

* DAA *Index

Page No :-	Topic
1 - 2	Introduction to DAA & what is Algorithm
2 - 3	Asymptotic Notation
4 - 6	Various property of Asymptotic Notation time complexity of all searching Algo.
6 - 9	Recurrence Relation / Substitution method with example.
10 - 12	Master Method with example
13 -	Divide and Conquer
14 -	Quick sort Sort. (DAC)
15	Performance of Quick Sort
16 - 18	Bubble Sort / Insertion Sort / Selection Sort
19 - 20	Scodo code / Best case, worst case
21	Binary tree and its type
22 - 26	Heap tree, Heapfy method, Deletion Heap tree, Heap Sort
27 - 32	Greedy Technique, Application, Knapsack, Huffman Coding, Job Sequencing, optimal merge pattern, Kruskal Algo.
33 - 38	Prims Algo, Dijkstras Algo and Analysis Bellman Ford Algo, Pseudo code.

- 39- 41 Dynamic programming
0/1 knapsack failed using greedy approach
0/1 knapsack Recursive eqn.
- 42- 44 Travelling Salesman Problem, sum of subset
Multi stage graph
- 45- 46 All pair shortest path (Floyd, warshall, Algo)
Floyd warshall working with Ex.
- 47- 49 Hashing , collision in Hashing , chaining
Linear probing
- 49- 50 Quadratic probing . Double Hashing
- 51- Recurrence Relation $[T(n) = 3T(n/4) + cn^2]$

D
DAA :-

Introduction to Algorithm and its Syllabus :-

1. Asymptotic Notation
2. Time and space complexity
3. Divide and conquer , All sorting Algos , Heap Tree
4. Greedy methods (Job sequencing , Knapsack , Optimal merge pattern , Huffman encoding , Dijkstra , MST ~~X~~ Prims , Kruskal)
5. Graph Traversal (DFS , BFS , connected components)
6. Dynamic Programming (All pair shortest Path , Multistage graph , Optimal Binary search tree , TSP , 0/1 Knapsack , LCS , matrix chain , multi , SOS)
7. Hashing
8. PNP, NPH, NPC] Grade X

What is Algorithm :-

- Finite set of step to solve a problem is called Algorithm.
- Analysis is process of comparing to algo w.r.t. time , space. etc.

S₁ : Read A

S₂ : Read B

S₃ : Sum = A+B

S₄ : Print (Sum)

Any Analysis

Primary Posterior

Exact

APP.

Relevant

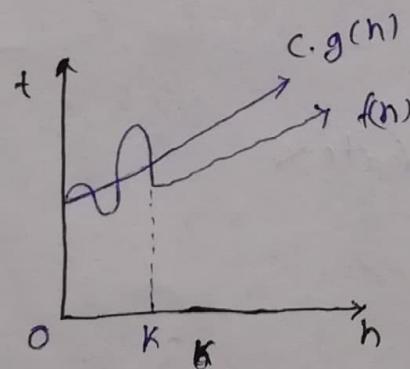
Independent

uniform . . .

main()
int a,b,sum
scanf(a,b)
sum = a+b;
Pf(sum);

Asymptotic Notation :-

(ii) Big- Θ (Θ)



→ worst case

→ upper bound (At most)

$$f(n) = \Theta(g(n))$$

$$f(n) \leq C.g(n)$$

$$C > 0$$

$$n \geq K$$

$$K \geq 0$$

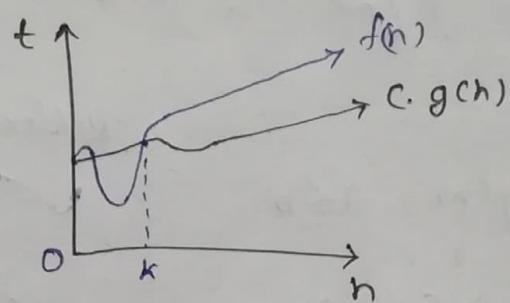
$$f(n) = 2n^2 + n \quad f(n) = \Theta(n^2)$$

$$2n^2 + n \leq 3 \cdot n^2$$

$$n \leq n^2$$

$$\boxed{n \geq 1}$$

(2) Big-Omega (Ω)



→ Best case

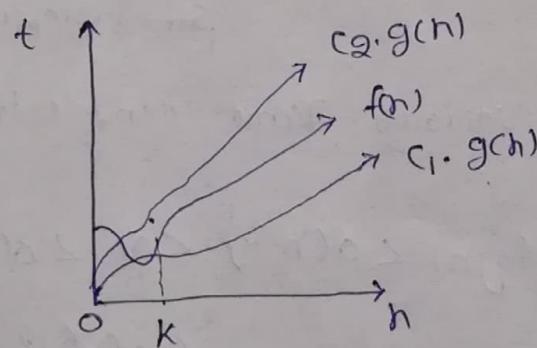
→ Lower Bound (At least)

$$f(n) = \Omega(g(n))$$

$$f(n) \geq c \cdot g(n)$$

$$2n^2 + n \geq c \cdot n^2$$

(3) Theta (Θ) :-



→ Average case

→ Exact time

$$\text{if } c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

$$n^2 \leq 2n^2 + n \leq 3n^2$$

Various Property of Asymptotic Notation :-

$$f(n) = o(g(n)) \Rightarrow g(n) > f(n)$$

Transitive

Big O, $f(n) \leq C.g(n)$, $a \leq b$

Reflexive

Symmetric

✓

Big Omega (Ω), $f(n) \geq C.g(n)$ $a \geq b$

✓

✗

✓

Theta (Θ), $C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$

✓

✓

✓

Small (o), $f(n) < C.g(n)$ $a = b$

✗

✗

✓

Small (ω), $f(n)^2 > C.g(n)$

$a > b$

✗

✗

✓

Same value but

Comparison of various Time complexities :-

$$\begin{aligned}
 o(1) &< o(\log \log n) < o(\log n) < o(n^{1/2}) < o(n) < o(n \log n) < o(n^2) < o(n^3) \rightarrow \\
 &\rightarrow < o(n^k) < o(2^n) < o(n^n) < o(2^{2^n})
 \end{aligned}$$

* Time complexities of all Searching and sorting Algorithm :-

- Binary Search - $\log_2 n$
- Sequential Search - $O(n)$
- *Quick Sort - $O(n \log n)$. Worst = $O(n^2)$
- Merge Sort - $O(n \log n)$
- *Insertion Sort - $O(n^2)$ Avg. $O(n)$ Worst
- Bubble Sort $\rightarrow O(n^2)$
- Heap Sort $\rightarrow O(n \log n)$
- Selection Sort $\rightarrow O(n^2)$

- Height of CBT = $O(\log n)$
- Insertion in Heap ($\log n$)
- Construct Heap ($n \log n$)
- Delete from Heap ($\log n$)
- Huffman ($n \log n$)
- Prim's matrix $\rightarrow O(n^2)$
- Kruskal $O(E \log E)$
- DFS, BFS $O(V+E)$
- All pair shortest $\rightarrow O(n^3)$
- Dijkstra $O(V^2)$

Q1:- $f_1(n) = n^2 \log_{10} n$ $f_2(n) = n (\log_{10} n)^{10}$

(a) $f(n) = O f_2(n)$

(b) $f_1(n) = n f_2(n)$

Let $n=16$

$$f_1 = 16^2 \times 4 \quad f_2(16) = 16 \log_2^{16}$$

$$= 16 \times 4^{10}$$

$$= 10^9$$

$$\text{Let } n = 10^9$$

$$f_1(10^9) = (10^9)^2 \log_{10}^{10^9}$$

$$= 10^9 \times 10^9 \times 9$$

$$= 10^9 \times 9^9$$

$$= 10^9 (\log_{10} 10^9)^{10}$$

$$= 10^9 \times (9)^{10}$$

$$= 9^{\frac{1}{10}} \times 10^9$$

⑥

$$\Rightarrow Q! - f_1(n) = 2^n, f_2(n) = n^{3/2}, f_3(n) = n \log_2 n, f_4(n) = n^{\log_2 2^n}$$

Arranging increasing order

a) $f_2 \ f_3 \ f_4 \ f_1$

b) $f_2 \ f_1 \ f_3 \ f_4$

b) $f_1 \ f_2 \ f_3 \ f_4$

~~d)~~ $f_3 \ f_2 \ f_4 \ f_1$

Let $n = 256$

$$2^{256}$$

$$256^{3/2}$$

$$2^{(8+3)/2}$$

$$2^{12}$$

$$(256) \log_2(256)$$

$$256 \log_2 2^8$$

$$2^8 \times 8$$

$$2^{11}$$

$$(256)^{\log_2 256}$$

$$(256)^8$$

$$2^{11} < 2^{12} < 256^8 \times 2^{256}$$

$$f_3 < f_2 < f_4 < f_1$$

Recurrence Relation :-

BS(a, i, j, n)

$$\text{mid} = (i+j)/2$$

$$\text{if } a[\text{mid}] == \text{target}$$

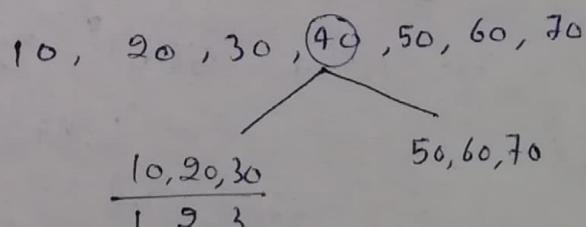
return mid;

else
if (a[mid] > target)

BS(a, i, mid - 1, n)

else

\rightarrow BS(a, mid + 1, j, n)



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

Substitution Method :-

$$T(n) = \begin{cases} T(n/2) + C & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

$$T(n) = T(n/2) + C \rightarrow ①$$

$$T(n) = T(n/4) + C + C$$

$$T(n/2) = T(n/4) + C \rightarrow ②$$

$$T_n = T(n/4) + 2C$$

$$T(n/4) = T(n/8) + C \rightarrow ③$$

$$\begin{aligned} &= T(n/8) + 2C \\ &= T(n/8) + 3C \end{aligned}$$

$$n = 2^k$$

$$T(n/2^k) + kC$$

$$\log n = k \log 2$$

K time

$$\frac{1+kC}{1+\log n \cdot C}$$

$$T(n/n) + kC$$

$$T(1) + kC$$

$$\boxed{O(\log n)}$$

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ n * T(n-1) & \text{if } n > 1 \end{cases}$$

$$T(n) = n * T(n-1) \rightarrow ①$$

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

$$= (n-1) * T(n-2) \rightarrow ②$$

$$T(n-2) = (n-2) * T(n-3) \rightarrow ③$$

$$T(n) = n * (n-1) * T(n-2)$$

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

1
}
(n-1) Step 3

$$n * (n-1) * (n-2) * (n-3) \dots T(n-(n-1))$$

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

$$T(1)$$

$$n * (n-1) * (n-2) * (n-3) \dots * 1$$

$$n * n \left(1 - \frac{1}{n}\right) * n \left(1 - \frac{2}{n}\right) * \dots * n \left(\frac{3}{n}\right) * n \left(\frac{2}{n}\right) * n \left(\frac{1}{n}\right)$$

$$\underline{O(n^n)} \quad \text{Ans}$$

Q: $T(n) = \begin{cases} 1 & \text{if } n=1 \\ 2T\left(\frac{n}{2}\right) + h & \text{otherwise} \end{cases}$

