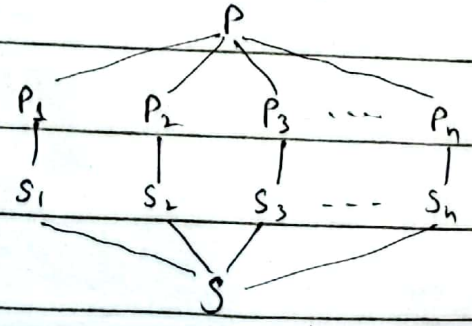
Date: _____

⇒ Design Paradigms / Approaches

→ Divide and Conquer
→ Greedy Algorithm
→ Dynamic Programming
→ Brute Force
→ Branch and Bound
→ Backtracking
→ Genetic Algorithms
→ Randomized Algorithms
→ Approximation Algorithms
→ Simulated Annealing
→ Swarm Intelligence
⋮

① Divide And Conquer:

→ Break/ divide/ split a problem of a larger size into subproblems of smaller size.
→ Obtain the solution of the subproblems
→ Combine/ Merge the solutions of subproblems to get the solution of original larger problem.



Date: _____

(i) Binary Search
(ii) Finding max and min element
(iii) Maximum Subarray Problem
(iv) Mergesort
(v) Quick sort
(vi) Strassan's Multiplications.

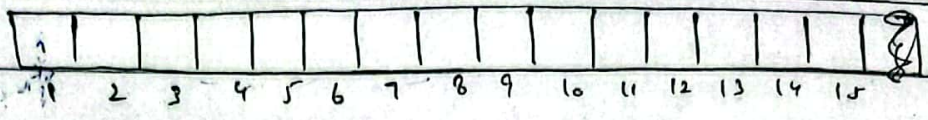
(i) → Binary Search:

→ follows divide & conquer

→ sorted list

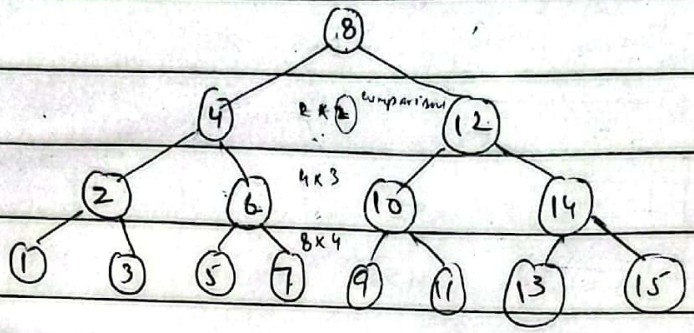
```
int Binary Search (A, n, key)
{  low = 1, high = n
   while (low ≤ high)
   {
      mid = (low + high) / 2
      if (key = A[mid])
         return
      if else (key < A[mid])
         high = mid-1
      else
         low = mid+1
   }
   return 0;
}
```

Date: _____



height = 4

$\log_2 16 = 4$



→ 1
→ 4
→ 12
→ 32

no. of comparisons on each node

Worst Case = O(log n)

Best Case = O(1)

Average case = O(log n)

1 + 4 + 12 + 32 / 15 = 49/15

= 3.2 = ⌈3.2⌉ = 4

Average Case ↑

→ Recursive Binary Search :

```
RBinarySearch ( low, high, key)
{
    if (low == high)
    {
        if (key = A[low])
            return low;
        else
            return 0;
    }
    else
    { Amid = (low + high)/2
        if (key = A[mid])
            return mid
        if (key < A[mid])
            return RBinarySearch (low, mid-1, key)
```

Base case

1

T(n/2)

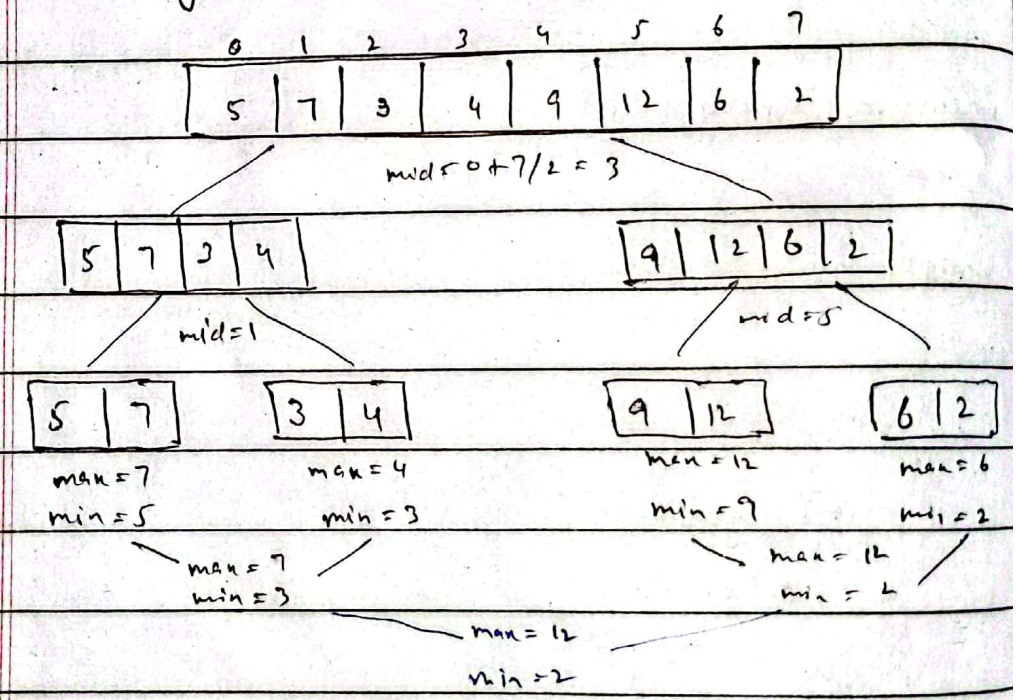
Date: _____

```
else
    return RBinarysearch(mid+1, high, key)
  }
}
```

$$T(n) = \begin{cases} 1 & n=1 \\ T\left(\frac{n}{2}\right)+1 & n>1 \end{cases}$$

$O(\log n)$ ←

(ii) → Finding max and min element.



$$T(n) = \begin{cases} 1 & n=1 \\ 1 & n=2 \\ 2T\left(\frac{n}{2}\right)+1 & n>2 \end{cases}$$

$O(n)$

Date:

```
MaxMin (i, j, Max, Min)                                   → T(n)
{
    if (i=j)                                              Base Case
      { Max = A[i], Min = A[i] }                          → 1
    else if (i=j-1)
      { if (A[i] < A[j])
          {
            Max = A[j]
            Min = A[i]
          }                                               → 1
        else
          {
            Max = A[i]
            Min = A[j]
          }
    }
    else
      { mid = (i+j)/2                                     → 1
        MaxMin (i, mid, Max, Min)                         → T(n/2)
        MaxMin (mid+1, j, Max1, Min1)                     → T(n/2)
        if (Max < Max1)
            Max = Max1                                    → 1
        if (Min > Min1)
            Min = Min1
      }
}
```

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

Date:

iii) Maximum Subarray Problem:

Given an array, find the subarray that has the maximum sum.

Brute force

-1	3	4	-5	9	-2
0	1	2	3	4	5

3 + 4 - 5 + 9 = 11

→ Brute Force: try out all possibilities ⇒ $O(n^2)$

0	1	2	3	4	5
0, 1	1, 2	2, 3	3, 4	4, 5	
0, 1, 2	1, 2, 3	2, 3, 4, 5	3, 4, 5		
0, 1, 2, 3	1, 2, 3, 4	2, 3, 4, 5			
0, 1, 2, 3, 4	1, 2, 3, 4, 5	n-2			
0, 1, 2, 3, 4, 5	n-1				
n					

→ Divide & Conquer Paradigm → 3 possibilities:

- left subarray
- right subarray
- crosses the mid

left = max left
Right = max Right
Crossings = max Crossings
→ Max

Date: _____

```
MaxSubarray (A, low, high)                                  → T(n)
    if (low = high)                                         → 1
        return A[low]                                       → 1
    mid = (low + high) / 2                                  → 1
    left = MaxSubarray (A, low, mid)                        → T(n/2)
    Right = MaxSubarray (A, mid+1, high)                    → T(n/2)
    Crossing = MaxSubarray (A, low, mid, high)              → O(n)
    return max(left, right, crossing)
```

$$T(n) = 2T(n/2) + \underbrace{n+1}_{n}$$

$$T(n) = \begin{cases} 1 & n=1 \\ 2T(n/2)+n & n>1 \end{cases} \Rightarrow O(n \log n)$$

(iv) ⇒ Merge Sort: (Stable)

A process of combining two sorted lists into a single sorted list. (m-way merging)

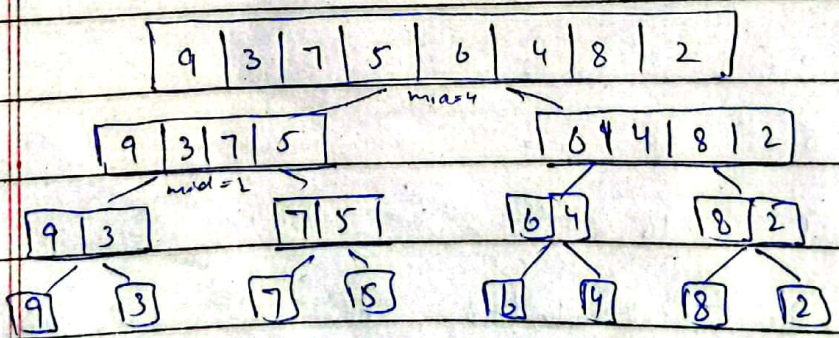
(m) A	(n) B	C	D
4	3	8	2
6	5	10	4
12	9	16	18

3, 4, 5, 6, 9, 12 — 2, 4, 8, 10, 16, 18

O(m+n)

Date:

```
Algorithm Merge (A, B, m, n)
{
  i=1, j=1, k=1;
  while (i<=m && j<=n)
    {
      if (A[i] < B[j])
        c[k++] = A[i++]
      else
        c[k++] = B[j++]
    }
  for ( ; i<=m ; i++)
    c[k++] = A[i]
  for ( ; j<=n ; j++)
    c[k++] = B[j]
}
```



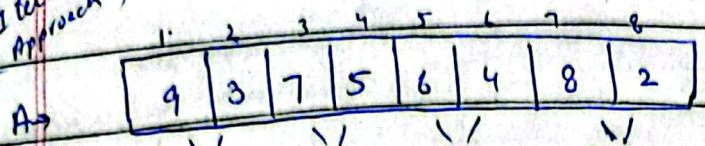
8 elements $\log 8 = 3$

So, 3 iterations

Date:

(Iterative Approach)

(two-way-merging)



