Name :- junaid Qazi

Stu.- MANUU HYD

DAA

Index

39- 41 Dynamic programming
0/1 knapsack failed using greedy approach
0/1 knapsack Recursive eqn.

42- 44 Travelling Salesman Problem, sum of subset
Multistage 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 $[T(n) = 3T(n/4) + cn^2]$

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, Dijstras, 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] Gate ×

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.

S_1 : Read A

S_2 : Read B

S_3 : Sum = A+B

S_4 : Print (Sum)

Any Analysis

Priory — Posterion

APP. — Exact

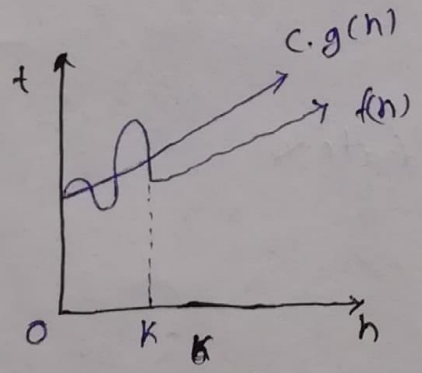
Independent — Dependent

uniform .....

```
main ()
int a, b, sum1
Scanf (a, b)
    Sum1 = a + b;
    Pf (sum1);
```

Asymptotic Notation :-

(i) Big- oh (O)



→ worst case

→ upper Bound (At most)

$f(n) = O(g(n))$

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

$c > 0$

$n \geq K$

$K \geq 0$

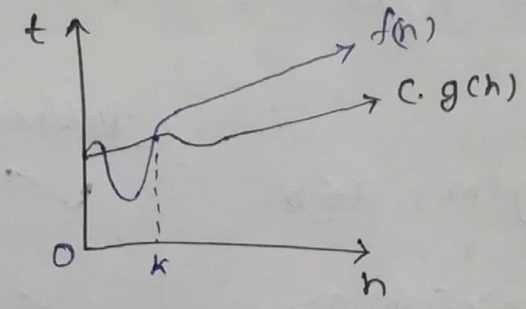
$f(n) = 2n^2 + n$ $\quad f(n) = O(\ )$

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

$n \leq n^2$

$n \geq 1$

(2) Big - Omega (Ω)



→ Best case

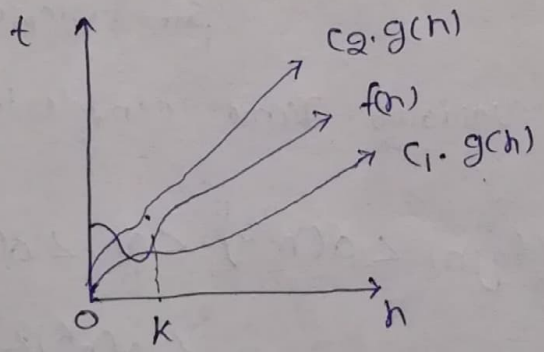
→ Lower Bound (At least)

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

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

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

(3) Theta (Θ) : -



→ Average case

→ Exact time

$C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$

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

Various Property of Asymptotic Notation :-

$f(n) = o(g(n))$ ↗ $g(n) \neq o(h(n))$

	Reflexive	Symmetric	Transitive
Big(O), $f(n) \leq c \cdot g(n)$, $a \leq b$	✓	×	✓
Big Omega (Ω), $f(n) \geq c \cdot g(n)$ $a \geq b$	✓	×	✓
Theta (Θ), $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ ($1 \cdot n^2$, $2n^2$, $3 \cdot n^2$)	✓	✓	✓
Small (o), $f(n) < c \cdot g(n)$ (n^2, n^3) $a = b$	×	×	✓
Small(ω), $f(n)^2 > c \cdot g(n)$ (n^2, n) $a > b$	×	×	✓

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

$a \not< b$

Same value put

Comparision of various Time complexities :-

$O(c) < O(\log\log n) < O(\log n) < O(n^{1/2}) < O(n) < O(n \log n) < O(n^2) < O(n^3) \rightarrow$