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

$$T(n/4) = 2T(n/8) + h/4 \rightarrow ③$$

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

$$= 2^2 T(n/4) + h + h/2$$

$$= 2^2 T(n/4) + 2h$$

Put equation No 3-

$$T(n) = 2^2 \left[2T\left(\frac{n}{8}\right) + \frac{2h}{4} \right] + 2h$$

$$= 2^3 T\left(\frac{n}{8}\right) + 3h$$

$$2^4 T\left(\frac{n}{2^4}\right) + 4h, \quad 2^5 T\left(\frac{n}{2^5}\right) + 5h$$

(9)

Run K time

$$2^K T\left(\frac{n}{2^K}\right) + kn \rightarrow n T\left(\frac{2^K}{2^K}\right) + n \log n$$

Let $n = 2^K$
 & taking log on both sides
 $\log n = \log 2^K$
 $\log n = K \log_2 2$
 $\boxed{\log n = K}$

$$\boxed{O(n \log n)} \text{ Ans}$$

$$Q!- T(n) = \begin{cases} 1 & , \text{if } n=1 \\ T(n-1) + \log n, & \text{if } n>1 \end{cases}$$

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

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

$$\begin{aligned} T(n) &= T(n-2) + \log(n-1) + \log n \\ &= T(n-3) + \log(n-2) + \log(n-1) + \log n \end{aligned}$$

K time

$$= T(n-K) + \log(n-(K-1)) + \log(n-(K-2)) + \dots + \log n$$

$$n-K=1$$

$$\begin{aligned} n=K &= T(1) + \log(K-K+1) + \log(K-K+2) + \dots + \log n \\ &= T(1) + \log(1) + \log(2) + \log(3) + \dots + \log n \\ &= 1 + \log(1 \cdot 2 \cdot 3 \cdot 4 \dots n) \\ &= 1 + \log n \quad \boxed{1+n \log n} \\ &= 1 + \log n \quad \boxed{O(n \log n)} \text{ Ans} \end{aligned}$$

+

Master Method :-

$$\rightarrow T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$$\boxed{a \geq 1}, \boxed{b > 1}$$

\rightarrow Solution is :-

$$T(n) = n^{\log_b a} [U(n)]$$

$\rightarrow U(n)$ depends on $f(n)$

$$\rightarrow \left\{ f(n) = \frac{f(n)}{n^{\log_b a}} \right\} = \frac{n^2}{n^{\log_2 \delta}} = \frac{n^2}{n^3} = \frac{1}{n} = n^{-1}$$

\rightarrow Relation between $f(n)$ and $U(n)$ is \rightarrow

if $f(n)$	$U(n)$
$n^{\alpha}, \alpha > 0$	$O(n^{\alpha})$
$n^{\alpha}, \alpha < 0$	$O(1)$
$\rightarrow (\log n)^i, i \geq 0$	$\frac{(\log_2 n)^{i+1}}{i+1}$

Q :- $T(n) = \delta T(n/2) + n^2$

$$T(n) = \delta + (n/2) + n^2$$

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

$$T(n) = n^{\log_b a} \cdot U(n)$$

$$= n^{\log_2 8} \cdot U(n)$$

$$= n^3 \frac{U(n)}{1}$$

$U(n)$ depends on $T(n)$

$$h(n) = \frac{f(n)}{n^{\log_b a}} = \frac{n^2}{n^{\log_2 8}} = \frac{n^2}{n^3} = \frac{1}{n} = n^{-1}$$

$$\rightarrow n^2, n < 0 \quad | \quad O(1)$$

$$= \underline{n^3 O(1)} = n^3 \cancel{O(1)}$$

Q :- $T(n) = T(n/2) + C$

$$a=1 \quad b=2 \quad f(n)=C$$

$$T(n) = n^{\log_b a} \cdot U(n)$$

$$= n^{\log_2 1} \cdot U(n) \Rightarrow n^0 \cdot U(n) \Rightarrow U(n)$$

$$h(n) = \frac{f(n)}{n^{\log_b a}} = (\log_2 n)^0 \cdot C$$

$$\frac{(\log_2 n)^{0+1}}{0+1} = (\log_2 n) \cdot C$$

$$= \log_2 n \cdot C$$

$$= O(\log_2 n) \text{ Ans}$$

$$T(n) = \begin{cases} T(\sqrt{n}) + \log n & \text{if } n \geq 2 \\ O(1) & \text{else} \end{cases}$$

$$T(n) = T(\sqrt{n}) + \log n$$

$$n = 2^m$$

$$T(2^m) = T(2^{m/2}) + m$$

$$T(2^m) = S(m)$$

$$S(m) = S(m/2) + m$$

$$a < b^k$$

$$1 < 2^1$$

$$T(n) = aT(n/b) + n^k \log_b n$$

$$\begin{cases} (n)^{1/2} \\ (2^m)^{1/2} \\ 2^{m/2} \end{cases}$$

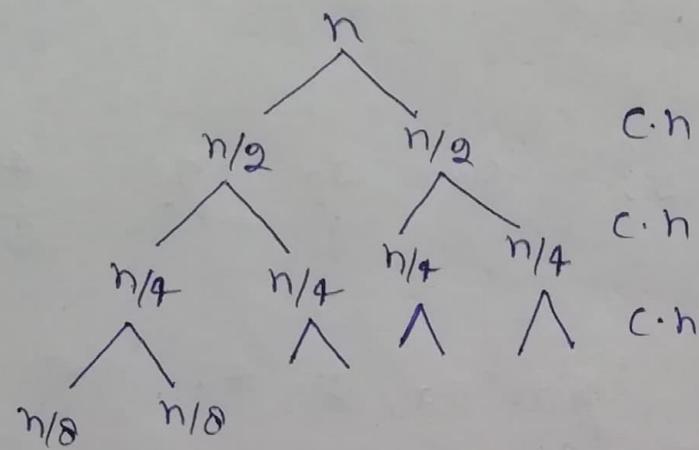
$$\begin{aligned} a &= 1 \\ b &= 2 \\ k &= 1 \\ p &= 0 \end{aligned}$$

$$= n^k \log_b n$$

$$= m^1 \quad (m) \Rightarrow \log n$$

Q:-

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



$$\begin{aligned} & c \cdot n \log n \\ & \downarrow \\ & O(n \log n) \end{aligned}$$

Divide and Conquer :-

DAC(P)

```
{ if (small( $P$ ))  
  { S( $P$ );  
  }
```

else

```
{ divide  $P$  into  $P_1 P_2 P_3 \dots P_K$ 
```

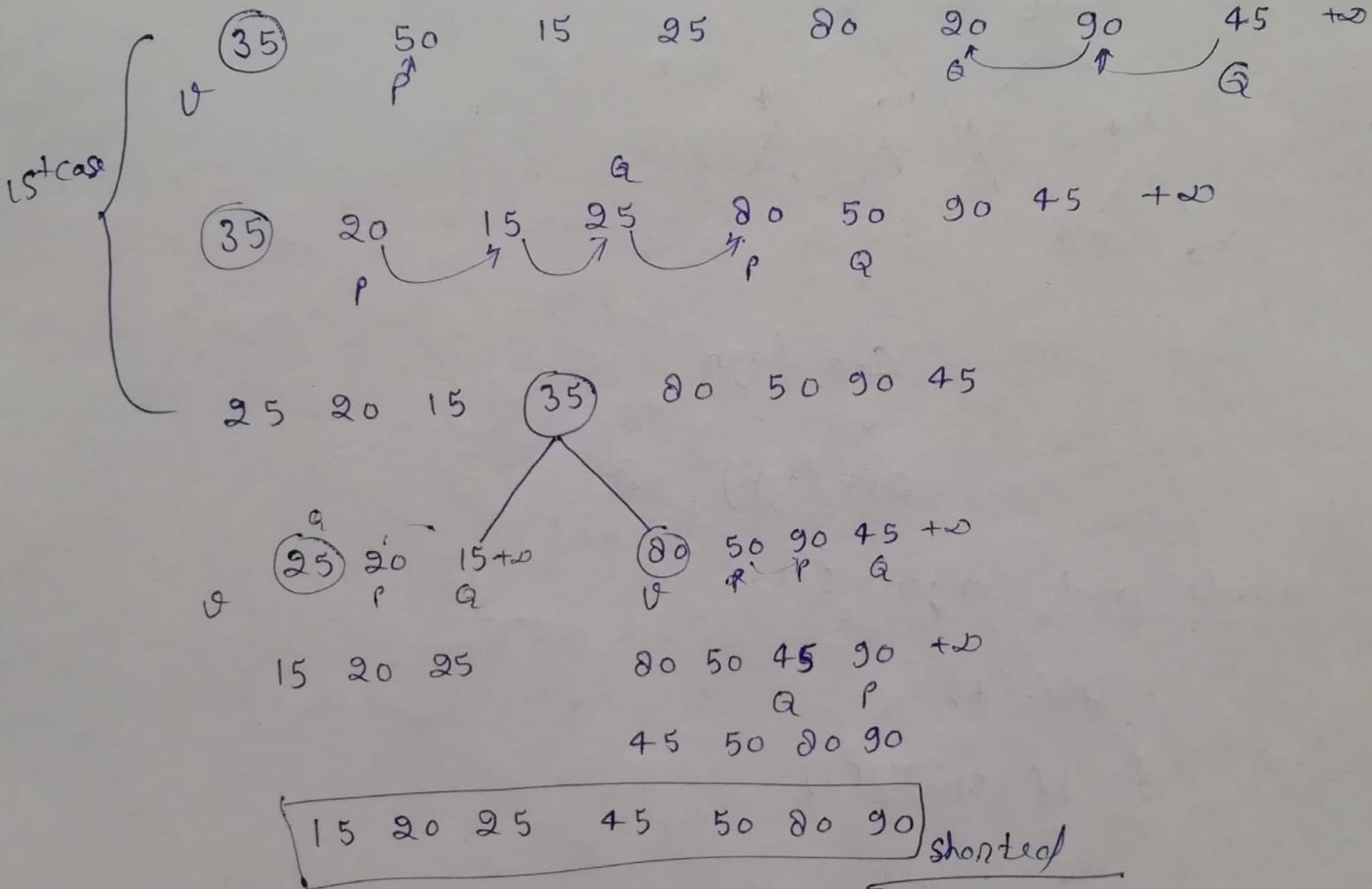
Apply $DAC(P_1), DAC(P_2) \dots DAC(P_K)$

Combine $DAC(P_1), DAC(P_2) \dots DAC(P_K)$

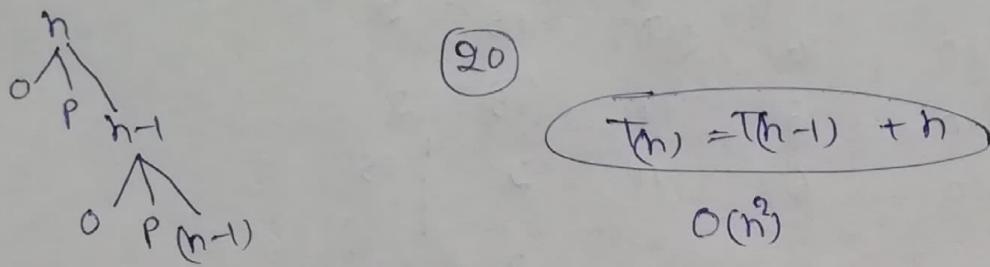
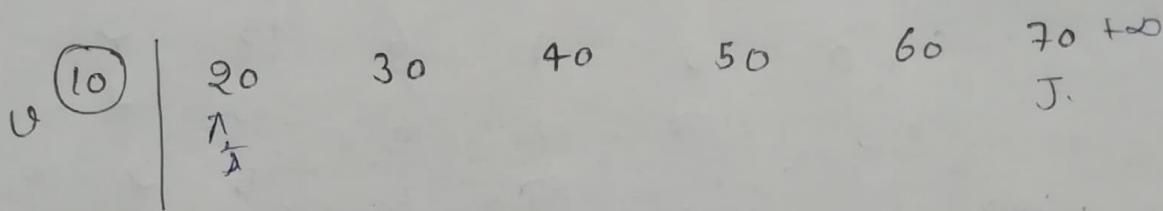
```
}
```

- (1) Binary Search
 - (2) Find Maximum and Minimum
 - (3) Quick Sort
 - (4) Merge Sort
 - (5) Strassen's Matrix multiplication.