$\log n$ iterations → 1st → n

3,9 5,7 4,6 2,8 → 2nd → n

3,5,7,9 2,4,6,8 → 3rd → n

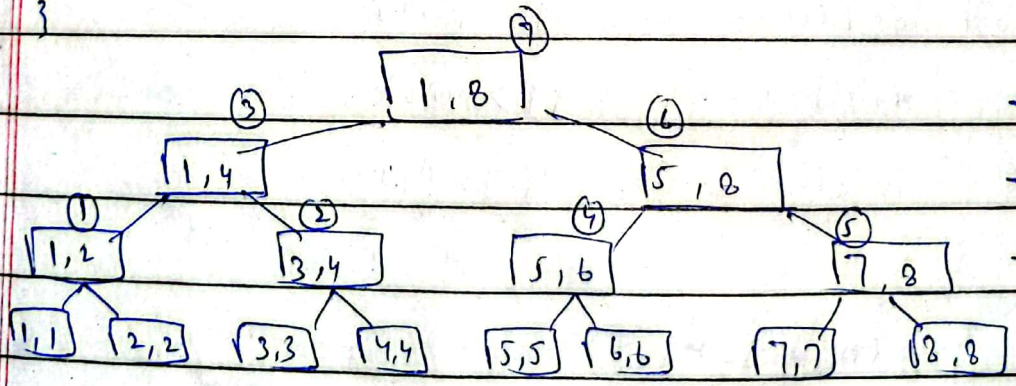
2,3,4,5,6,7,8,9

$\Theta(n \log n)$

sorting dealing with n(8) elements in each iteration/level

⇒ Merge Sort (Recursive Approach)

```
MergeSort (low, high)                    → T(n)
{
  if (low < high)                        → 1
  {
    mid = (low + high) / 2               → 1
    MergeSort (low, mid)                 → T(n/2)
    MergeSort (mid+1, high)              → T(n/2)
    MergeSort (low, mid, high)           → n
  }
}
```



→ 1st

→ 2nd

→ 3rd

$\Theta(n \log n)$

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

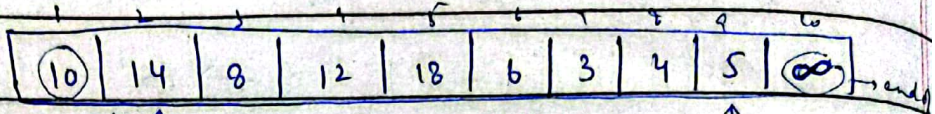
$\Theta(n \log n)$

Date:

(v) ⇒ Quick Sort

elements on left side → smaller
elements on right side → bigger

A →



pivot = 10

i → search for element greater than pivot.

j → search for element smaller than pivot.

i = 2 , j = 9 → Now swap 10, 5, 8, 12, 18, 6, 3, 4, 14, ∞

i = ~~8~~ 4 , j = 8 → swap 10 5 8 4

i = 5 , j = 7 → swap

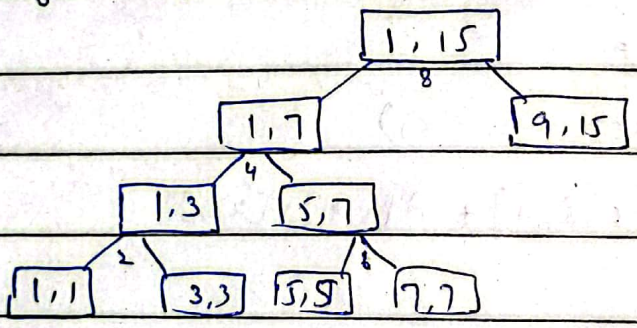
```
Partition ( low, high)
{   pivot = A[low]   i = low   j = high
while (i<j)
  {  do { i++} while( A[i] <= pivot)
     do (j--) while (A[j] > pivot)
     if (i<j) { swap (A[i], A[j])}
  }
  Swap(A[low], A[j])
  return j;
}
```

Date:

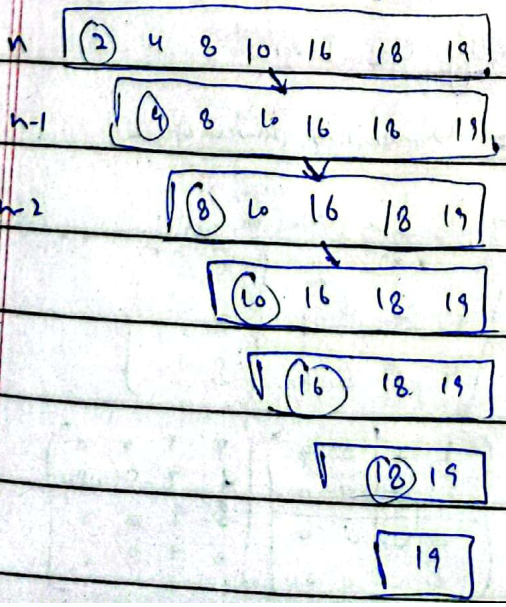
```
Quicksort ( low, high)                                  → T(n)
{
    if ( low < high)
    {
        j = Partition ( low, high)
        Quicksort (low, j )
        Quicksort ( j+1 , high)
    }
}
```

→ Analysis : Assume pivot is sorted in the middle (median) (Best case)

(Array is already sorted) $\theta(n \log n)$



→ Worst case : If array is already sorted in ascending order | pivot is first element



$1 + 2 + \dots + (n-1) + n$

$\frac{n(n+1)}{2} \Rightarrow O(n^2)$

Date:

Matrix Multiplication

Using Divide & Conquer Paradigm

r_1, c_1 / r_2, c_2

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

$$c_{ij} = \sum_{k=1}^{n^1 x 2} A_{ik} B_{kj}$$

A B C

$2 \times \boxed{2} \quad \boxed{2} \times 2 \quad 2 \times 2$

i k j

Base Cases: 2x2 matrix, 1x1 matrix

```
for ( i=0 ⟶ r1
  {
    for ( j=0 ; ⟶ c2 )
      {
        c[i,j] = 0
        for ( k=0 ⟶ c1 )
          {
            c[i,j] + A[i,k] * [k,j]
          }
```

Base Case $\Theta(1)$

$c_{11} = a_{11} b_{11} + a_{12} b_{21}$

$c_{12} = a_{11} b_{12} + a_{12} b_{22}$

$c_{21} = a_{21} b_{11} + a_{22} b_{21}$

$c_{22} = a_{21} b_{12} + a_{22} b_{22}$

→ Non square ⟶ to ⟶ Square Matrix

Padding Operation:

→ Dimensions power of 2

→ Square Matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \begin{bmatrix} 4 & 5 \\ 6 & 7 \\ 8 & 9 \end{bmatrix} \longrightarrow 2 \times 2$$

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 4 & 5 & 0 & 0 \\ 6 & 7 & 0 & 0 \\ 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Date:

$n=4$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

A_{11} A_{12} A_{21} A_{22} B_{11} B_{12} B_{21} B_{22}

```
MM(A, B, n)
{
    if(n<=2)
    { 4 instructions.
```

```
else
{
```

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

$MM(A_{11}, B_{12}, n/2) + MM(A_{12}, B_{22}, n/2)$

$MM(A_{21}, B_{11}, n/2) + MM(A_{22}, B_{21}, n/2)$

$MM(A_{21}, B_{12}, n/2) + MM(A_{22}, B_{22}, n/2)$

Date:

col matrices are adding

$$T(n) = \begin{cases} 1 & n <= 2 \\ 8T(n/2) + n^2 & n > 2 \end{cases}$$

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

$= 8\left[8T(n/4) + (n/2)^2\right] + n^2$

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

$= 8^2\left\{8T(n/8) + (n/4)^2\right] + 3n^2$

$= 8^3 T(n/2^3) + 7n^2$

$= 8^k T(n/2^k) + (2^k - 1)n^2$

$n/2^k = 1 \implies k = \log n$

$= 8^{\log n} T\left(n/2^{\log n}\right) + (2^{\log n} - 1)n^2$

$= n^{\log 8} T\left(n/n^{\log 2}\right) + (n^{\log 2} - 1)n^2$

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

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

$= n^3 + n^3 - n^2$

$= 2n^3 - n^2$

$\implies \Theta(n^3)$

Date:

⇒ Strassen's Multiplication

→ reduce # of multiplications 8 → 7

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

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

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

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

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

$U = (A_{22} - A_{11}) \cdot (B_{11} + B_{12})$

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

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

$C_{12} = R + T$

$C_{21} = Q + S$

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

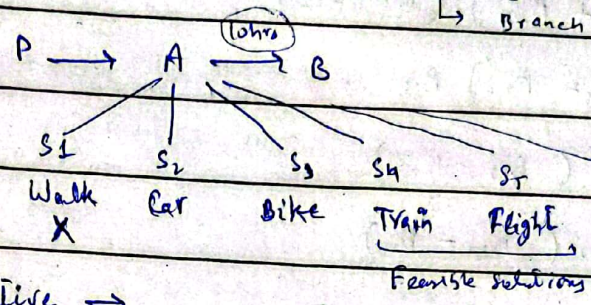
$T(n) = 7T(n/2) + n^2$ $\quad \Theta(n^{\log 7})$

$\Theta(n^{2.81})$

Date:

⇒ Greedy Paradigm / Approach → (∵ does not always give optimal solution guarantee)

- A design for solving a problem
- used for optimization problems < Maximization / Minimization
  - Dynamic Programming
  - Branch And Bound



Objective →

(1ohrs) Constraints →

Only 1 optimal solution

- A problem should be solved in stages.

→ Knapsack Problem,

	X	$\frac{1}{X_1}$	$\frac{2/3}{X_2}$	$\frac{1}{X_3}$	X_4	$\frac{1}{X_5}$	$\frac{1}{X_6}$	$\frac{1}{X_7}$
Objects	0	1	2	3	4	5	6	7
Profits	P	10	5	15	7	6	18	3
Weight	W	2	3	5	7	1	4	1

n=7
m=15

Constraint → Don't exceed the limit of bags

Objective → To maximize profit.

← That's why we don't use it ← because some space will remain in the bag ← it is not ideal ← i) 0/1 knapsack prob

i) Fractional knapsack.