$\rightarrow < O(n^k) < O(2^n) < O(n^n) < O(2^{2^n})$

* Time complexities of all Searching and Shorting 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. worst $O(n)$
- Bubble Sort → $O(n^2)$
- Heap Sort → $O(n \log n)$
- Selection Sort → $O(n^2)$

- Height of CBT = $O(\log n)$
- Insertion in Heap $(\log n)$
- Construnct Heap $(n \log n)$
- Delete from Heap $(\log_2 n)$
- Huff man $(n \log n)$
- Prims matrix → $O(n^2)$
- Kruskal $O(E \log E)$
- DFS, BFS $O(V+E)$
- All pair Shortest $O(n^3)$
- Dijkstra $O(V^2)$

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

✓(a) $f(n) = O f_2(n)$ (b) $f_1(n) = \Omega f_2(n)$

Let $n = 16$

$f_1 = 16^2 \times 4$ $f_2 = (16)^8 \log_2 16$

$16 \times (4)^{10}$

10^9

Let $= 10^9$

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

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

$10^9 \times 9$

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

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

9×9^9

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

Arranging increasing order

a) $f_2\ f_3\ f_4\ f_1$

b) $f_1\ f_2\ f_3\ f_4$

b) $f_2\ f_1\ f_3\ f_4$

d) $f_3\ f_2\ f_4\ f_1$

Let $n = 256$

2^{256}

$256^{3/2}$

$2^{8*3/2}$

2^{12}

$(256)\log_2(256)$

$256)\log_2 2^8$

$2^8 * 8$

2^{11}

$(256)^{\log_2 256}$

$(256)^8$

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

$f_3 < f_2 < f_4 < f_1$

Recurrence Relation :-

```
BS (a, i, j, x)
    mid = (i+j)/2
      if(a[mid] == x)
      return (mid);
    else
        if(a[mid] > x]
    BS(a, i, mid -1, x)
      else
    → BS (a, mid +1, j, x)
```

10, 20, 30, (40), 50, 60, 70

10, 20, 30 | 50, 60, 70

1 2 3

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

Substitution Method :-

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

$T(n) = T(n/2) + C \longrightarrow (1)$

$T(n/2) = T(n/4) + C \longrightarrow (2)$

$T(n/4) = T(n/8) + C \longrightarrow (3)$

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

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

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

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

⋮

K time

$n = 2^k$

$\log n = k \log 2$

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

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

$T(n/n) + kC$

$T(1) + kC$

$O(\log_2 n)$ Ans

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

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

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

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

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

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

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

⋮

$(n-1)$ step3

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

$T\ \cancel{x-x}+1$

$T(1)$

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

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

$O(n^n)$ Ans

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

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

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

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

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

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

Put equation No 3-

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

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

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

Run k time

$$2^k T\left(\frac{n}{2^k}\right) + kn$$

Let

$n = 2^k$

taking log on both --

$\log n = \log 2^k$

$\log n = k \log 2$

$\log n = k$

$$n\, T\left(\frac{2^k}{2^k}\right) + n \log n$$

$n + n\log(n)$

$O(n \log n)$ Ans

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

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

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

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

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

k time

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

$n-k = 1$

$n = k$

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

$= T(1) + \log(1) + \log(2) + \log(3) \cdots \log n$

$= 1 + \log(1 \cdot 2 \cdot 3 \cdot 4 \cdots n)$

$= 1 + \log \lfloor n$

$= 1 + \log n^n$

$1 + n \log n$

$O(n \log n)$ Ans

***

Master Method :-

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

$a \geq 1$, $b > 1$

→ Solution is :-

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

→ $U(n)$ depends on $h(n)$

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

→ Relation between $h(n)$ and $U(n)$ is →

if $h(n)$	$U(n)$
$n^s, n>0$	$O(n^s)$
$n^s, s<0$	$O(1)$
→ $(\log n)^i, i \geq 0$	$\frac{(\log_2 n)^{i+1}}{i+1}$