Quick Sort :- (DAC)



Performance of Quick Sort:-



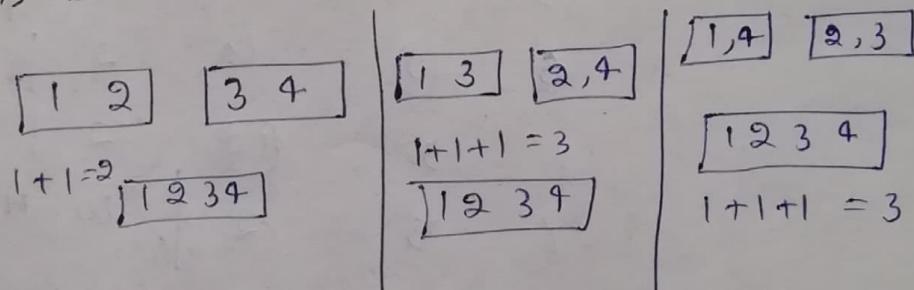
* * Q!- The average no. of comparisons performed by merge sort algorithm, in merging 2 sorted list of length 2 is 2.67

$$n \log n$$

$$4 \cdot \log_2 4$$

$$4 \times 2$$

$$\frac{8}{4} = 2 \text{ As}$$



$$\frac{8}{3} = 2.67$$

Bubble Sort :-

	10	9	11	6	15	2
1st pass	9	10	11	6	15	2
	9	10	11	6	15	2
	9	10	6	11	15	2
	9	10	6	11	15	2
	9	10	6	11	2	15
2nd pass	9	10	6	11	2	15
	9	6	10	11	2	15
	9	6	10	11	2	15
	9	6	10	2	11	15
	9	6	10	2	11	15
	9	6	10	2	11	15
	6	9	10	2	-	-
	6	9	2	-	10	11 15

Time complexity :-

$$\begin{aligned}
 & n \times (n-1) \\
 & = \frac{n^2 - n}{2} \\
 & = O(n^2)
 \end{aligned}$$

Insertion - Sort (A) :-

for $j = 2$ to $A.length$

Key = $A[j]$

// Insertion $A[j]$ into the sorted sequence $A[1..j-1]$.

$i = j-1$

while $i > 0$ and $A[i] > \text{key}$

$A[i+1] = A[i]$

$i = i-1$

$A[i+1] = \text{key}$

20	40				
40	20	60	10	50	30



40 20 40

10 20 40 50 60

10 20 40 50 60

10 20 30 40 50 60 A

Best case (Ascending order) :-

10 20 30 40 50 60

$(n-1)$

$\underline{\text{O}(n)}$ ↗

Comparing Swap

0 0
| |
|

Worst Case (Descending order) :-

60 50 40 30 20 10

Comp

$$\frac{n(n-1)}{2}$$

$$O(n^2)$$

Swept

$$\frac{n^2-n}{2}$$

Comp

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

Swept

6

7

8

9

10

n-1

Average Case

Worst case performance :- $O(n^2)$ comparisons and swap.

Best-case performance :- $O(n)$ comparisons, $O(1)$ swap.

Average performance :- $O(n^2)$ comparisons and swap.

Selection Sort :-

for $i \leftarrow 1$ to $n-1$ do

min $j \leftarrow i;$

min $x \leftarrow A[i]$

for $j \leftarrow i+1$ to n do

If $A[j] < \text{min } x$ then

min $j \leftarrow j$

min $x \leftarrow A[j]$

$A[i] \leftarrow \text{min } x$

40 20 60 10 50 30

Scodo Code:-

```

for i = 1 to n-1
    /* set current element as minimum */
    min = i
    /* check the element to be minimum */

    for j = i+1 to n
        if list[j] < list[min] then
            min = j;
        end if
    end for

    /* swap the minimum element with the current element */
    if index[min] != i then
        swap list[min] and list[i]
    end if
end for

for i = 1 to n-1
    /* set current element as minimum */
    min = i
    /* check the element to be minimum */

```

for $j = i+1$ to n

if $\text{list}[j] < \text{list}[\text{min}]$ then

$\text{min} = j;$

end if

end for

/* Swap the minimum element with the current element */

if $\text{index}(\text{min}) = i$ then

Swap $\text{list}[\text{min}]$ and $\text{list}[i]$

end if

end for

Best case (Assending Order):-

10 20 30 40 50 60

$O(n^2)$

$n-1$
 $n-2$
 $n-3$
⋮
0

$$\frac{n(n-1)}{2} = O(n^2)$$

Worst case (descending order):-

60 50 40 30 20 10

min_0

56

46

36

26

16

$(n-1)$

10 | 50 40 30 20 | 0

10 20 | 50 40 30

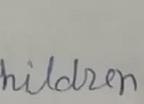
10 20 30 | 50 40

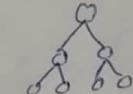
10 20 30 40 | 50

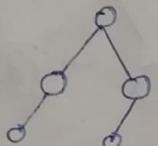
10 20 30 40 50

Binary Tree and its Variations (Full BT, ACBT, Complete BT, BST)

→ Binary Tree :- At most 2 children leaf nodes

→ Full BT :- Either 0 or 2 children → 

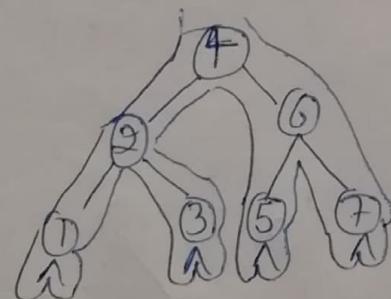
→ Complete BT :- No Holes → 

→ ACBT :- 

→ BST :-

Create Binary search tree (BST) with following keys :-

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



Inorder 1 2 3 4 5 6 7

* Introduction of "Heap Tree" (Insertion, Deletion, Heap sort,

1- Structural property

2- ordering property

Heapify . Array representation, Priority

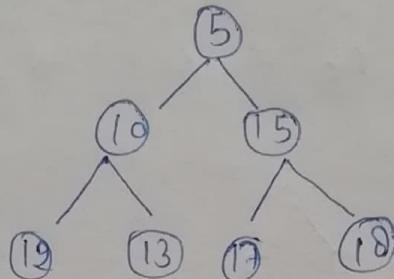
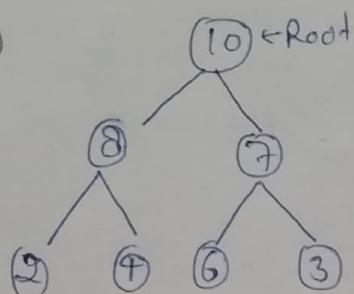
max Heap

(Parents > child)

min Heap

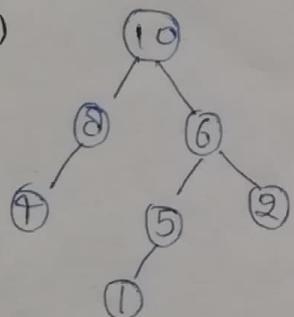
(Parents < child)

(ACBT)

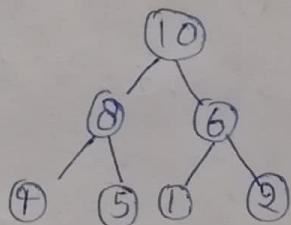


Q1- which of the following is max. Heap?

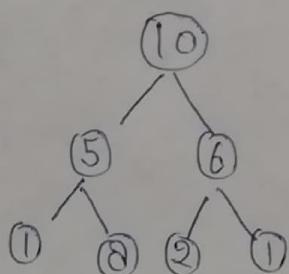
(A)



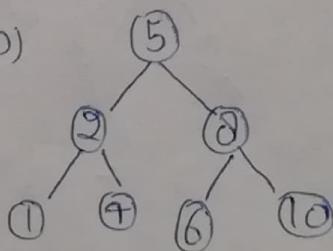
(B)



(C)



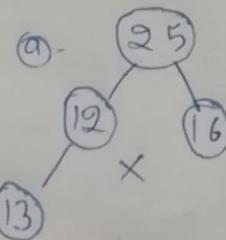
(D)



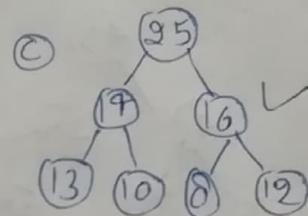
Consider a binary max heap implemented using an array

Q1:- which one of the following array represented a binary heap tree?

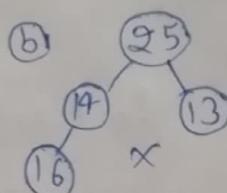
(a) 25, 12, 16, 13, 10, 8, 14



(b) 25, 14, 13, 16, 10, 8, 12



(c) 25, 14, 16, 13, 10, 8, 12



(d) 25, 14, 12, 13, 10, 8, 16

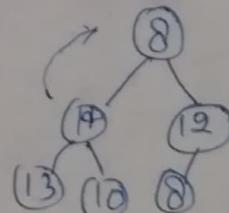
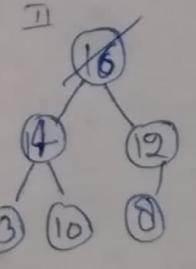
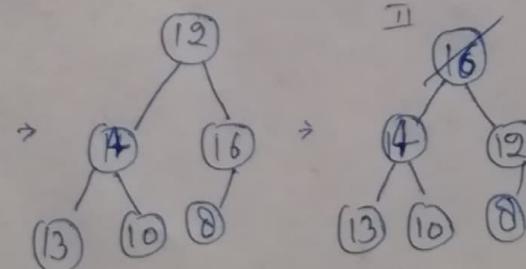
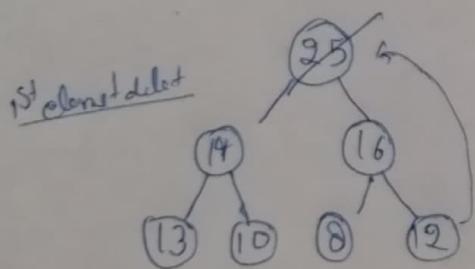
Q2:- what is the content of array after 2 delete operations
on correct answer of above question.