(1)	P/W →	5 (4)	1.67	3 (2)	1	6 (1)	4.5 (3)	3 (5)

15-1 = 14
14-2 = 12
12-4 = 8
8-5 = 3

3-1 = 2
2-2 = 0

$\sum x_i w_i \leq m$ (Constraint)

Date:

$\sum x_i w_i = (1\times2) + (2/3\times3) + (1\times5) + (0\times7) + (1\times1) + (1\times4) + (1\times1)$

$= 2 + 2 + 5 + 0 + 1 + 4 + 1$

$\checkmark = 15$

Max. Profit $= \sum_{i=1}^{n} x_i P_i$

$= (1\times10) + (2/3\times5) + (1\times15) + (0\times7) + (1\times6) + (1\times18) + (1\times3)$

$= 55.3$ → optimal solution

(2) ∴ Maximum Profit Approach

	X	X_1 (1)	X_2 (1)	X_3 (1)	X_4 (4/5)	X_5 (1)	X_6 (1)	X_7 (1)
Objects	0	1	2	3	4	5	6	7
Profits	P	10	5	15	7	6	18	3
weight	W	2	3	5	7	1	4	1

(3) ∴ Minimum Weight Approach

P→ 15 − 4 = 11, 11 − 5 = 6, 6 − 2 = 4, 4 − 4 = 0

W→ 15 − 1 = 14, 14 − 1 = 13, 13 − 2 = 11, 11 − 3 = 8, 8 − 4 = 4, 4 − 4 = 0 (5)

$\sum x_i w_i = (1\times2) + (0\times3) + (1\times5) + (4/7\times7) + (0\times1) + (1\times4) + (0\times1)$

$= 2 + 5 + 4 + 4$

$\checkmark = 15$

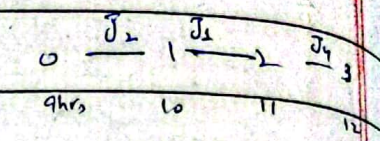
Max Profit $= \sum x_i P_i$

$= (1\times10) + (0\times5) + (1\times15) + (4/7\times7) + (0\times6) + (1\times18) + (0\times3)$

$= 10 + 15 + 4 + 18 = 47$ → other optimal solution

Date:

⇒ Job sequencing with deadlines



Jobs	J_1	J_2	J_3	J_4	J_5
Profits (Descending order)	20	15	10	5	1
Deadlines	2	2	1	3	3
			×		×

J_2 J_1 J_4

$15 + 20 + 5 = 40$

Constraint : Time slot

Objective : Maximizing Profit

Jobs	Slot Assigned	Solution	Profit
J_1	[1,2]	$\{J_1\}$	20
J_2	[0,1][1,2]	$\{J_1, J_2\}$	20+15
J_3	[0,1][1,2]	$\{J_1, J_2\}$	20+15
J_4	[0,1][1,2][2,3]	$\{J_1, J_2, J_4\}$	20+15+5
J_5	"	"	" = 40

Date:

Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profits	35	30	25	10	15	40	45
Deadlines	3	4	4	2	3	1	2

Jobs	J_7	J_6	J_1	J_2	J_3	J_5	J_4
Profits	45	40	35	30	25	15	10
Deadlines	2	1	3	4	4	3	2

0 J_6 1 J_7 2 J_1 3 J_2 4

Date:

Activity Selection:

Given Activities (lectures) and one room. Now schedule as many activities as possible such that they do not conflict with each other.

	i	1	2	3	4	5	6	7	8	9
	s_i	2	2	4	1	5	8	9	11	13
	f_i	3	5	7	8	9	10	11	14	16
duration	d_i	1	3	7	7	4	2	2	3	3
By difference	App 1	✓	X	✓	X	X	✓	X	✓	X
By Low Ending Time	App 2	✓	X	✓	X	X	✓	X	✓	X
		X	✓	X	X	✓	X	✓	X	✓

Total Subsets $= 2^9$

Solution:

$\{a_1, a_3, a_6, a_8\}$ → Optimal Solution

It will be difficult to achieve optimal solution if there are more conflicts.

Date:

i	1	2	3
S_i	4	1	5
F_i	6	5	16
di	2	4	5
	✓	✗	✗
	✗	✓	✓

Greedy solution = $\{a_1\}$ Approach 1

Optimal solution = $\{a_2, a_3\}$ Approach 2

1- Min Duration ✗

2- Min Start Time ✗

3- Min End Time ✓ → Optimal Solution.

4- Max Start Time

⇒ Activity Selection Algorithm

∴ S (start) & F (finish) are array

∵ n = no. of activities

```
Greedy AS (S, F, n)
// sort activities in both arrays by finish time
   in ascending order.        ↳ (n log n)
   add A[0] to A
   for (i = 1 to i = n-i)      → n
      k = 1
```

Date:

if $s[i] \geq F[k]$

Add $A[i]$ to A

$k = i$

Time Complexity = $n + n \log n \Rightarrow O(n \log n)$

Date: _____

⇒ Optimal Merge Pattern

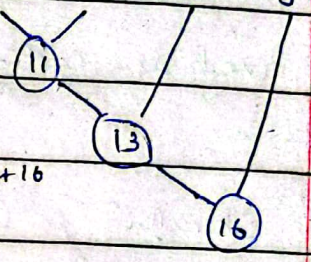
A	B	C
3	5	
8	9	
12	11	
20	16	
n	m	

⇒ O (n time)

if more than two lists

List A B C D

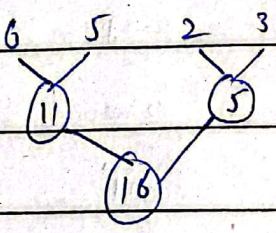
Sizes 6 5 2 3



→ Cost = 11 + 13 + 16
= 40

List A B C D

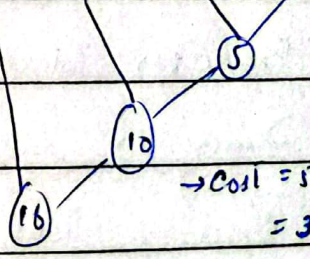
Sizes 6 5 2 3



→ Cost = 11 + 5 + 16
= 32

List A B C D

Sizes 6 5 2 3

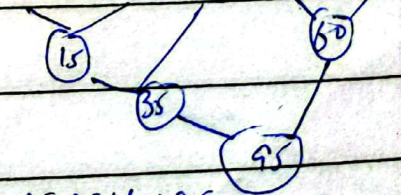


→ Cost = 5 + 10 + 16
= 31

Lists	x_1	x_2	x_3	x_4	x_5
Sizes	20	30	10	5	30

Ascending Order

x_4	x_3	x_1	x_2	x_5
5	10	20	30	30



$\sum d_i x_i = (3 \times 5) + (3 \times 10) + (2 \times 20) + (3 \times 30) + (2 \times 30)$

= 205

Cost = 15 + 35 + 60 + 95
= 205

Date:

⇒ Huffman Coding:

→ a compression technique

message → BCCA BB DDA ECC BBA E DDCC

Length = 20 ⟶ Total bits = 20 × 8 = 160 bits

A = 65 → 01000001

B = 66

⇒ Fixed size Coding:

A	111
B	000
C	001
D	010
E	011

Character	Count/Frequency	Code
A	3	000
B	5	001
C	6	010
D	4	011
E	2	100
	8B=40	15 bits (3×5)

Encode: → Total bits = 20 × 3 = 60

Decode: → Total bits = 40 + 15 = 55

Total = 60 + 55 = 115 bits

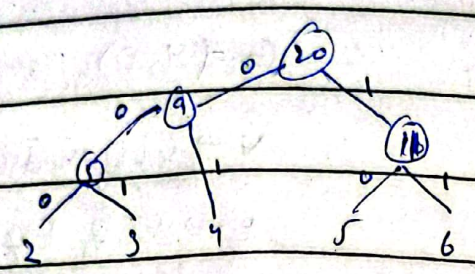
Date:

Huffman coding

⇒ Variable size coding

→ follow optimal merge pattern

→ give small size code to frequently appearing characters.



E A D B C

→ first arrange in ascending order.
→ then left side '0' right side '1'

Character	Count / Freq	Code	
A	3	001	9 (3×3)
B	5	10	10 (5×2)
C	6	11	12 (6×2)
D	4	01	8 (4×2)
E	2	000	6 (2×3)

5×8 = 40 bits

12 bits = 45 bits (Encoding)

Decoding = 40 + 12 = 52 bits

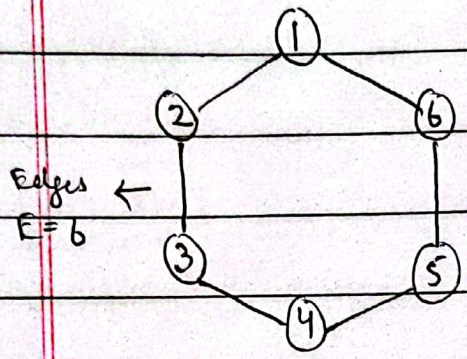
Total = 52 + 45 = 97 bits.

Date:

Greedy Algorithm

Minimum Cost Spanning Tree

↳ a subgraph of graph.



$G = (V, E)$

V → set of vertices/nodes

E → set of Edges

$V = \{1, 2, 3, 4, 5, 6\}$

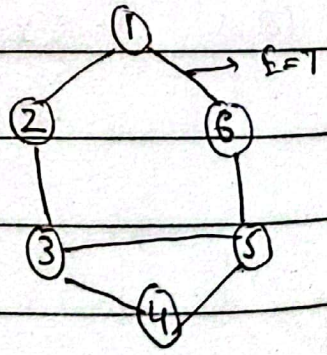
$E = \{(1,2), (2,3), (3,4), (4,5), (5,6), (6,1)\}$

→ Vertices of the graph with $|V|-1$ Edges and no cycle

$6C_5 = 6$ ⟶ Out of 6 Edges, we have to choose 5.

$$\frac{6!}{5!(6-5)!} = \frac{6 * 5!}{5!\,1!} = \frac{6}{1} = 6$$

$$nC_r = \frac{n!}{r!\,(n-r)!}$$



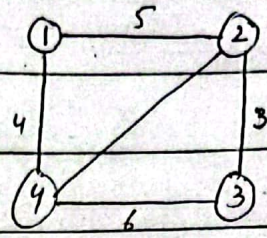
$7C_5 - 2$ ⇐

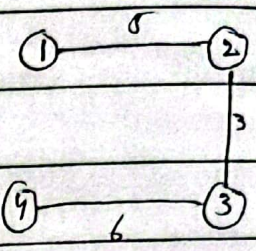
↓

represents no. of cycles.

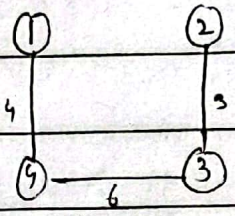
Date:

Weighted Graphs:

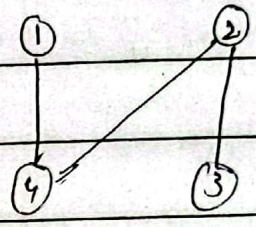




Cost = 14



Cost = 13

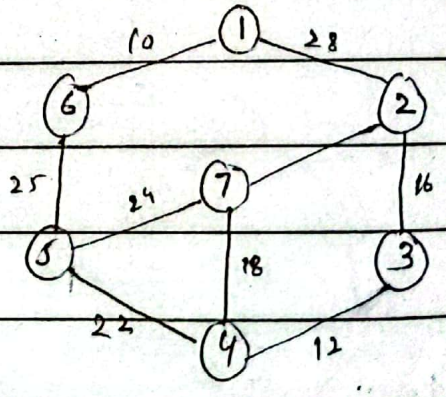


Cost = 9

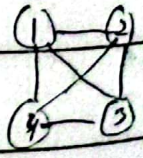
How will we extract minimum spanning tree?