Q:- $T(n) = 8T(n/2) + n^2$

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

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

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

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

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

$U(n)$ depends on $h(n)$

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

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

$= n^3 \, O(1) \quad = n^3$ Ans

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

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

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

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

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

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

$$= \log_2 n \cdot C$$

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

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

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

$$n = 2^m$$

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

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

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

$$a < b^K$$

$$1 < 2^1$$

$$= n^K \log^b n$$

$$= m^1$$

$m \Rightarrow \log n$

$$T(n) = aT(n/b) + n^K \log_n^b$$

$(n)^{1/2}$

$(2^m)^{1/2}$

$2^{m/2}$

$a = 1$

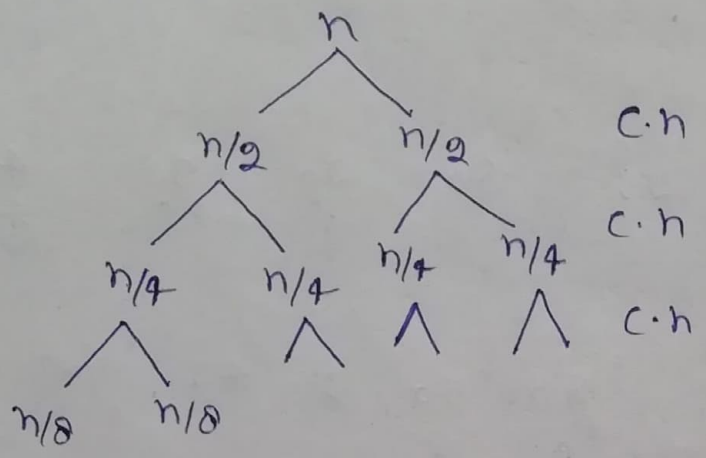
$b = 2$

$K = 1$

$b = 0$

Q:-

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



$c.n \log n$

↓

$O(n \log n)$

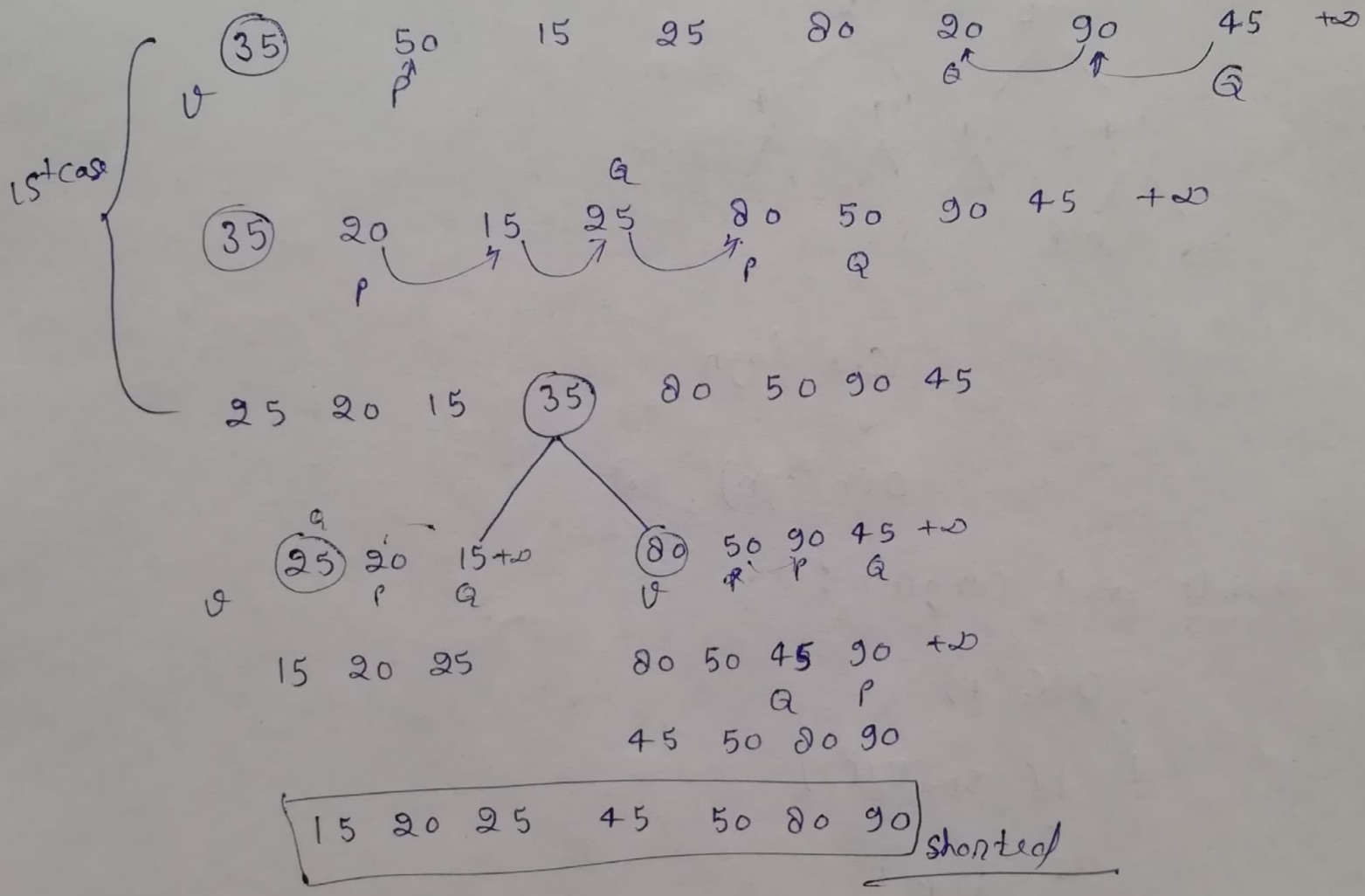
Divide and Conquer :-

```
DAC(P)
{
  if (small(P))
  {
    S(P);
  }
else
{
  divide P into P1 P2 P3 ... PK
  Apply DAC(P1), DAC(P2) ... DAC(PK)
  Combine( DAC(P1) , DAC(P2) ..... DAC(PK)
}
}
```