(a) 14, 13, 12, 10, 8

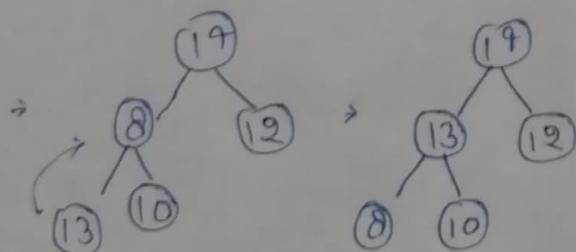
(b) 14, 12, 13, 8, 10

(c) 14, 13, 8, 12, 10

(d) 14, 13, 12, 8, 10



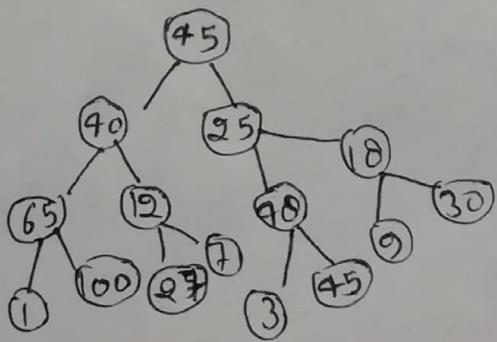
25, 16



= 14, 13, 12, 8, 10
Ans

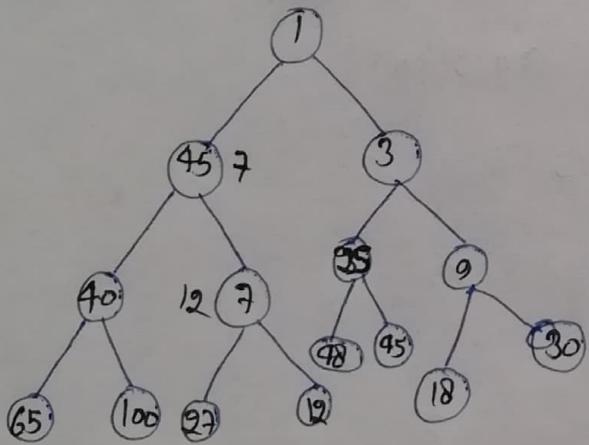
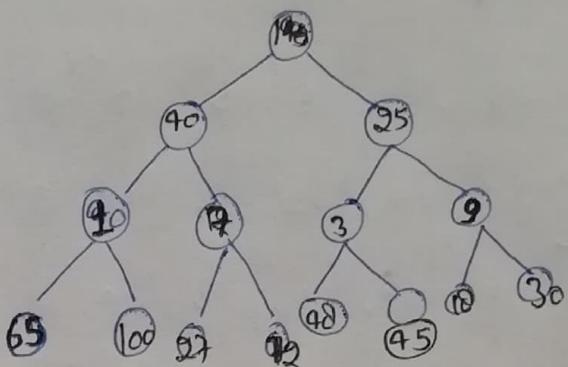
Heapsify method :-

145, 40, 25, 65, 12, 48, 18, 1, 100, 27, 7, 3, 45, 9, 30

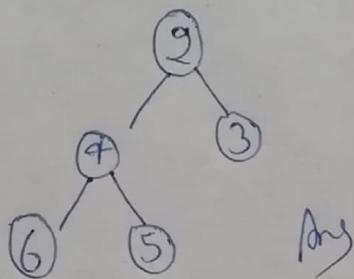
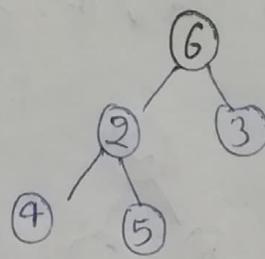
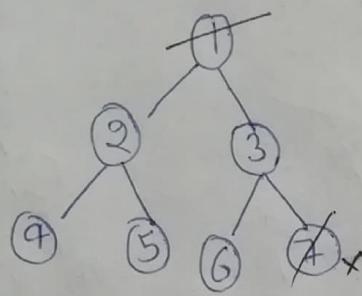


min heap?

leaf node = 0
n / 2 leaf.



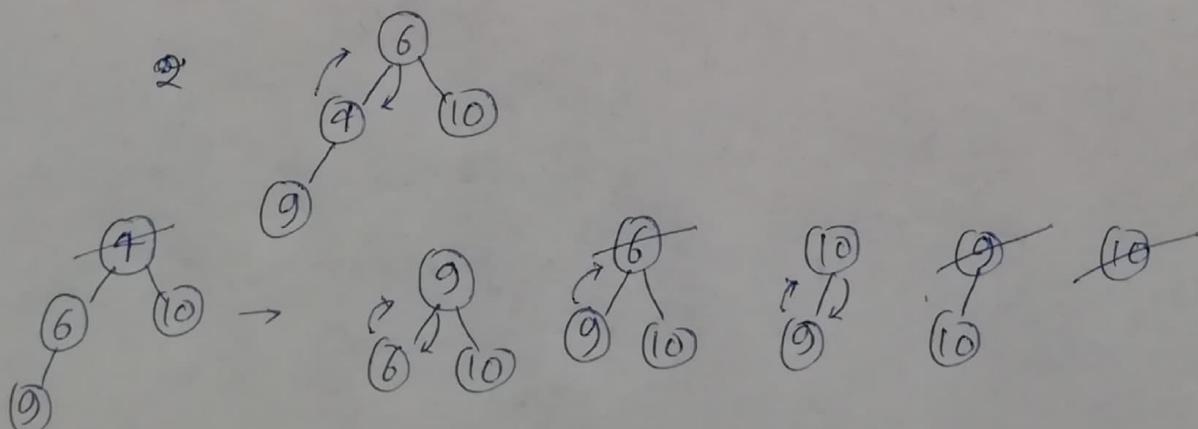
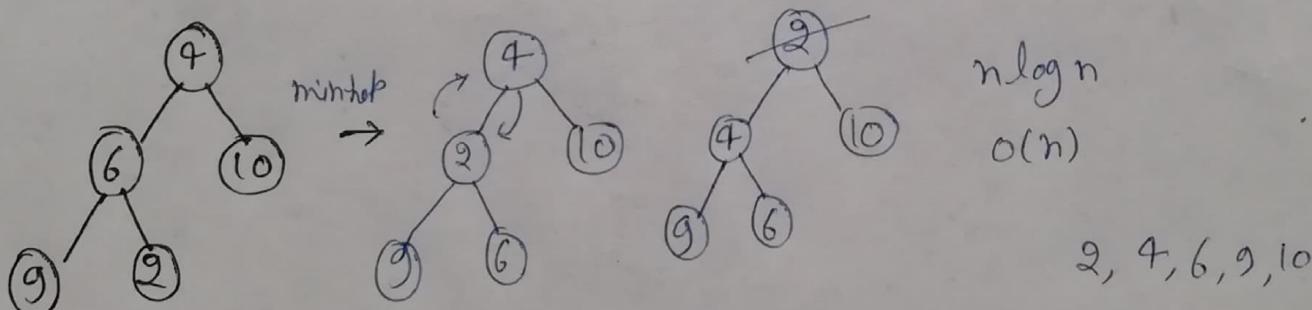
Deletion in Heaps Tree :-



• O(1) Best
O(log n) Worst

Heaps start (In-place, unstable, $O(n \log n)$)

4, 6, 10, 9, 2 \rightarrow 2, 4, 6, 9, 10

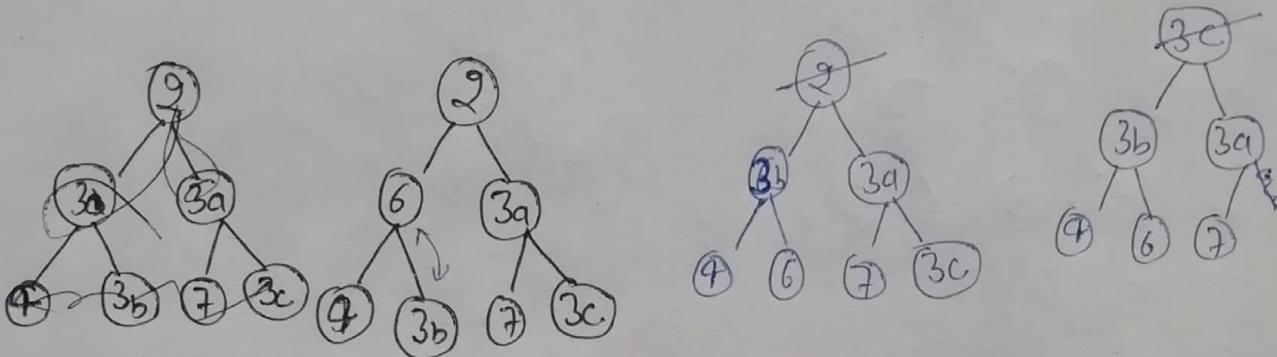


Time complexity $\rightarrow O(n) + n \log n$
 $O(n \log n)$ As

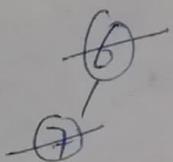
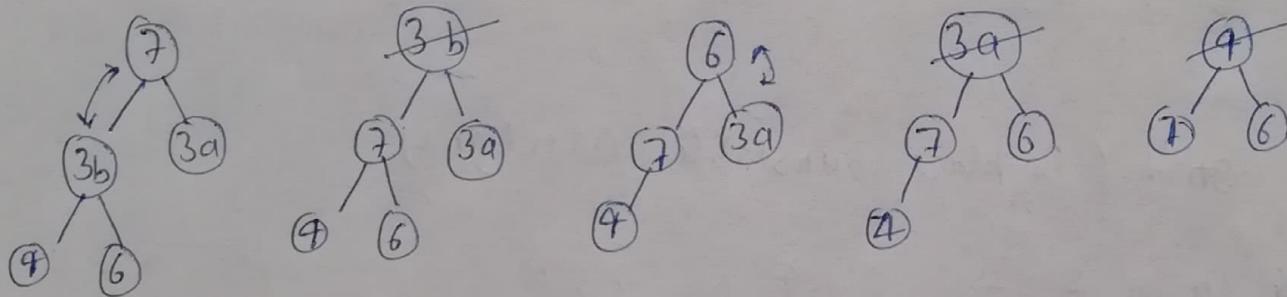
Heap Sort (imbalanced, unstable, $O(n \log n)$) :-

(26)

2, 6, 3a, 4, 3b, 7, 3c

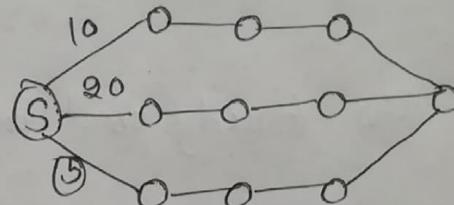


2, 3c, 3b, 3a, 4, 6, 7 ↴



~~Greedy~~ Technique (Algorithm) :-

- follows local optimal choice of each stage with intend of finding global optimum.
- feasible solution (selection)
- optimal solution



- min cost
- max profit.
- min Risk

- Application :-
- 1) knapsack problem
 - 2) Job Sequencing
 - 3) minimum spanning tree
 - 4) optimal merge pattern
 - 5) Huffman coding
 - 6) Dijkstra's Algo.

