 \quad (C_{BA})$

$g[C, \phi] = 6 \quad (C_{CA})$

$g[D, \phi] = 8 \quad (C_{DA})$

$|S| = 1$

$\therefore g[B, \{C\}] = \min\{ C_{BC} + g[C, \phi] \} = 13 + 6 = 19$ //

$g[B, \{D\}] = \min\{ C_{BD} + g[D, \phi] \} = 10 + 8 = 18$ //

$g[C, \{B\}] = \min\{ C_{CB} + g[B, \phi] \} = 13 + 5 = 18$ //

$g[C, \{D\}] = \min\{ C_{CD} + g[D, \phi] \} = 12 + 8 = 20$ //

$g(D,\{B\}) = \min\{C_{DB} + g(B,\phi)\} = 8+5 = 13$

$g(D,\{C\}) = \min\{C_{DC} + g(C,\phi)\} = 9+6 = 15$

$|S|=2$

$$\therefore g(B,\{C,D\}) = \min\begin{cases} C_{BC} + g(C,\{D\}) \\ C_{BD} + g(D,\{C\}) \end{cases} = \begin{matrix} 9+20 = 29 \\ 10+15 = 25 \checkmark \end{matrix}$$

$$g(C,\{B,D\}) = \min\begin{cases} C_{CB} + g(B,\{D\}) \\ C_{CD} + g(D,\{B\}) \end{cases} = \begin{matrix} 13+18 = 31 \\ 12+13 = 25 \checkmark \end{matrix}$$

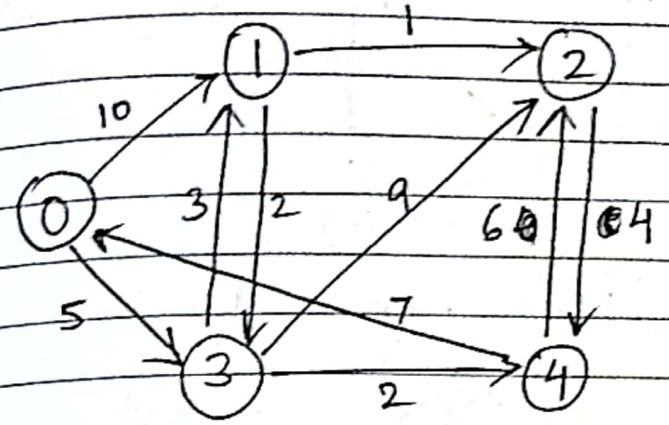
$$g(D,\{B,C\}) = \min\begin{cases} C_{DB} + g(B,\{C\}) \\ C_{DC} + g(C,\{D\}) \end{cases} = \begin{matrix} 8+15 = 23 \checkmark \\ 9+18 = 27 \end{matrix}$$

$|S|=3$

$$g(A,\{B,C,D\}) = \min\begin{cases} C_{AB} + g(B,\{C,D\}) \\ C_{AC} + g(C,\{B,D\}) \\ C_{AD} + g(D,\{B,C\}) \end{cases} = \begin{matrix} 10+25 = 35 \checkmark \\ 15+25 = 40 \\ 20+23 = 43 \end{matrix}$$

∴

Greedy Algos — Dijkstras



Iteration	Path	Vertex Selected	0	1	2	3	4
0		0	0	∞	∞	∞	∞
1	{0}	3	0	10	∞	5	∞
2	{0,3}	4	0	~~8~~	14	5	7
3	{0,3,4}	1	0	8	13	5	7
4	{0,3,4,1}	2	0	8	9	5	7
5	{0,3,4,1,2}	-	0	8	9	5	7

Formula → if (dist(u) + cost(u,v) < dist(v))

↳ then, dist(v) = dist(u) + cost(u,v)

→ Shortest Paths:

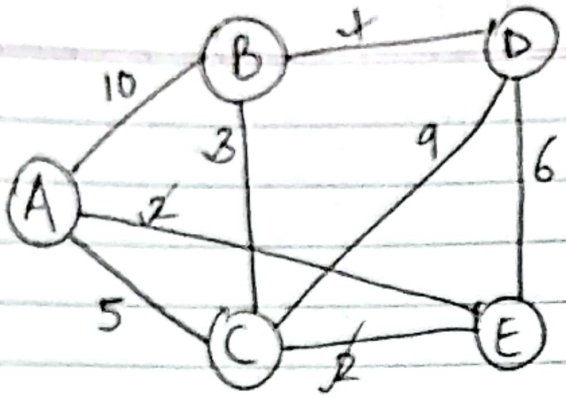
0 to 1 → 0-3-1 = 8

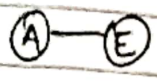
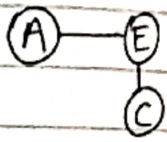
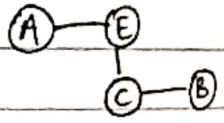
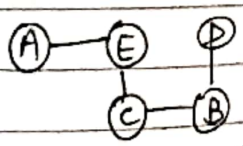
0 to 2 → 0-3-1-2 = 9

0 to 3 → 0-3 = 5

0 to 4 → 0-3-4 = 7

Prims



A	B	Edges taken	Cost	Updated MST
{ }	{A, B, C, D, E}			
{A}	{B, C, D, E}	(A, B), (A, C), (A, E)	2	
{A, E}	{B, C, D}	(A, B), (A, C), (E, D), (E, C)	2	
{A, E, C}	{B, D}	(A, B), (A, C), (E, D), (B, C)	3	
{A, E, C, B}	{D}	(A, B), (A, C), (E, D), (B, D)	1	

MST Cost → 8