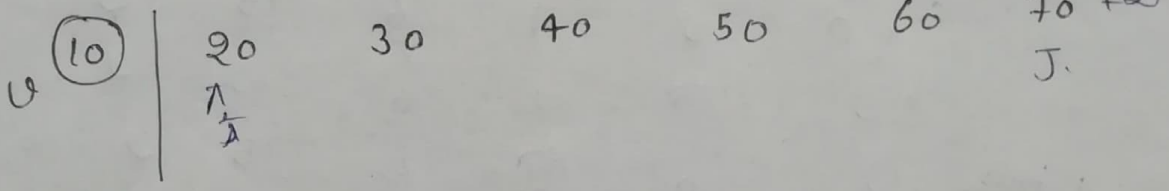
(1) Binary Search (2) Find maximum and minimum

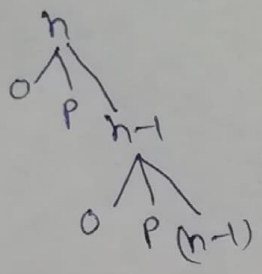
(3) Quick Sort (4) Merge Sort (5) Strassen's matrix multiplication.

Quick Sort :- (DAC)



Performance of Quick Sort:-





(20)

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

$O(n^2)$

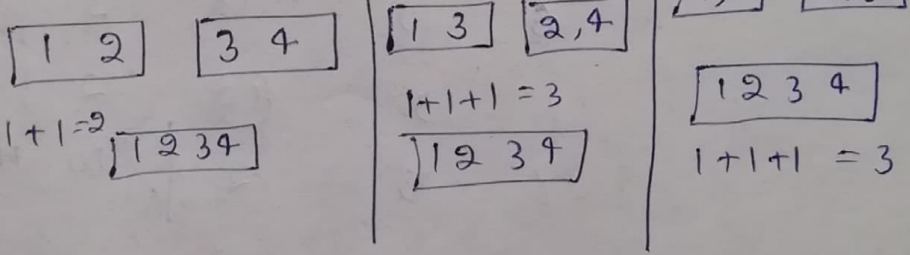
** Q:- The average no. of comparisions performed by merge sort algorithm, in merging 2 sorted list of length 2 is 2.67