→ Prim's Algorithm

→ Kruskal's Algorithm

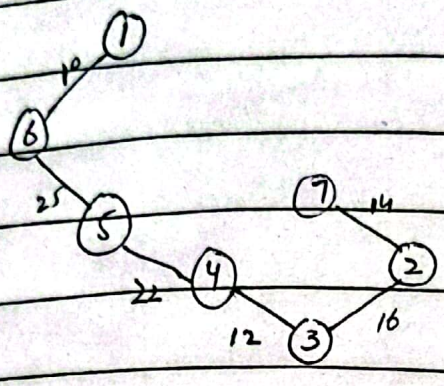


One graph consists of $\frac{n(n-1)}{2}$ edges



Date:

⇒ Prim's Algorithm



n = 7 → all vertices ✓

E = 6 → V-1

cost = 10+25+22+12+16+14

= 99

⇒ Kruskal's Algorithm

- weights in ascending order

(1, 6) → 10

(3, 4) → 12

(2, 7) → (14)

(2, 3) → 16

(4, 7) → 18 X

it will make circle

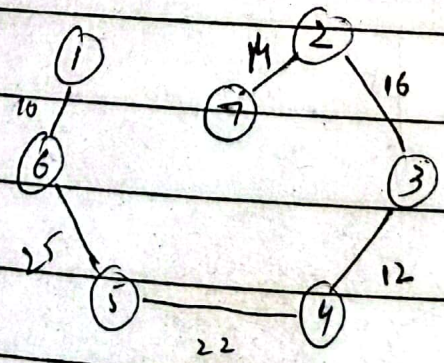
so, we will not consider it

(4, 5) → 22

(5, 7) → 24 X

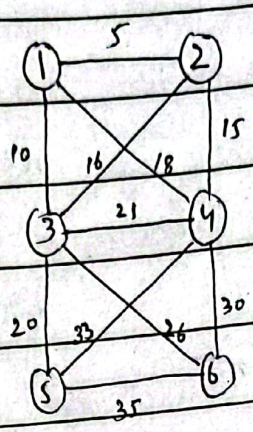
(5, 6) → 25

(1, 2) → 28 X



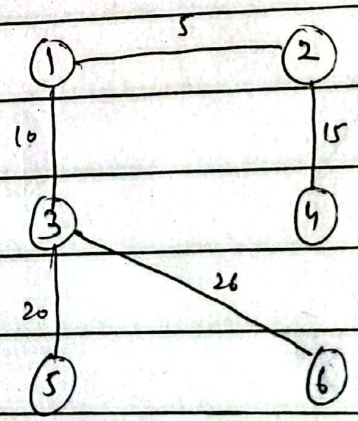
COST = 99

Date:



⇒ Kruskal's Algorithm

✓ (1, 2) → 5
✓ (1, 3) → 10
✓ (2, 4) → 15
✗ (2, 3) → 16
✗ (1, 4) → 18
✓ (3, 5) → 20
✗ (3, 4) → 21
✓ (3, 6) → 26
✗ (4, 6) → 30
✗ (4, 5) → 33
✗ (5, 6) → 35

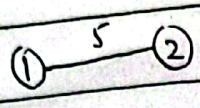


Cost = 5 + 10 + 15 + 20 + 26 = 76

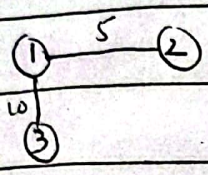
Date:

⇒ Prim's Algorithm

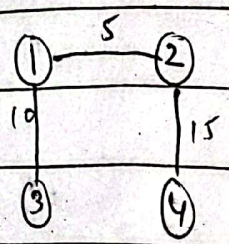
(i)



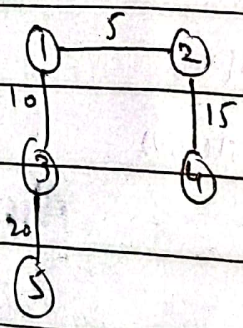
(ii)



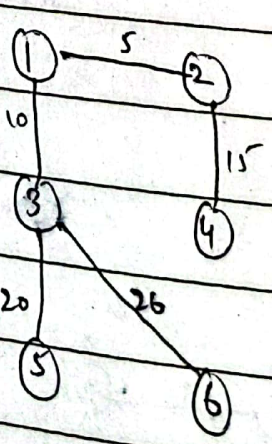
(iii)



(iv)



(v)

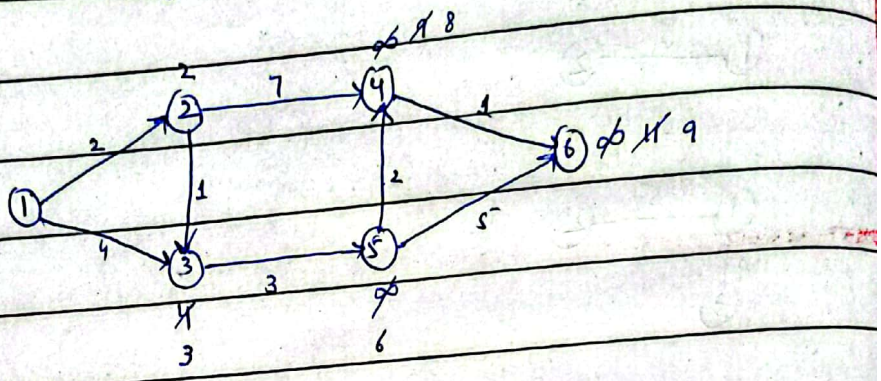


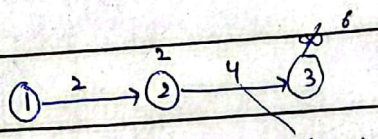
⇒ Cost = 5+10+15+20+26 = 76

Date:

Greedy Algorithms

⇒ Dijkstra Algorithm — single source, shortest path





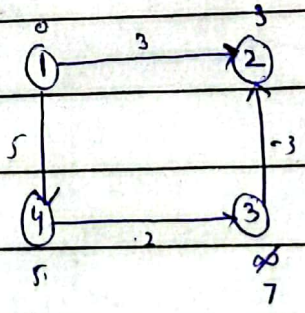
Updation

$$if\ (d[u] + cost(u,v) < d[v])$$

$$d[v] = d[u] + cost(u,v)$$

smallest chosen vertex no. 2

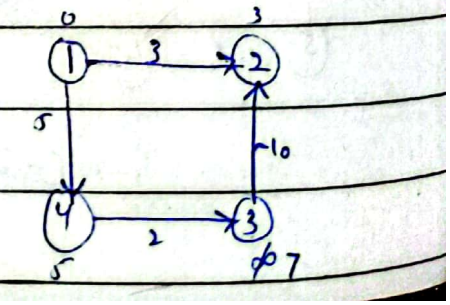
no outlink of 2 so, no updation



There is a possibility it will not show optimal solution in negative weights

$7 - 10 = -3 < 3$

but if is already chosen



Date: _____

⇒ Dynamic Programming

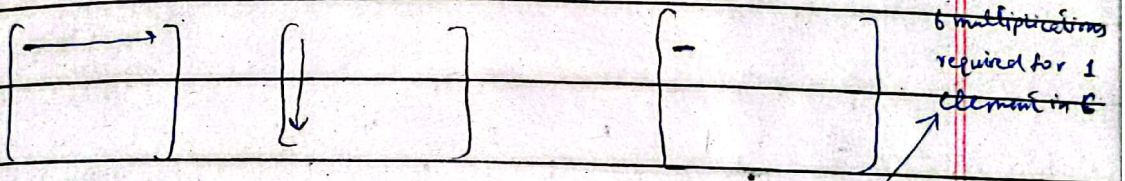
- used for optimization problem
- exhaustive search in a controlled way.

→ Matrix chain Multiplication

A_1 A_2 A_3 A_4

3×5 5×3 3×8 8×6

A B = C



5×6 6×8 5×8 × 6 = 240

6 multiplications required for 1 element in C

X Y Z

3×4 4×5 5×6

3×5 5×6

(3×5)×4 (3×6)×5

(XY)Z = 60 + 90 = 150

X(YZ) = 120 + 72 = 192

(4×4)×5 (3×6)×4

3×4 4×6

3×4 4×6

18
5
24
5
110

Date:

5x6 6x4 4x2 2x7

A_1 A_2 A_3 A_4

5x4 4x6 6x2 2x7

d_0 d_1 d_2 d_3 d_4

$M_{1,2} = A_1 \times A_2$

$M_{2,3} = A_2 \times A_3$

M	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

used for min number of multipl

S	1	2	3	4
1			1	3
2				3
3				
4				

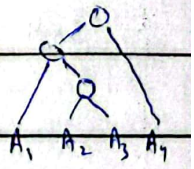
∴ value of k

best possible or

where the condi

breaks.

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



two matrices

$m[1,2] = 120$

$m[2,3] = 48$

$m[3,4] = 84$

3 matrices

A_1, A_2, A_3 → $(A_1A_2)A_3$ or $A_1(A_2A_3)$

$m[1,3]$

2 possibilities

5x6 6x2

$m[1,3] \rightarrow (A_1A_2)A_3$

$= 120 + 0 + 60 = 180$

5x4 4x2

$m[1,3] = A_1(A_2A_3)$

$= 0 + 48 + 40 = 88$

k no. of possibilities
condition k break
ठीक

Date: _____

$$m[i,j] = \min \begin{cases} m[i,k] + m[k+1,j] + d_{i-1} d_k d_j \\ i \le k \le j \end{cases}$$

$1 \le k < 3$

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + (5 \times 4 \times 2) & k=1 \\ m[1,2] + m[3,3] + (5 \times 6 \times 2) & k=2 \end{cases}$$

$$= \min \begin{cases} 0 + 48 + 40 = 88 \\ 120 + 0 + 60 = 180 \end{cases}$$

$$m[1,3] = 88$$

$2 \le k < 4$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + (4 \times 6 \times 7) & k=2 \\ m[2,3] + m[4,4] + (4 \times 2 \times 7) & k=3 \end{cases}$$

$$= \min \begin{cases} 0 + 84 + 168 = 252 \\ 48 + 0 + 56 = 104 \end{cases}$$

$$m[2,4] = 104$$

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

$=$

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

$=$

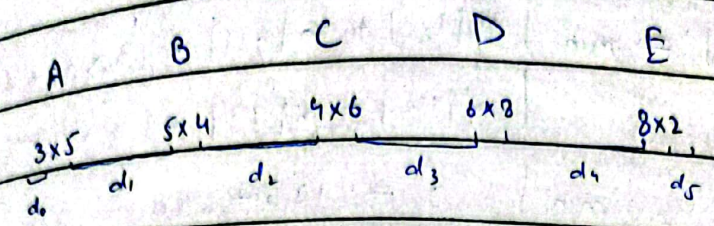
Date:

$$m[1,4] = \min_{1 \le k < 4} \begin{cases} m[1,1] + m[2,4] + (5 \times 4 \times 7) \\ m[1,2] + m[3,4] + (5 \times 6 \times 7) \\ m[1,3] + m[4,4] + (5 \times 2 \times 7) \end{cases}$$

$$= \min \begin{cases} 0 + 104 + 140 = 244 \\ 120 + 84 + 210 = 414 \\ 88 + 0 + 158 = (158) \end{cases}$$