Knapsack problem :-

Objects	obj ₁	obj ₂	obj ₃	Knapsack capacity (m) = 20
Profit	25	24	15	
Weight	18	15	10	

$\frac{25}{18} = 1.3$ $\frac{24}{15} = 1.6$ $\frac{15}{10} = 1.5$

$\boxed{20}$

① Greedy about profit $\boxed{\frac{ab_i}{ab_j}}$

$$25 + \frac{24}{15} = 28.2$$

$$= 25 + \frac{48}{15} = 28.2$$

② Greedy about weight

$$\left(\begin{array}{c} \text{obj}_1 \\ \vdots \\ \text{obj}_3 \end{array} \right) \left\{ \begin{array}{l} 5 \\ 15 + \frac{10 \times 24}{15} \\ = 15 + \frac{240}{15} = 31 \end{array} \right.$$

③ P/W Greedy about both

$$\left(\begin{array}{c} \dots \\ \text{obj}_2 \end{array} \right) \left\{ \begin{array}{l} 5 \\ 15 \cdot 24 + \frac{5}{10} \times 5 \\ = 31.5 \end{array} \right.$$

on \rightarrow for $i=1$ to n calculate profit / weight

analog \rightarrow sort objects in decreasing order of P/W Ratio

\rightarrow for $i=1$ to n
if $M > 0$ and $w_i \leq M$)

$$M = M - w_i;$$

^{24 case}
also break;

if ($M > 0$)

$$\underline{P = P + P_i \left[\frac{M}{w_i} \right] \alpha}$$

Huffman Coding :- (Greedy Technique) :-

(29)

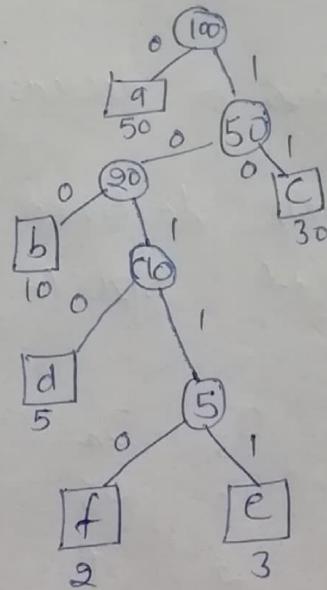
a = 50	e = 3
b = 10	f = 2
c = 30	
d = 5	

M = 100 Characters

ASCII

0 - 127

$$7 \times 100 = 700 \text{ bits}$$



$$\begin{aligned} \text{Average required to} \\ \text{represent each character} &= \frac{185}{100} \\ &= 1.85 \text{ bits/character} \end{aligned}$$

**

Q! - Consider the following message:

aa bbbba bbb ccc ddd eee ccc eoe dd eee

find the no. of bits required for Huffman encoding of above message ?

also find the avg. bits required to represent a character ?

$$a = 3$$

$$b = 7$$

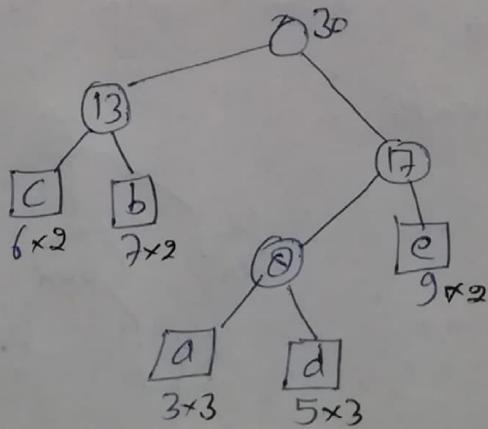
$$c = 6$$

$$d = 5$$

$$e = 9$$

13, 17

$$\frac{68}{30} = 2.26$$



$$= 50 \times 1 = 50$$

$$a = 0$$

$$b = 100 \quad 10 \times 3 = 30$$

$$c = 11 \quad 30 \times 2 = 60$$

$$d = 1010 = 5 \times 4 = 20$$

$$e = 10111 \quad 3 \times 5 = 15$$

$$f = 1d10 \quad 2 \times 5 = 10$$

185 bits

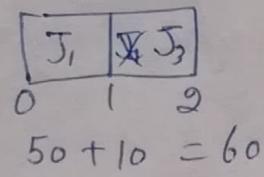
Job Sequencing (Greedy Technique) :-

(30)

Algo:-

- 1) arrange all Jobs in decreasing order of profit. ($n \log n$)
- 2) for each Job ($m i$), do linear search to find particular slot in array of size (n), where
 $n = \text{maximum deadline}$
 $m = \text{total Jobs.}$

Q!.	J_1	J_2	J_3	J_4
Profit	50	15	10	25
Deadline	2	1	2	1

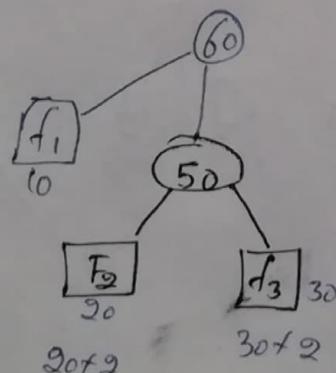


Ans. $25 + 50 = 75$ ✓ Ans

Optimal Merge pattern :-

Algorithm:-

- 1- Create a min heap with ' n ' files
- 2- At each level remove two minimum
- 3- 'a' and 'b' from min heap and put $a+b$ again in to heap.

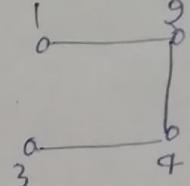
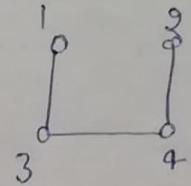
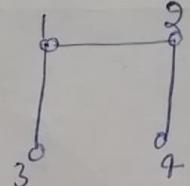
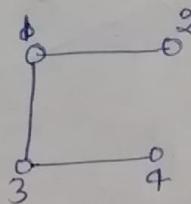
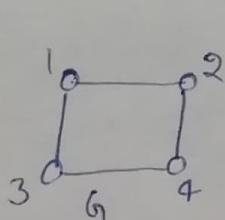


$\log n + \log n$.

(3))

Spanning Tree :- A connected subgraph 'S' of graph $G(V, E)$ is said to be spanning iff.

- 1) 'S' should contain all vertices of 'G'
- 2) 'S' should contain $(|V| - 1)$ edges.



$$(4-1) = 3$$

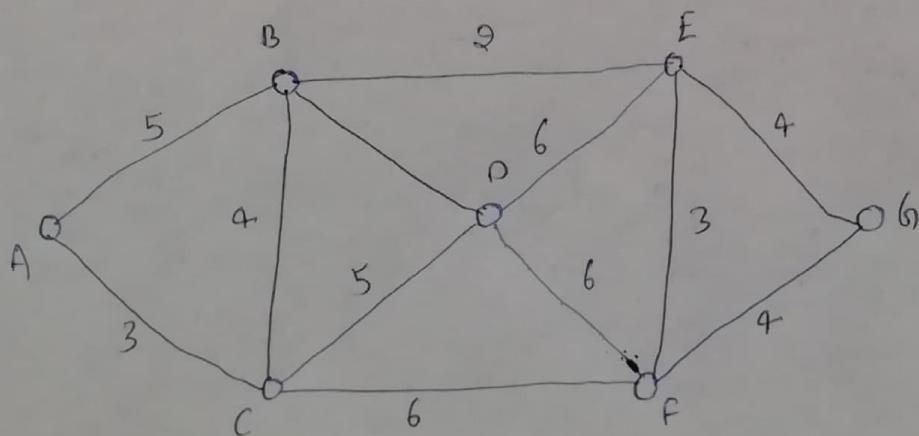
Kruskal Algo:-

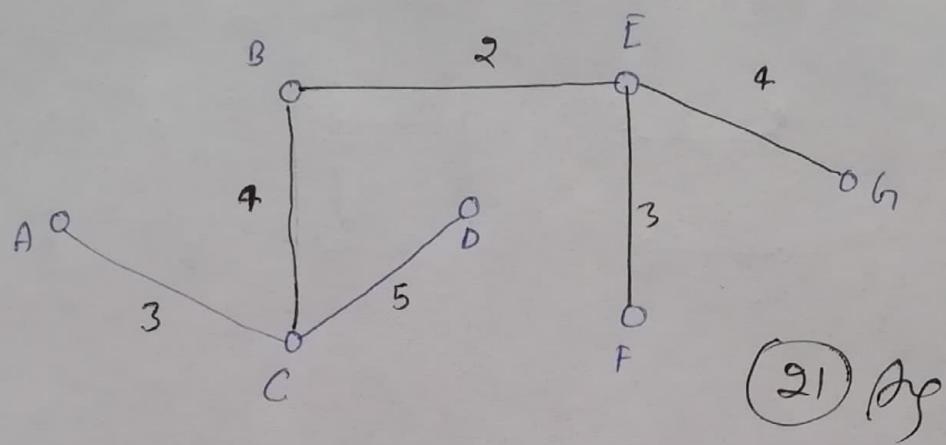
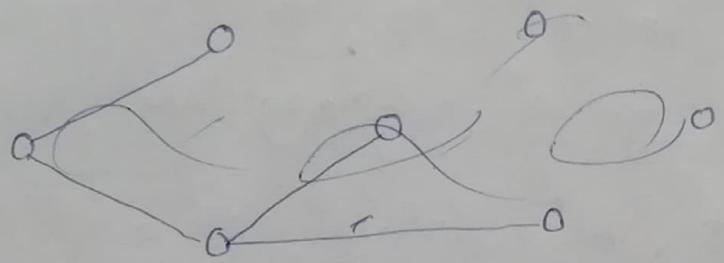
- (1) construct min heap with 'e' edge.
- (2) Take one by one edge and add in spanning tree.
(cycle should not be created).

Best case $(n-1)$ edge

worst case 'e' edge.

$(n-1) \log e$.