n log n

4. $\log_2 4$

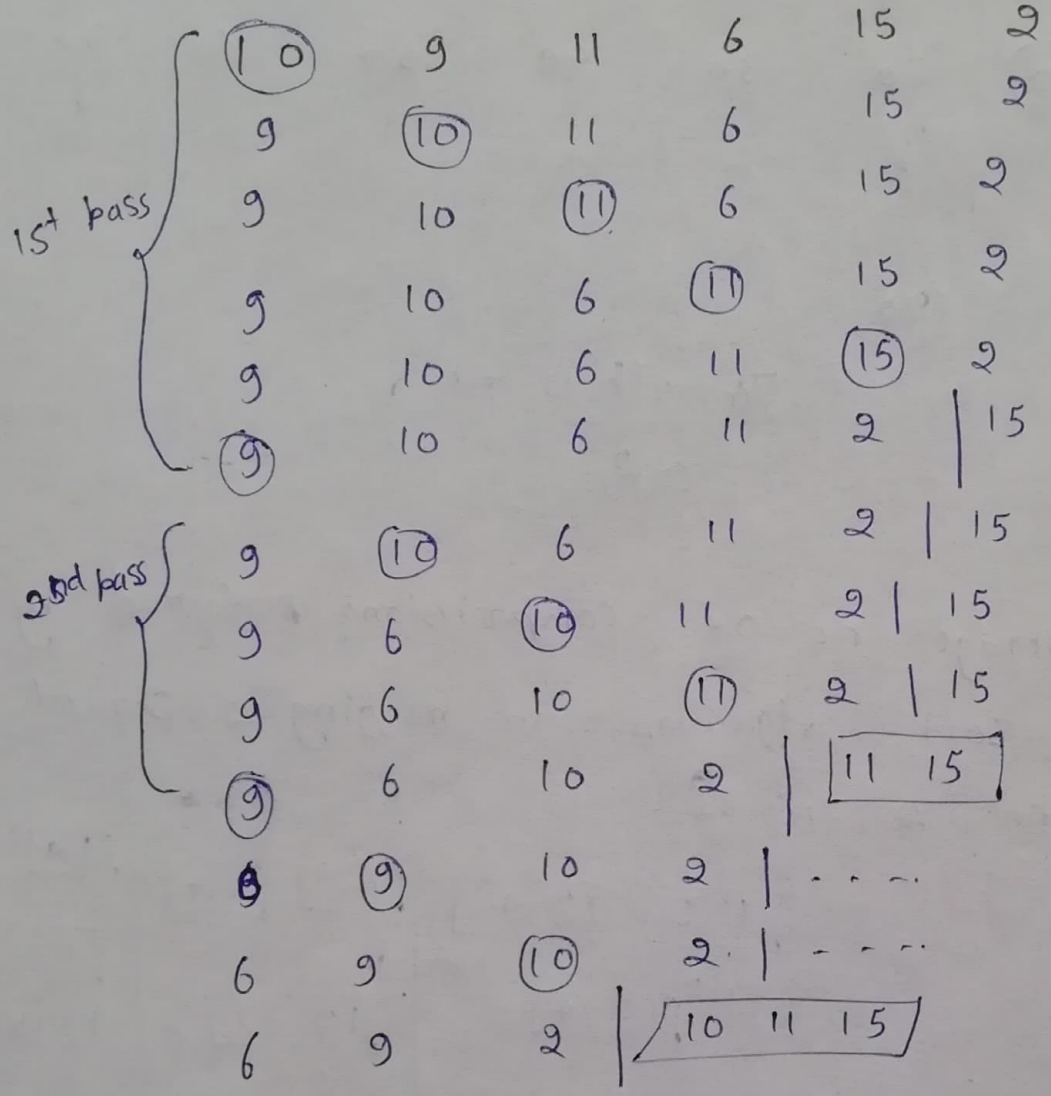
4×2

8/4 = 2 Ans



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

Bubble Sort :-



Time complexity :-

$n-1$

$n-2$

$n-3$

$\vdots$

1

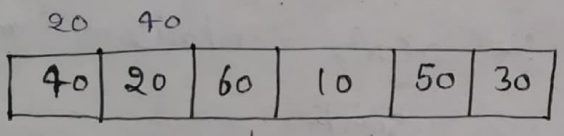
$n \times (n-1)$

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

$= O(n^2)$

Insertion-Sort(A) :-

```
for j = 2 to A.length
    key = A[j]
// Insert A[j] into the sorted sequence A[1..j-1].
i = j-1
while i>0 and A[i]>key
    A[i+1] = A[i]
    i = i-1
A[i+1] = key
```