$m[1,4] = 158$

Date: _____



m	1	2	3	4	5
1	0	60	132		
2		0	120		
3			0	192	
4				0	96
5					0

S	1	2	3	4	5
1			2		
2					
3					
4					
5					

Date:

$$m[1,3] = \min_{1 \le k < 3} \begin{cases} m[1,1] + m[2,3] + (3 \times 5 \times 6) \\ m[1,2] + m[3,3] + (3 \times 4 \times 6) \end{cases}$$

$$= \min \begin{cases} 0 + 120 + 90 = 210 \\ 60 + 0 + 72 = 132 \end{cases}$$

$m[1,3] = 132$

Date:

Dynamic Programming

- Identify and solve subproblems
- Use optimal substructure to solve larger problem.
  ↳ optimal solution to subproblem

⇒ 0/1 knapsack problem: maximization problem $m = 8$, $n = 4$

bag capacity w

	P_i	w_i	0	1	2	3	4	5	6	7	8
			0	0	0	0	0	0	0	0	0
	1	2	0	(0)	1	1	1	1	1	1	1
2→	2	3	0	0	1	(2)	2	3	3	3	3
	5	4	0	0	1	(2)	5	5	6	7	7
4→	6	5	0	0	1	2	5	6	6	7	(8)

← populating row & column — used for backtracking

y_1	y_2	y_3	y_4
0	1	0	1

$8 - 6 = 2$

$2 - 2 = 0$

$$x[i, w] = \max \{ x[i-1, w], x[i-1, w - w[i]] + P[i] \}$$

$$x[4, 1] = \max \{ x[3, 1], x[3, 1-5] + 6 \}$$

So this will be the max ← $x[3, -4]$ undefined

$$x[4, 5] = \max \{ x[3, 5], x[3, 5-5] + 6 \}$$

$$= \max \{ x[3, 5], x[3, 0] + 6 \}$$

$$= \max \{ 5, 6 \}$$

$$= 6$$

Date:

$x[4,6] = \max\{x[3,6], x[3,6-5]+6\}$

$= \max\{x[3,6], x[3,1]+6\}$

$= \max\{6,6\}$

$= 6$

$x[4,7] = \max\{x[3,7], x[3,7-5]+6\}$

$= \max\{x[3,7], x[3,2]+6\}$

$= \max\{7,7\}$

$= 7$

$x[4,8] = \max\{x[3,8], x[3,8-5]+6\}$

$= \max\{x[3,8], x[3,3]+6\}$

$= \max\{7,8\}$

$= 8$

Example

Weight items = {2, 4, 3, 1, 5} $m=8, n=5$

Profit = {10, 5, 6, 8, 3}

		y_1	y_2	y_3	y_4	y_5	
		1	0	1	1	0	

Pi	Wi	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0
10	2	0	0	(10)	10	10	10	10	10	10
5	4	0	0	(10)	10	10	10	15	15	15
6	3	0	0	10	10	10	16	16	16	(16)
8	1	0	8	10	18	18	18	24	24	(24)
3	5	0	8	10	18	18	18	24	24	(24)

internet
tenretni

netervint
inIranet

Date: _____ intrnt

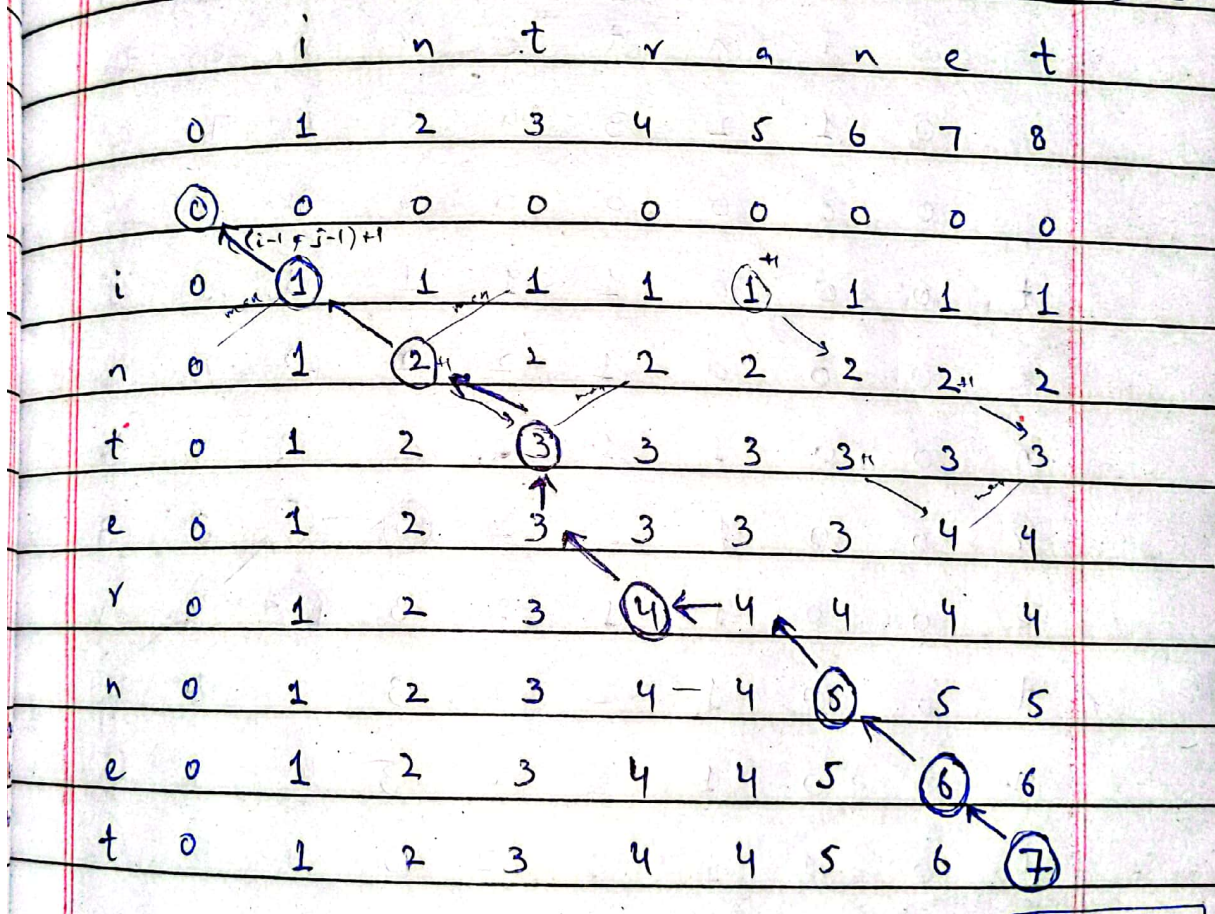
⇒ Dynamic Programming

- Identify and solve the smaller subproblems.
- Use optimal substructure (local optimal solution of the smaller subproblem) to construct/build the solution to the larger problem

⇒ Longest Common Subsequence

internet
intranet

intrnet (7)



intrnet

O(n×m)

Date:

if (A[i] = B[i])

MAT [i,j] = MAT [i-1, j-1] + 1

else

$$MAT[i,j] = \max \begin{cases} MAT[i-1, j] \\ MAT[i, j-1] \end{cases}$$

		i	n	t	e	r	n	e	t
	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0
t	0	0	0	1	1	1	1	1	1
e	0	0	0	1	2	2	2	2	2
n	0	0	1	1	2	2	3	3	3
r	0	0	1	1	2	(3)	(3)	3	3
e	0	0	1	1	2	3	(3)	(4)	4
t	0	0	1	2	2	3	3	4	(5)
n	0	0	1	2	2	3	4	4	(5)
i	0	1	1	2	2	3	4	4	(5)

tenet

terret

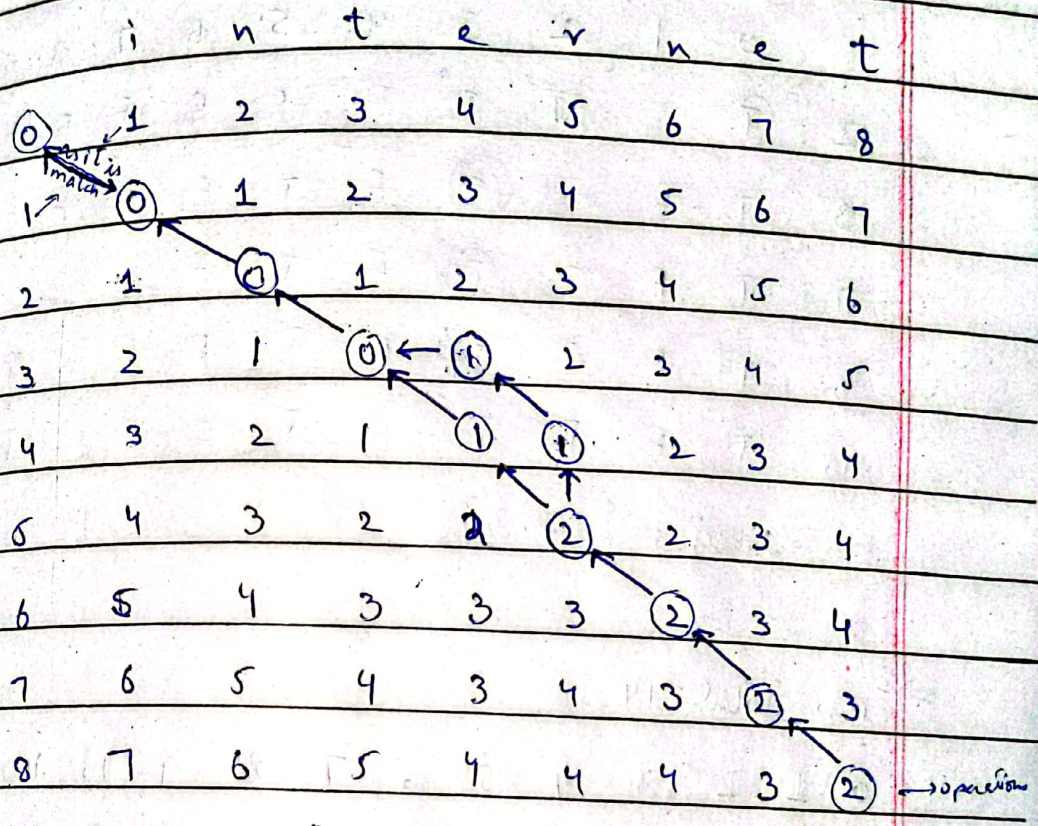
Date:

Minimum Edit Distance

→ if match → diagonal value → as it is
→ if no match → Read three values + 1 shortest from

edit
insert
delete

inte r net
int r a net



we are converting internet to intranet

↖ → edit
↑ → insert
← → delete

① edit → r will be converted to a
① r → a : edit
② e → r : edit

if match → no operations

a net
① insert a before net
② edit : ~~delete~~ e → r

Date:

⇒ Subset Sum Problem

⇒ Total = 11 $\{2, 3, 7, 8, 10\}$

	0	1	2	3	4	5	6	7	8	9	10	11
2	T	F	F	F	F	F	F	F	F	F	F	F
3	T	F	T	T	F	T	F	F	F	F	F	F
7	T	F	T	T	F	T	F				T	F
8	T	F	T	T	F	T	F			T	T	T
10	T	F	T	T	F	T	F			T	T	T

- Subset = {0, 3, 8}

⇒ Now, Total = 14

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	T	F	T	F	F	F	F	F	F	F	F	F	F	F
3	T	F	T	T	F			F			F	F	F	F
7	T		T	T	F	T	F	F	F	T	T	F	T	F
8	T	F	T	T	F	T	F	F	F	T	T	T	T	T
10	T	F	T	T	F	T	F	F	F	T	T	T	T	T

Date:

if (j < A[i])

Mat[i,j] = Mat[i-1, j]

else

Mat[i,j] = Mat[i-1, j] || Mat[i-1, j - A[i]]

⇒ Coin Changing Problem

T = 15

Coins {1, 2, 5, 10}

Total = 11

Coins = 1, 5, 6, 8

	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	3	4	5	6	7	8	9	10	11
5	(0)	1	2	3	4	(1)	2	3	4	5	2	3
6	0	1	2	3	4	(1)	1	2	3	4	2	2
8	0	1	2	3	4	1	1	2	1	2	2	(2)

7 6, 5 1

Subset = {5, 6} ⇒ minimum 2 coins will be used.

if (j ≥ A[i])

Mat[i,j] = min { Mat[i-1, j], Mat[i, j - A[i]] + 1 }

else

Mat[i,j] = Mat[i-1, j]

Date:

⇒ Cutting Rod to Maximize Profit

1	2	3	4
↓	↓	↓	↓
2	5	7	8

Length = 5

P	D	0	1	2	3	4	5
2	1	(0)	(2)	4	6	8	10
5	2	0	(2)	5	(7)	10	(12)
7	3	0	2	(5)	7	10	(12)
8	4	0	2	5	7	10	(12)

{2, 2, 1} {3, 2}

if (j ≥ i)

$Mat[i,j] = \max\{Mat[i-1,j], Mat[i,j-1] + A[i]$

else

$Mat[i,j] = Mat[i-1,j]$

if above doesn't match write '0'
if matches add 1 in diagonal

Date:

Dynamic Programming

⇒ Longest Common Substring

Execution

Intention (4)

		e	x	e	c	u	t	i	o	n
	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	1	0	0
n	0	0	0	0	0	0	0	0	0	1
t	0	0	0	0	0	0	1	0	0	0
e	0	1	0	1	0	0	0	0	0	0
n	0	0	0	0	0	(0)	0	0	0	1
t	0	0	0	0	0	0	(1)	0	0	0
i	0	0	0	0	0	0	0	(2)	0	0
o	0	0	0	0	0	0	0	0	(3)	0
n	0	0	0	0	0	0	0	0	0	(4)

end point

max value is the optimal solution

it can occur anywhere so you have to backtrack from that point

t i o n

if (str1[i] == str2[j])

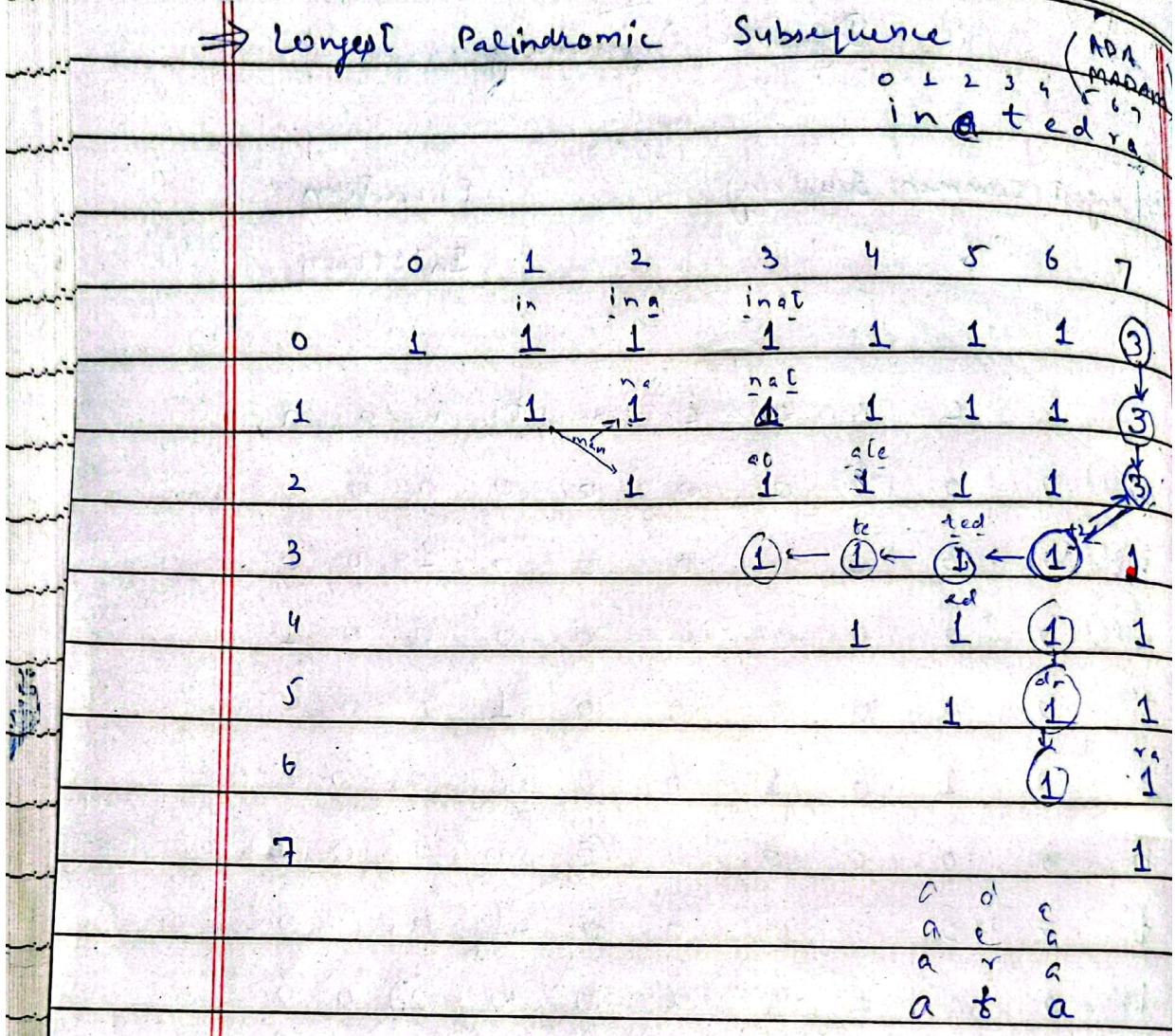
$T[i,j] = T[i-1, j-1] + 1$

else

$T[i,j] = 0$

Date:

⇒ Longest Palindromic Subsequence



if (str[i] = str[j])

$$T[i,j] = T[i+1, j-1] + 2$$

else.

$$T[i,j] = \max \begin{cases} T[i, j-1] \\ T[i+1, j] \end{cases}$$

Date:

Palindrome Partition

0	1	2	3	4
a	b	c	b	m

	0	1	2	3	4
0	0	1	2	1	2
1		0	1	0	1
2			0	1	2
3				0	1
4					0

minimum no. of splits (optimal solution)

0	1	2	3	4
a	b	c	b	m

if (str[i] = str[j])

$T[i,j] = T[i+1, j-1] + 2$

else

$T[i,j] =$

$T[0,3] = 1 + \min_{0 \le k < 3} \begin{cases} T[0][0] + T[1][3] \\ T[0][1] + T[2][3] \\ T[0][2] + T[3][3] \end{cases}$

$= 1 + \min \begin{cases} 0+0 & 0+1 & k=0 \\ 1+1 & 2 & k=1 \\ 2+0 & 2 & k=2 \end{cases}$

if ispalindrome (i,j)

$T[i,j] = 0$

else

$T[2,4] = 1 + \min_{2 \le k < 4} \begin{cases} T[2][2] + T[3][4] & k=2 \\ T[2][3] + T[4][4] & k=3 \end{cases}$

$\begin{cases} 0+1 = 1 & k=2 \\ 1+0 = 1 & k=3 \end{cases}$

$T[0][1] + T[2][2] \quad 1+0 \quad 1 \quad k=1$

$T[0][0] + T[1][2] \quad 0+1 \quad 1 \quad k=0$

$T[i,j] = 1 + \min \begin{cases} T[i][k] + T[k+1][j] \\ i \le k < j \end{cases}$

Date:

0	1	2	3	4	5	6	7
a	l	i	a	d	a	p	s	p

	0	1	2	3	4	5	6	7
0	0	1	0	1	2^0	3^0	4^0	3^0
1		0	1	2^1	1^1	2^1	3^1	2^1
2			0	1	0	1^4	2^4	1^4
3				0	1	2^3	3^3	2^3
4					0	1	2^4	1^4
5						0	1	0
6							0	1
7								0

$T[1,3] = T[1][1] + T[2][3] \rightarrow 0+1 \rightarrow 1 \quad k=1$

1,2,3 $\quad T[1][2] + T[3][3] \rightarrow 1+0 \rightarrow 1 \quad k=2$

$T[3,5] = \begin{cases} T[3][3] + T[4][5] \rightarrow 0+1 \rightarrow 1 & k=3 \\ T[3][4] + T[5][5] \rightarrow 1+0 \rightarrow 1 & k=4 \end{cases}$

3,4

$T[0,2] = \begin{cases} T[0][0] + T[1][2] \rightarrow 0+2 \rightarrow 2 & k=0 \\ T[0][1] + T[2][2] \rightarrow 1+1 \rightarrow 2 & 1 \\ T[0][2] + T[3][3] \rightarrow 0+0 \rightarrow 0 & 2 \end{cases}$

0,1,2

$T[3,6] = \begin{cases} T[3][3] + T[4][6] \rightarrow 0+2 \rightarrow 2 & k=3 \\ T[3][4] + T[5][6] \rightarrow 1+1 \rightarrow 2 & 4 \\ T[3][5] + T[6][6] \rightarrow 2+0 \rightarrow 2 & 5 \end{cases}$