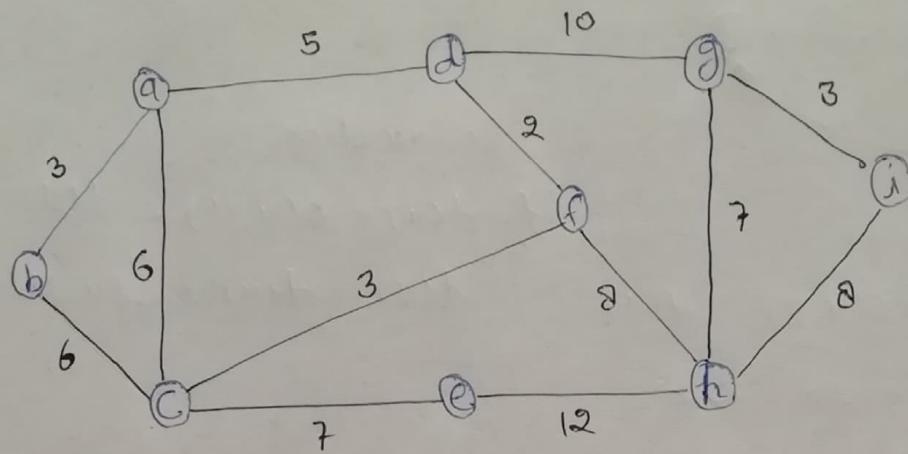


Primes

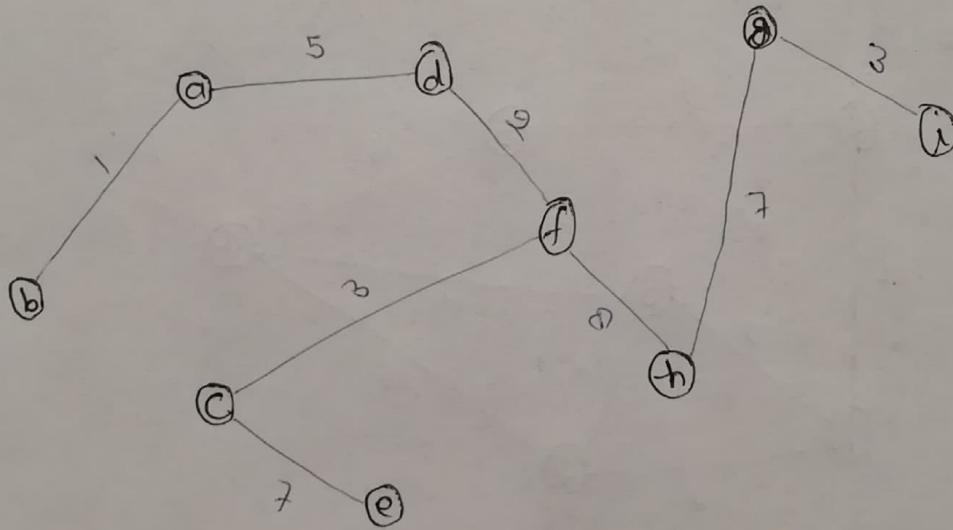
Prims Algo :-

Min cost spanning tree.

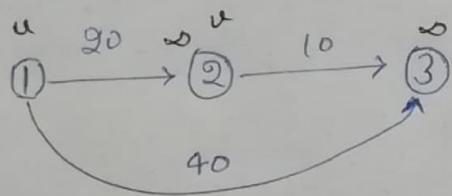
V



Sol:-



Dijkstra

Dijkstra's Algorithm (Single Source Shortest Path) :-

Relaxation

$$\text{if } d(u) + c(u, v) < d(v)$$

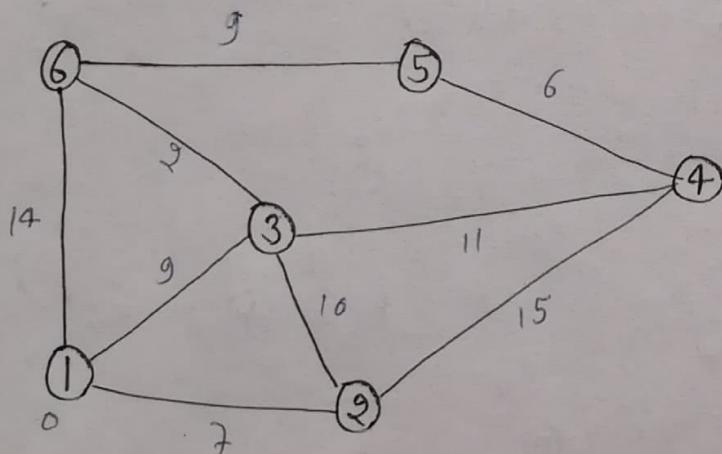
$$d(u) + c(u, v) \quad d(v) \quad d(v) = d(u) + c(u, v)$$

$$0 + 20 < \infty$$

$$0 + 40 < \infty$$

$$40 < \infty$$

Q1



Source	Destination	2	3	4	5	6
1	2	∞	∞	∞	∞	∞
1,2	3	7	9	∞	∞	14
1,2,3	4	7	9	22	∞	14
1,2,3,6	5	7	9	20	∞	11
1,2,3,6,4	6	7	9	20	20	11
1,2,3,6,4,5					20	20

Dijkstra's Algorithm Analysis :-

Dijkstra's (Graph Source)

Create vertex set α
for each vertex v in graph

$dist[v] = \infty$ $O(\alpha)$
add v to α Build Heap
 $dist[\text{source}] = 0$ $O(v)$

while α is not empty

$u = \text{Extract-min}[\alpha]$

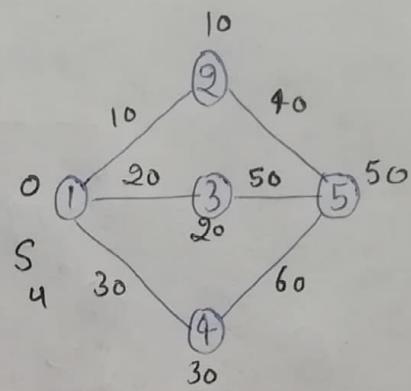
for each neighbour v of u

Relax (u, v)

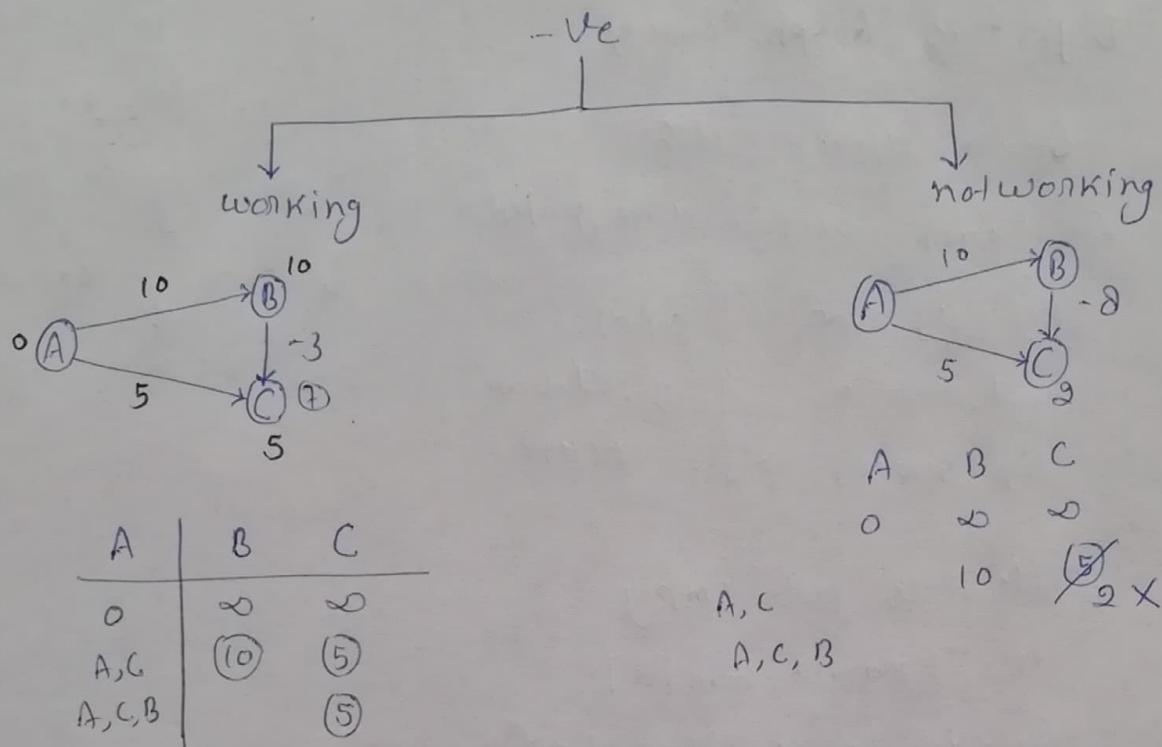
time complexity

$$O(u) + O(v) + v \log v + E \cdot \log v$$

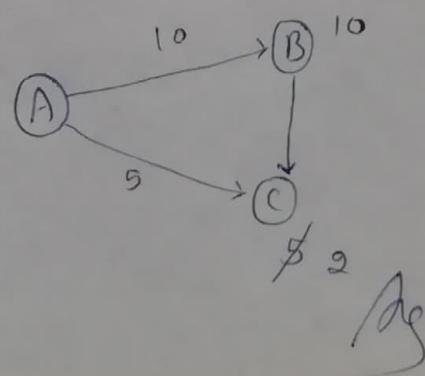
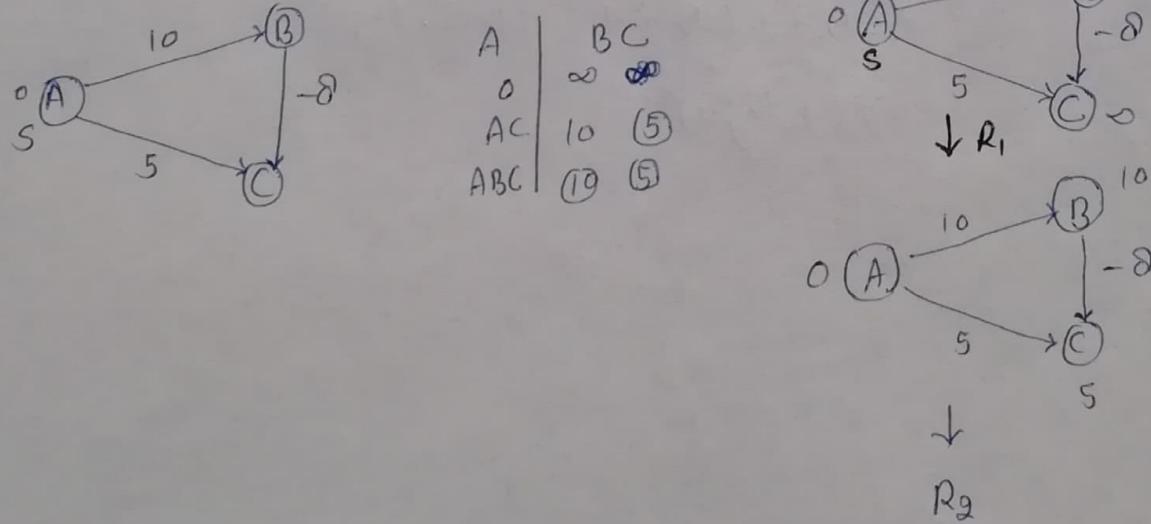
$$= \underline{\underline{O(E \log v)}}$$



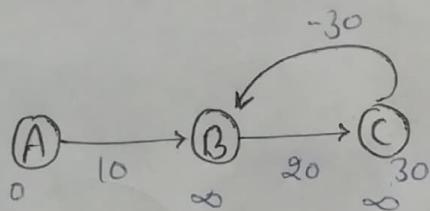
Why does Dijkstra fail on Negative Weights ??