20 40

10 20 40 50 60

10 20 40 50 60

10 20 30 40 50 60

Ω

Best case (Asending order) :-

	Comparing	Swap
10 20 30 40 50 60	0	0
	1	
	1	
	1	
	(n-1)	

$\Omega(n)$ Ω

Worst Case (Desending order) :-

60 50 40 30 20 10

~~10 20 30 40 50 60~~

Comp	Swap
0	0
1	1
2	2
3	3
:	:
:	:
n-1	n-1

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

$O(n^2)$

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

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 ← 1 to n-1 do
min j ← i;
min x ← A[i]
for [j] ← i+1 to n do
If A[j] < min x then
min j ← j
min x ← A[j]
A[i] ← 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 indexmin ! = 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 list[j] < list[min] then
        min = j;
    end if
end for
/* Swap the minimum element with the current element */
if indexmin i=1 then
Swap list[min] and 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 (desending order):-

60 50 40 30 20 10

min 60
~~56~~
~~40~~
~~36~~
~~20~~
10

(n-1)

~~50~~
~~40~~
~~30~~
20

10 | 50 40 30 20 10

10 20 | 50 40 30

10 20 30 | 50 40

10 20 30 40 | 50

10 20 30 40 50

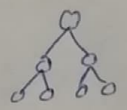
Binary Tree and its Variations (Full BT, ACBT, Complete BT, BST)

→ Binary Tree :- At most 2 children

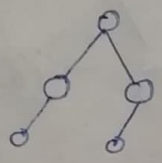
leaf nodes

→ Full (→ strict) BT :- Either 0 or 2 children

→ Complete BT :- No Holes →



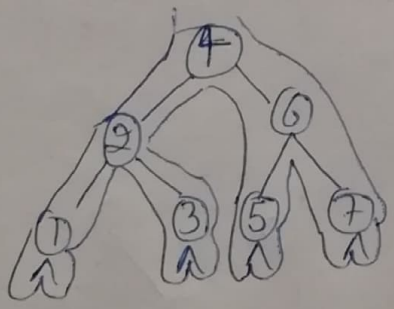
→ ACBT :-



→ BST :-

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

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



Inorder 1 2 3 4 5 6 7

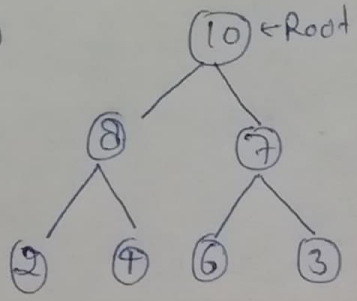
* Introduction of "Heap Tree" (Insertion, Deletion, Heap sort, Heapify, Array representation, Priority

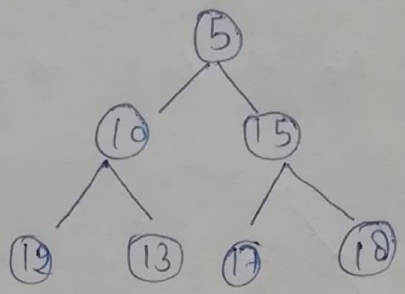
1- Structual property
2- ordering property

max Heap
(Parents > child)

min Heap
(Parents < child)

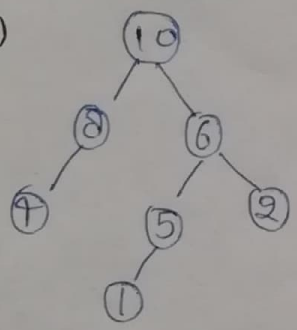
(ACBT)



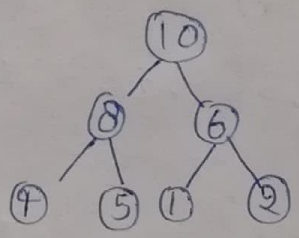


Q!- which of the following is max. Heap?