3,4,5

Date: 10-01-2024

Word split / Break Problem

I am king → True

I am k → False

Given a string and English Vocabulary, can this string be splitted into multiple words in such a way that each word belong to the vocabulary.

if (str[i---j] is in vocabulary)

MAT[i,j] = T

else

MAT[i,j] = T

if there exists a k such that MAT[i,k] && MAT[k+1,j]

k=0 → MAT[0,1] = MAT[0,0] && MAT[1,1] → True

⇒ I am King

	0	1	2	3	4	5	6
0	T	T	T°	F	F	T°	(T°)
1		T	T	F	F	T²	(T²)
2			F	F	F	F	F
3				F	F	T (3-5)	(T (3-6))
4					T	T	F
5						F	F
6							F

0	I	T
1	a	T
2	m	F
3	k	F
4	l	T
5	n	F
6	g	F

1 2 3 4 5
a m k i n

Date:

i=1 MAT[1,5] = MAT[1,1] && MAT[2,5] = T && F = F

i=2 MAT[2,5] = MAT[1,2] && MAT[3,5] = T && T = T

0 1 2 3 4 5
I a m k i n

k=0 MAT[0,5] = MAT[0,0] && MAT[1,5] = T && T = T

⇒ 0 1 2 3
I a m k

	0	1	2	3
0	T	T°	T°	F
1		T	F	F
2			F	F
3				F

MAT[0,1] = MAT[0,0] && MAT[1,1] = T && T = T

MAT[3,4] = MAT

0 1 2 3 4 5
⇒ U v k i n g

	0	1	2	3	4	5
0	F	F	F	F	F	F
1		F	F	F	F	F
2			F	F	T (2-4)	T (2-5)
3				T	T (3-4)	F
4					F	F
5						F

Date: 12-01-2024

Dynamic Programming :

Wildcard Matching.

Decision Problem : Solution in Yes/No or T/F.

* : 0 or more characters

? : any one character

Pattern	Strings	
(i) a*b	ab, aab, abab, ..., axyb, ...,	= T
	a string has to begin with a and end with b in this case	
	b, a, ac, abc, ...	= F
(ii) a?b	aab, abb, axb, ...	= T
	u can only put one character in between.	
	ab, ac, b, acd, a, acdb, ...	= F
(iii) x?y*z	xayz, xxyyzz, xaybcz, ...	= T accepting state
	xaz, ayz, xyz, xxx, xbc, ...	= F non-accepting state

```
if (str[i] == pattern[j]) || pattern[j] = '?'
    Mat[i,j] = Mat[i-1, j-1]
else if (pattern[j] == '*')
    Mat[i,j] = Mat[i-1, j] || mat[i, j-1]
else
    Mat[i,j] = 'F'
```

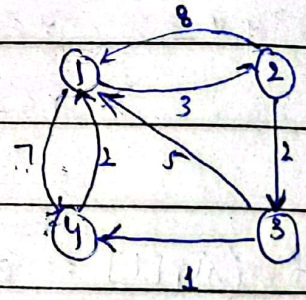
Date:

pattern = x ? y * z

string = x a y l m z

		ε	x	?	y	*	z
		0	1	2	3	4	5
0	ε	T	F	F	F	F	F
1	x	F	T	F	F	F	F
2	a	F	F	T	F	F	F
3	y	F	F	F	T	T	F
4	l	F	F	F	F	T	F
5	m	F	F	F	F	T	F
6	z	F	F	F	F	T	T

⇒ Floyd Warshall's Algorithm



→ All pair shortest path:

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Date:

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & \boxed{2} & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$

∞
$2+\infty$

$A^1[2,3] = \min \begin{cases} A^0[2,3] & = 2 \\ A^0[2,1] + A^0[1,3] & = 8+\infty \end{cases}$

intermediate node

$A^1[2,4] = \min \begin{cases} A^0[2,4] & = \infty \\ A^0[2,1] + A^0[1,4] & = 8+7 = 15 \end{cases}$

$$\Rightarrow A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

1
$15+8 = 23$

$A^2[1,3] = \min \begin{cases} A^1[1,3] = & = \infty \\ A^1[1,2] + A^1[2,3] = 3+2 & = 5 \end{cases}$

$A^2[1,4] = \min \begin{cases} A^1[1,4] = & = 7 \\ A^1[1,2] + A^1[2,4] = 3+15 = 18 \end{cases}$

Date:

$\Rightarrow A^3 =$

	1	2	3	4
1	0	3	5	6
2	7	0	2	3
3	5	8	0	1
4	2	5	7	0

$A^3[1,2] = \min \begin{cases} A^2[1,2] & = 3 \\ A^2[1,3] + A^2[3,2] & = 5+8 = 13 \end{cases}$

This is the solution

$\Rightarrow A^4 =$

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

$A^4[$

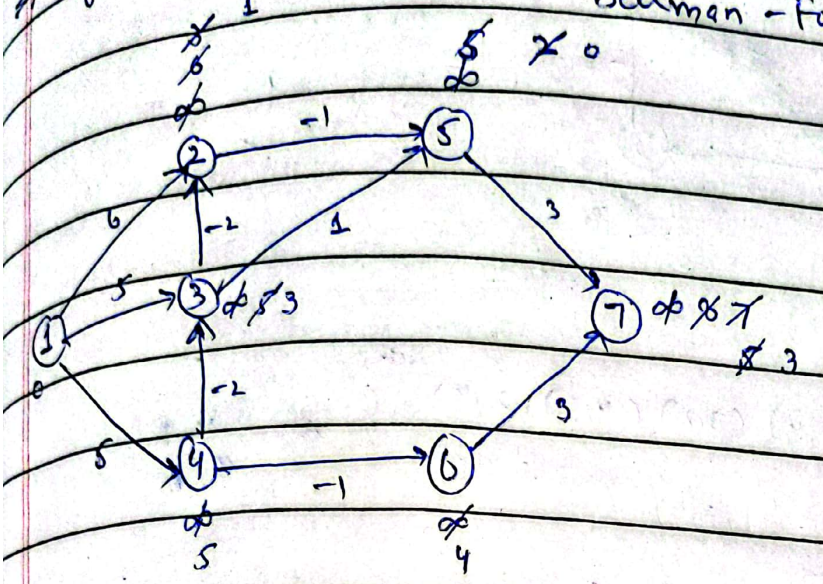
$A^k[i,j] = \min \begin{cases} A^{k-1}[i,j] \\ A^{k-1}[i,k] + A^{k-1}[k,j] \end{cases}$

$\Rightarrow O(n^3)$: To populate one matrix we require $n \times n = n^2$ and there are 'n' no. of matrices, so, $n^2 \times n = n^3$

Date:

Single Source Shortest Path

Bellman - Ford



Updation:

```
if (d[u] + c(u,v) < d[v])
    d[v] = d[u] + c[u,v]
```

(3,5)

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

1st iteration ✓

2nd iteration ✓

3rd iteration ✓

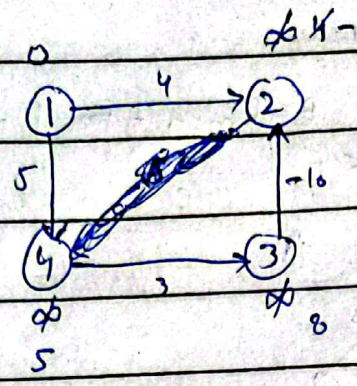
4th iteration ✓ (No change in 3rd iteration, so, we exit)

1-0	5-0	$O(\frac{\|V\|}{n}\frac{\|E\|}{n})$	In case of complete graph (all possible edges $n(n-1)/2$)
2-1	6-4	In this case incomplete graph $O(n^2)$	$O\left(n\left[\frac{n(n-1)}{2}\right]\right)$
3-3	7-3		
4-5			$O(n^3)$

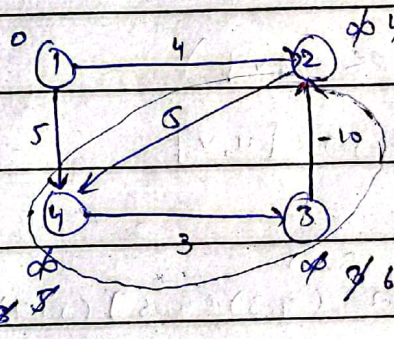
one less vertices iterations.

Date:

4 vertices
So, 3 iterations.



(1,2) (1,4) (3,2) (4,3) (2,4)



Negative weight cy

5+3-10 = -2

(1,2) (1,4) (3,2) (4,3) (2,4)

There is updation in iteration 04 due to Negative weight cycle and Bell-man Ford Algorithm fails.

Date:

Branch and Bound

- a problem solving strategy
- uses state space tree
- optimization problems – only for minimization
- FIFO branch and bound
- LIFO branch and bound
- least cost branch and bound

Jobs Sequencing

(Penalty minimize krni hai)