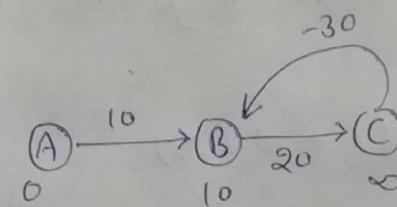
Bellman Ford Algorithm:-



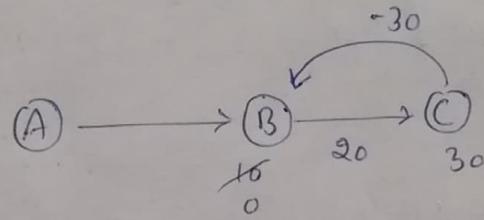
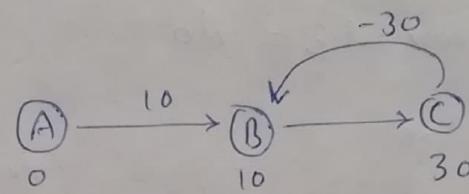
Q! -



$v-1 + 1$ time



↓



negative edge then stop.

Bellman Ford Pseudo Code :-

Bellman Ford Algorithm

Bellman Ford (G, V, E, S)

Step 1

for each vertex $v \in G$ do

$\text{dist}[v] = \infty$

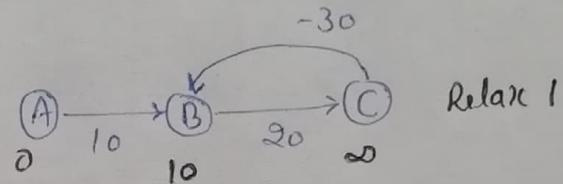
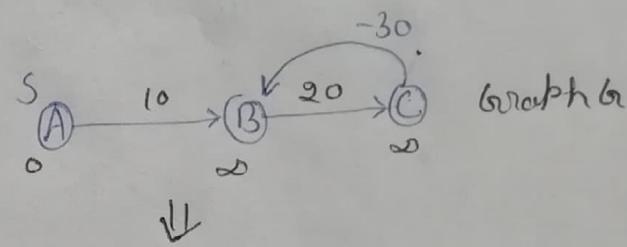
$\text{dist}[\text{source}] = 0$

Step 2

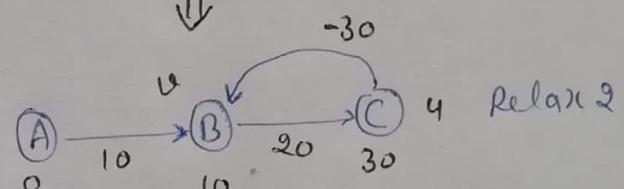
for $i=1$ to $|V|-1$

for each edge $(u, v) \in G$

Relax (u, v, w)



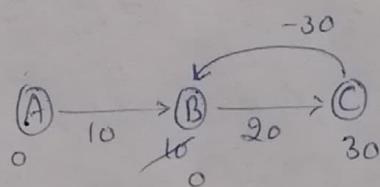
Relax 1



Relax 2

$30 - 30 < 10$

$0 < 10$



Step 3

for each edge $(u, v) \in G$

if $(\text{dist}(u) + w(u, v) < \text{dist}(v))$

return "Graph contain Negative weight cycle"

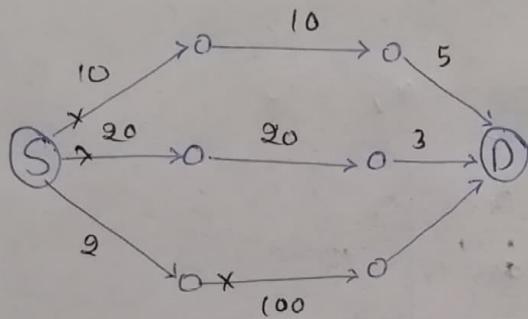
return distance.

Dynamic Programming :-

At divide the problem into

series of overlapping sub-problems.

- Two features :-
- 1) optimal substructure
 - 2) overlapping subproblems.



$$n = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$

$$f(n) = 0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13$$

$$f(n) = f(n-1) + f(n-2)$$

$$1 \quad n = 1$$

$$0 \quad n = 0$$

0/1 Knapsack failed using Greedy approach :-

0/1 Knapsack (0 - absent , 1 - present)

Object	obj ₁	obj ₂	obj ₃
weight	2	4	8
profit	20	25	60

Knapsack capacity (M) = 12

$$10 \left\{ \begin{array}{|c|} \hline 2 \\ \hline \diagup \diagdown \\ \hline \text{obj}_3 \\ \hline \diagup \diagdown \\ \hline \text{obj}_1 \\ \hline 12 \\ \hline \end{array} \right\} 12 \\ 20 + 60 \\ = 80$$

1	2	3	4	5	6	... h
↓	↓	↓	↓
2	2	2				2

ob ₁	ob ₂	ob ₃		
0	0	0	-	NP
0	0	1	-	60
0	1	0	-	25
0	1	1	-	<u>85</u> ✓ NP
1	0	0	-	20
1	0	1	-	80
1	1	0	-	45
1	1	1	-	NP

(41)

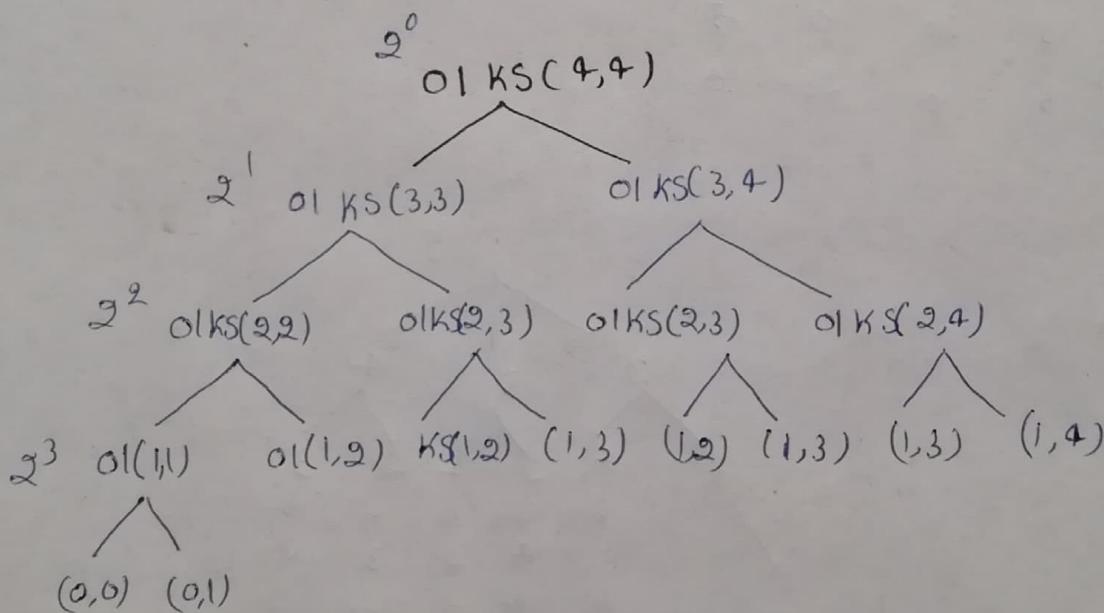
0/1 Knapsack Recursive eqn. :-

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \dots \quad n \\ 2 \times 2 \times 2 \times 2 \times 2 \dots 2 = 2^n$$

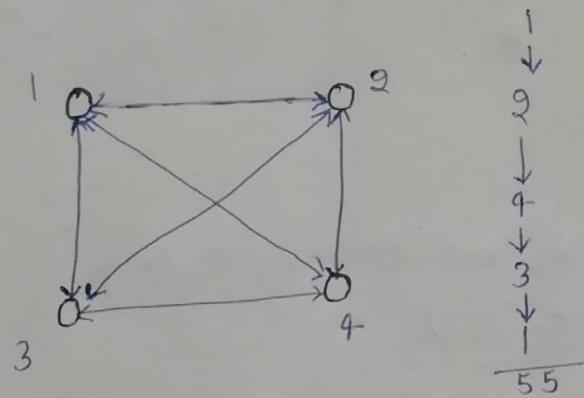
0/1 Knapsack (n, m) =

$$\begin{cases} \max & \left\{ \begin{array}{l} 0/1 \text{ KS}(n-1), (m-w_n) + p_n \\ 0/1 \text{ KS}(n-1), m \end{array} \right\} \\ 0/1 \text{ KS}(n-1, m) & w_n > m \\ 0 & n=0 \\ m=0 & \end{cases}$$

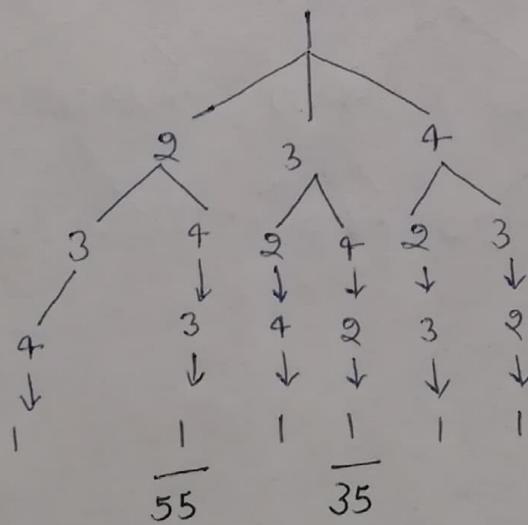
Q2 0/1 KS (4, 4)



Travelling Salesman Problem :-



	1	2	3	4
1	10	10	15	20
2	5	0	25	10
3	15	30	0	5
4	15	10	20	0



Shortest path

time complexity $n-1$

$O(n!)$

$O(n^n)$

"Sum of Subset" :-

$$S = (10, 20, 30, 40)$$

$$m = 50$$

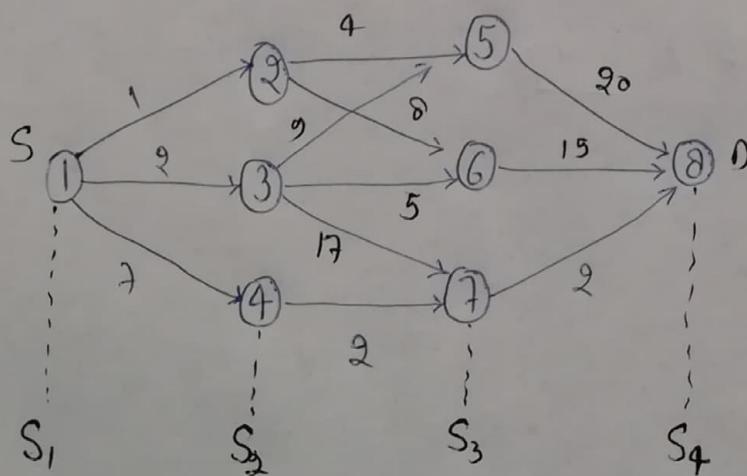
$$SOS(n, m) = \begin{cases} SOS(n-1, m) \\ SOS(n-1, m-w_n) \\ SOS(n-1, m) w_n > m \end{cases}$$