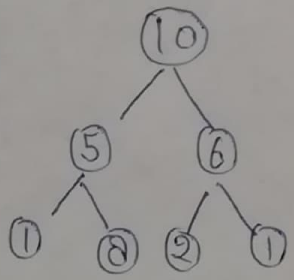
(A)



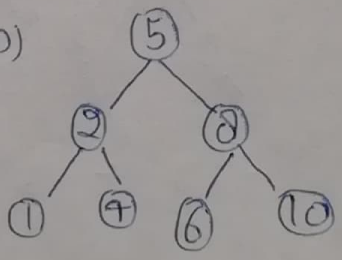
(B)



(C)



(D)



Consider a binary max heap implemented using an array

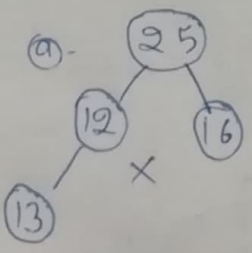
Q:- which one of the following array represented a binary heap tree?

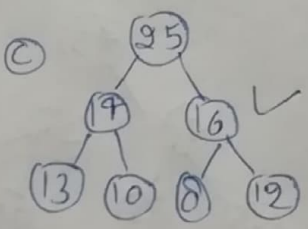
(a) 25, 12, 16, 13, 10, 8, 14

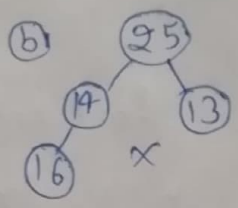
(b) 25, 14, 13, 16, 10, 8, 12

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

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







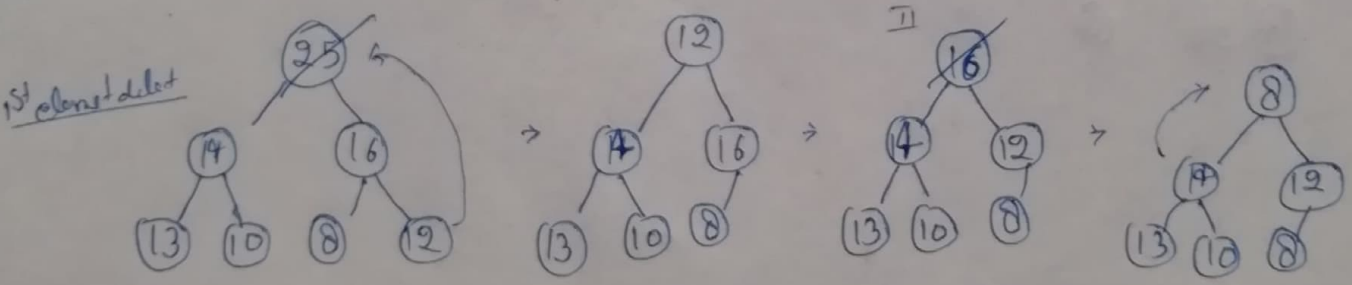
Q:- what is the content of array after 2 delete operations on correct answer of above Question.