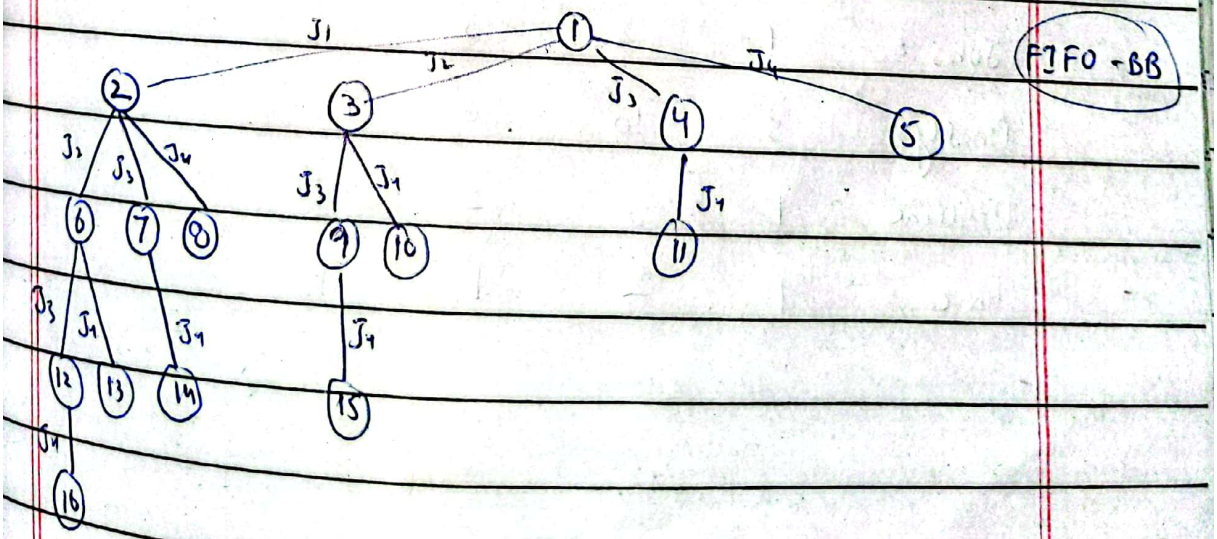
Jobs = $\{J_1, J_2, J_3, J_4\}$

$P_{penalty} = \{10, 5, 8, 3\}$

$d = \{1, 2, 1, 2\}$

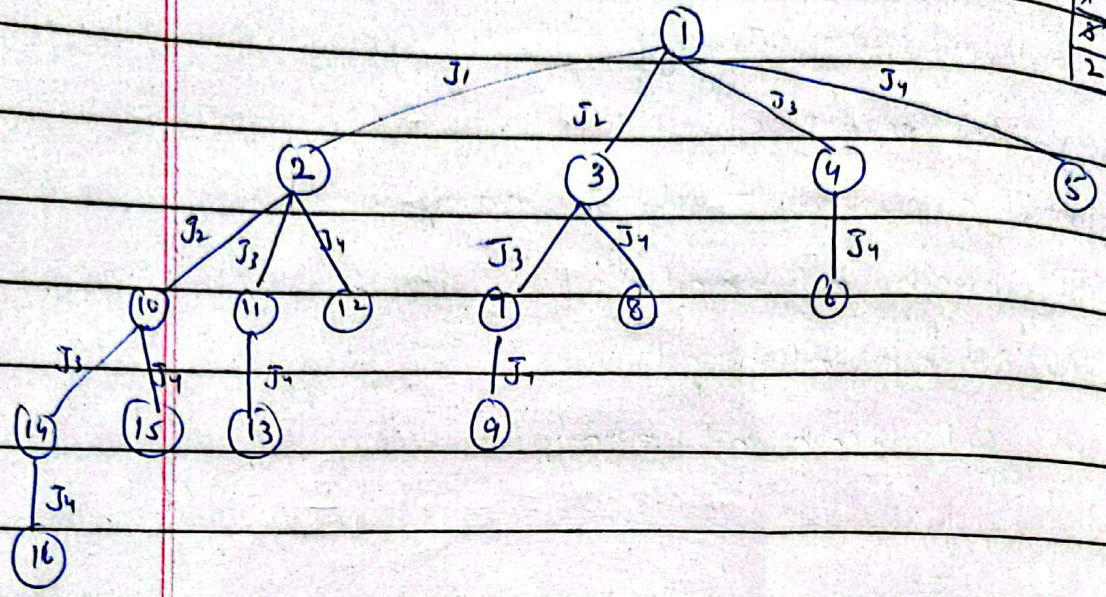
variable site $\{J_1, J_4\}$

Fixed site $\{1, 0, 0, 1\}$



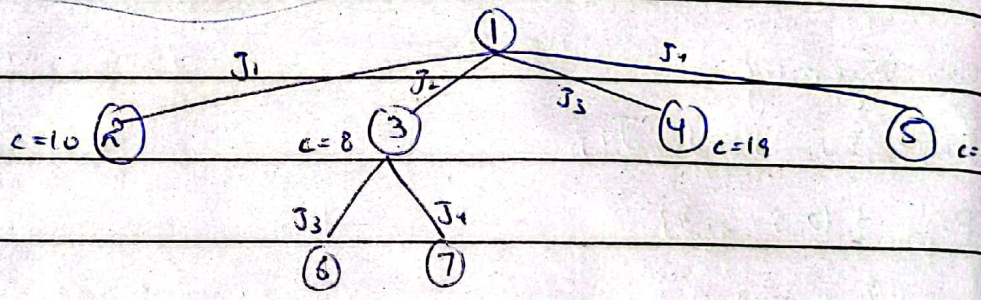
Date:

⇒ LIFO – B&B



⇒ Least-cost – B&B

much faster



Branch and Bound

	Jobs	1	2	3	4
Penalty minimize ←	Penalty	5	10	6	3
can wait ←	Deadline	1	3	2	1
time required to finish a job ←	Time	1	2	1	1

Now compare the value of $\hat{c}$ with Upper. If it exceeds, u will branch out and there will be no need to further explore it. otherwise we'll update the upper.

c representing how many jobs you have left.

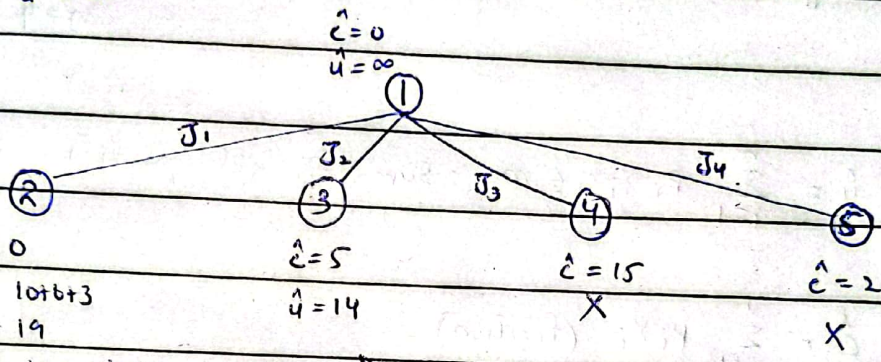
Date: Upper = ∞ 14 14 9 8

$\hat{c} = \sum_{i \in S_k} P_i$ Sum of Penalty till the last job

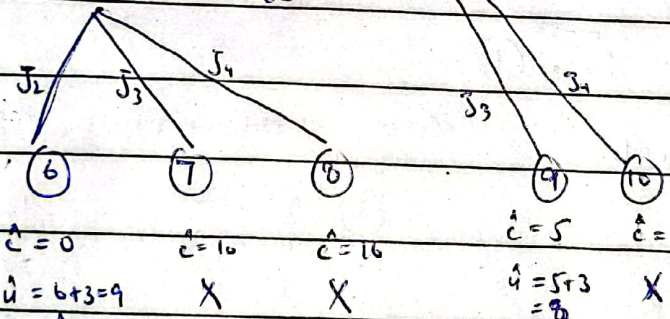
$\hat{u} = \sum_{i \notin S} P_i$ Sum of all penalties except those involved in the solution.

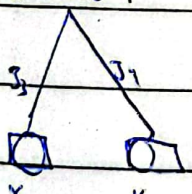
$\hat{c} = 0$

$\hat{u} = \infty$



u have to explore the minimum cost node





J_3 and J_4 is not feasible solution becoz the slots are already booked

Last node is infeasible solution.

Last acting node was (9) and if we follow that path we have J_2, J_3 which is the optimal solution and we have saved 8 penalties

$\{J_2, J_3\}$

Date: Upper of -32 -38

⇒ 0/1 knapsack problem maximization into minimization

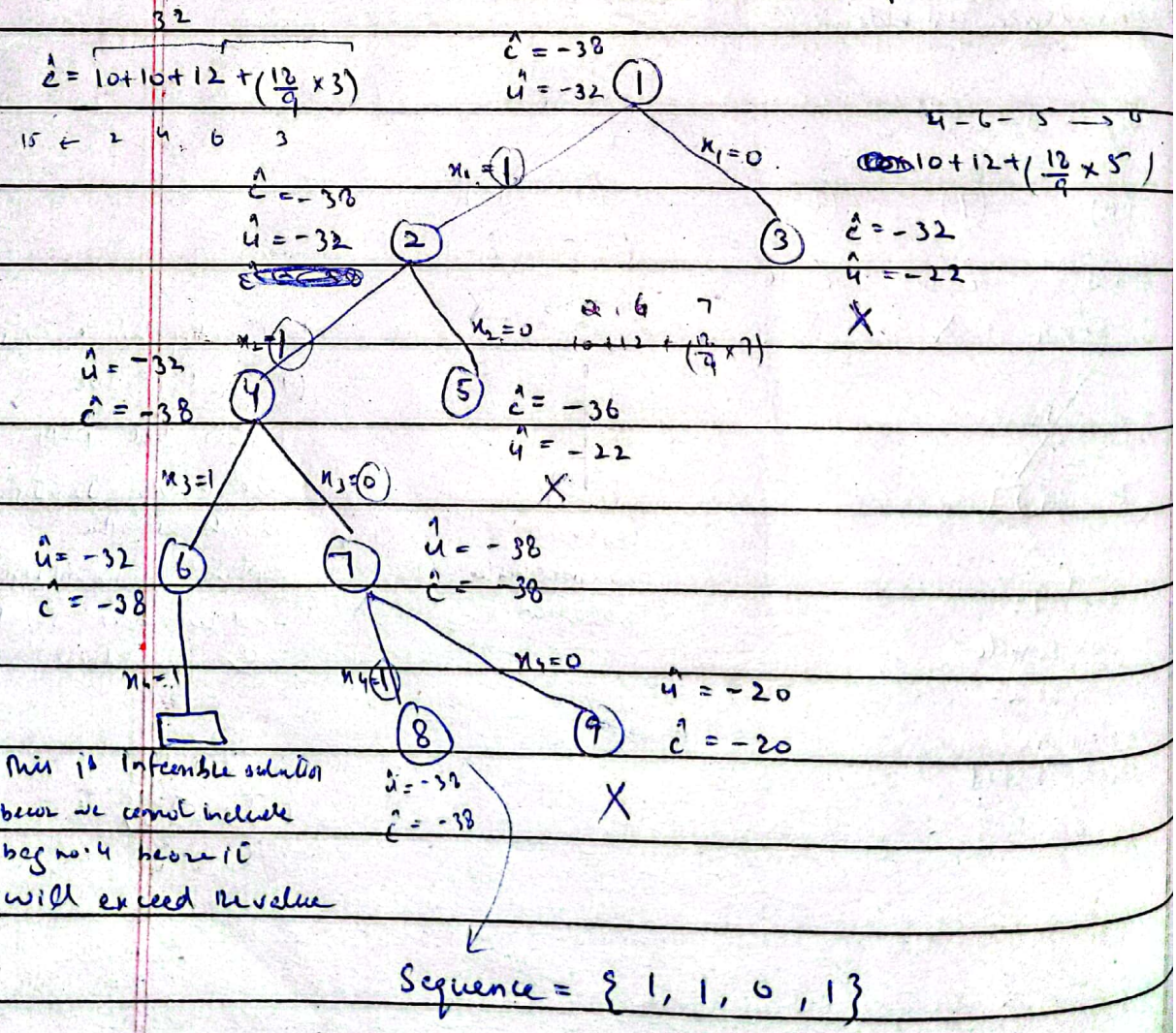
	1	2	3	4
Profit	10	10	12	18
Weight	2	4	06	9

m = 15
n = 4

$\hat{u} = \sum_{i=1}^{n} P_i X_i \leq m$: Sum of all profits

$\hat{c} = \sum_{i=1}^{u} P_i X_i$ (fraction)

minus '-' because minimization problem



Sequence = {1, 1, 0, 1}