True $n=0, m=0$

False $n=0, m \neq 0$

True $n \neq 0, m=0$

Multi Stage Graph :-

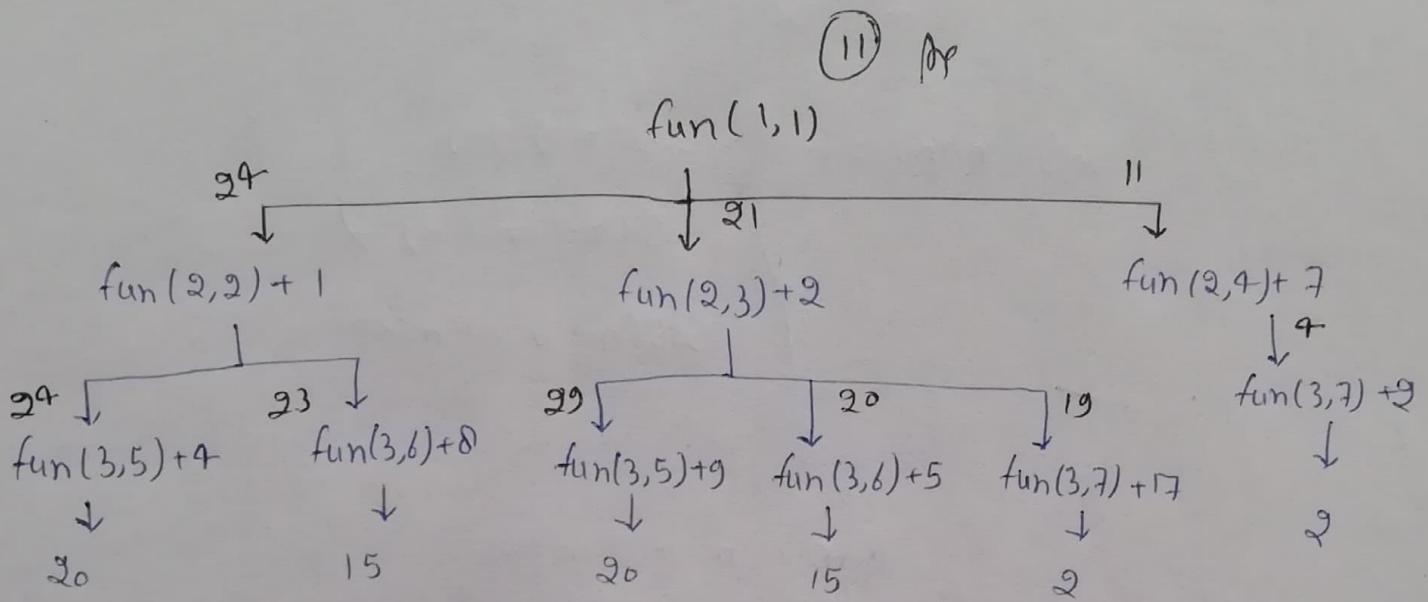


$$1 - 2 - 5 - 8 \quad (25)$$

$$\text{fun}(S_i, V_j)$$

$$\text{fun}(2, 3)$$

$$\min \left\{ \begin{array}{l} \text{fun}(S_{i+1}, K) + C(v_j, K) \\ \boxed{C(v_j, D) \quad S_j = S_i - 1} \end{array} \right.$$



time complexity

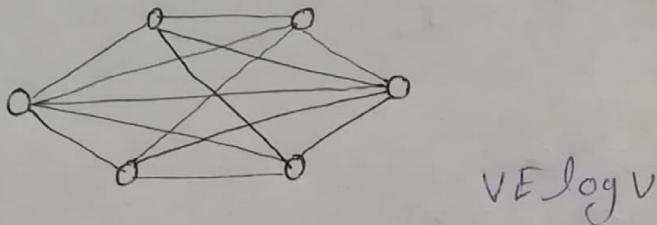
$$O(V + E)$$

$$\underline{O(E)}$$

↗

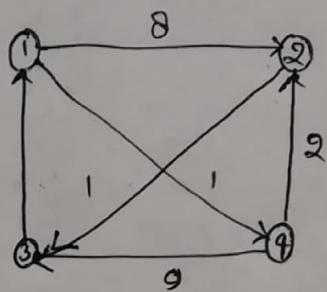
(45)

All pair Shortest path (Floyd, warshall . Algo) :-



$U.E$	$VE \log V$
$V.U.E$	$V.V^2 \log V$
$V^2 E$	$V^3 \log V$ <u>diff'stroy</u>
$V^2 \cdot W^2$	
(V^4)	

Floyd Warshall working with Ex. :-



$$D = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 8 & \infty \\ 2 & \infty & 0 & 1 \\ 3 & 4 & 12 & 0 \\ 4 & \infty & 2 & 9 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 8 & \infty \\ 2 & \infty & 0 & 1 \\ 3 & 4 & \infty & \infty \\ 4 & \infty & 2 & 9 \end{bmatrix}$$

$$D^2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 8 & 9 \\ 2 & \infty & 0 & 1 \\ 3 & 4 & 12 & 0 \\ 4 & \infty & 2 & 3 \end{bmatrix}$$

$$D^3 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

$$D^4 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

n^2
 $n^2 \times n$
 $\mathcal{O}(n^3)$

$$\text{Time complexity} = n^2 \times n$$

$\mathcal{O}(n^3)$

$$\text{Space complexity} = n^2 \times 2$$

$\mathcal{O}(n^2)$

Hashing (Storing and Retrieving Data in O(1) time) :-

- * Search Key (24, 52, 91, 67, 48, 83)
- * Hash table
- * Hash function (K mod 10, K mod n, Mid Square, Folding Method)

$$K \bmod 10$$

$$24 \bmod 10 = 4$$

$$\text{S-Hash value} = 4$$

	0
91	1
52	2
83	3
24	4
	5
	6
67	7
48	8
	9

Hash Table

Collision in Hashing :-

Collision Resolution

Techniques

Chaining
(Open Hashing)

$$h(\text{value}) = 0$$

$$0+1^2 \bmod 6 = 1$$

$$0+2^2$$

$$4 \bmod 6 = 4$$

$$R+i^2 \bmod$$

$$30 \bmod 6 = 0$$

open Addressing
(closed Hashing)

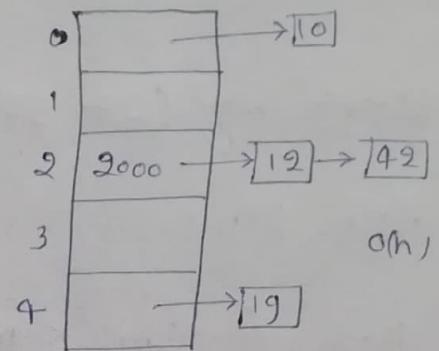
- Linear probing
- Quadratic probing
- Double Hashing

Chaining :- (Open Hashing)

Keys : 42, 19, 10, 12

$k \bmod 5$

Advantages :- Deletion is easy
 $O(1)$ Insert



Disadvantages! - $O(n)$

Extra space लेता है।

Load factor :-

$$\alpha = \frac{\text{elements / key}}{\text{slots}} = \frac{4}{5}$$

Linear probing :-

$$h(k) = k \bmod 10$$

$$h'(k,i) = (h(k)+i) \bmod 10$$

Keys: 43, 135, 72, 23, 99, 19, 82

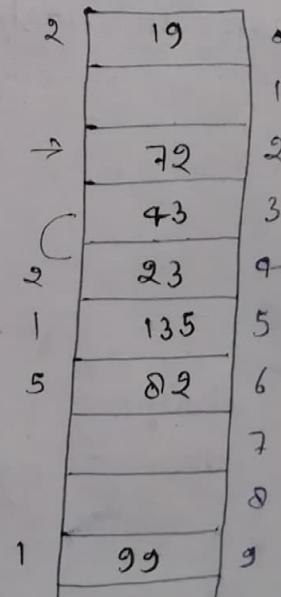
Advantages:- No extra space

Disadvantage:- Search time $\in O(n)$

Deletion Difficult

Primary clustering

Secondary clustering



(49)

Q:- The keys 1, 3, 12, 4, 25, 6, 18, 20, 8 are inserted into empty hash table of length 10 using open addressing with hash function $h(i) = i^2 \bmod 10$ and linear probing what is the resultant hash table and find the maximum probe value?

$$h(i) = i^2 \bmod 10$$

$$1^2 \bmod 10 \quad 1^2 \bmod 10$$

$$4^2 \bmod 10 \quad 3^2 \bmod 10$$

$$9^2 \bmod 10 \quad 6^2 \bmod 10$$

Probe value of 8 $\Rightarrow 9$

1 \rightarrow	20	0
	1	1
	8	2 $\leftarrow (9)$
		3
	12	4
	25	5
	4	6
	6	7
	18	8
	13	9

maximum probe value of 9 $\rightarrow 8$ β

Quadratic Probing :-

$$h(k) = k \bmod 10$$

$$h'(k,i) = h(k) + i^2 \bmod 10$$

Keys : 42, 16, 91, 33, 18, 27, 36, 62

Advantage :- No extra space

Disadvantage :- Search $O(n)$

- * Secondary clustering
- * No guarantee of finding slot.

$$6+1^2 = 7$$

$$6+2^2 = 10$$

$$2+1^2 = 3$$

$$2+2^2 = 6$$

$$2+3^2 = 11$$

$$2+4^2 = 18$$

$$2+5^2 = 27$$

36	0
91	1
42	2
33	3
	4
	5
16	6
27	7
18	8
	9

Double Hashing :-

$$f_1(k) = k \bmod 11$$

$$f_2(k) = 8 - (k \bmod 8)$$

$$(f_1(k) + i \cdot f_2(k)) \bmod 11$$

$$1 + 3 = 4$$

56

1

$$8 - (56 \bmod 8)$$

$$8 - (45 \bmod 8)$$

$$8 - 5$$

3

$$8 - (70 \bmod 8)$$

6

Keys: 90, 34, 45, 70, 56

$$f(k) + i$$

$$f(k) + i^2$$

0	
1	34
2	
3	56
4	45
5	
6	70
7	
8	20
9	20
10	

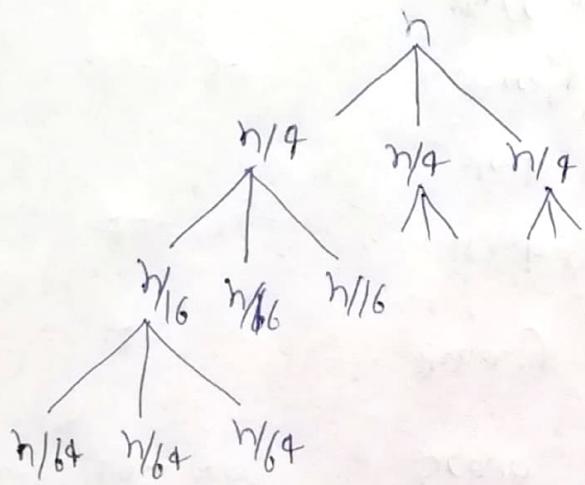
Advantage :- No extra space

No primary clustering

No Secondary " "

Disadvantage :- $O(n)$

Recurrence Relation $[T(n) = 3T(n/4) + cn^2]$



$$cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \left(\frac{3}{16}\right)^3 cn^2 + \dots$$

$$cn^2 \left[1 + \frac{3}{16} + \left(\frac{3}{16}\right)^2 + \left(\frac{3}{16}\right)^3 \dots \right]$$

$$1 + \alpha + \alpha^2 + \alpha^3 + \dots$$

$$cn^2 \left[\frac{1}{1 - \frac{3}{16}} \right] \xrightarrow{\text{1 - } \alpha}$$

$$\underline{cn^2 \frac{16}{13}}$$