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

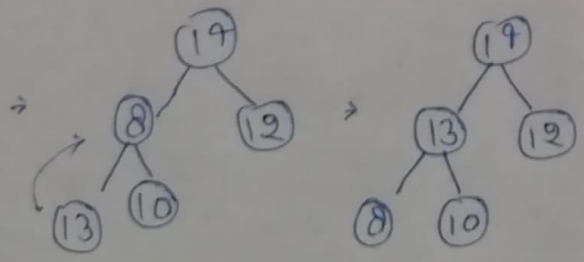
(b) 14, 12, 13, 8, 10

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

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



25, 16

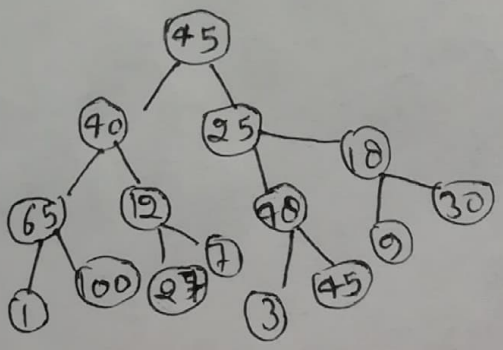


= 14, 13, 12, 8, 10

Ans

Heapify method :-

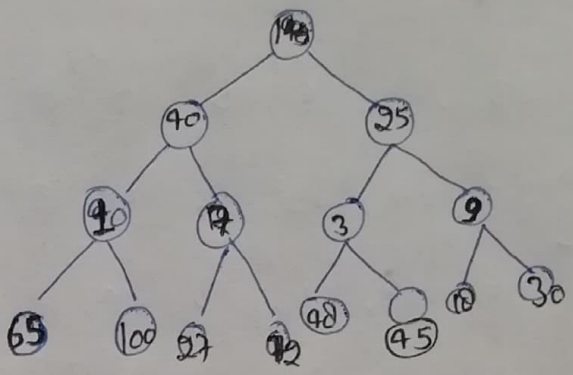
145, 40, 25, 65, 12, 48, 18, 1, 100, 27, 7, 3, 45, 9, 30

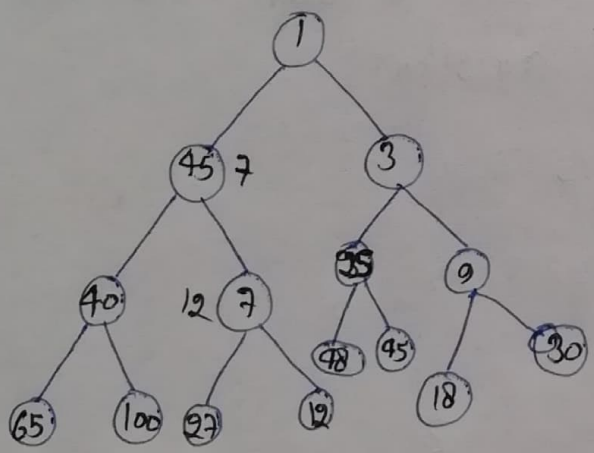


min heap ?

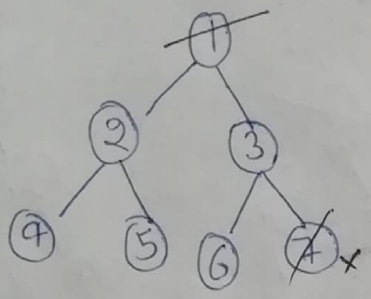
leaf node = 0

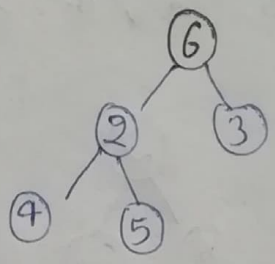
n/2 leaf.

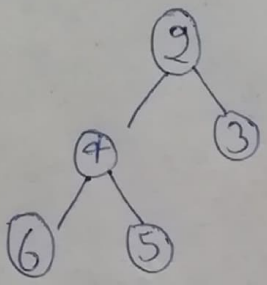




Deletion in Heap Tree :-







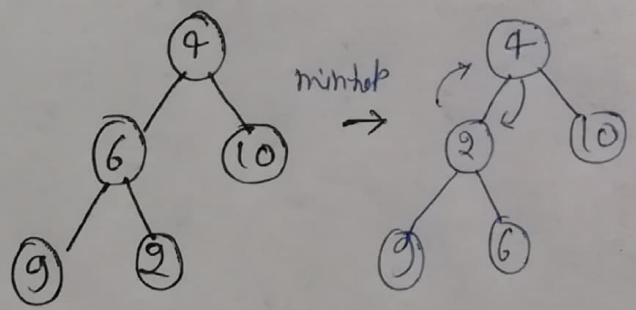
Ans

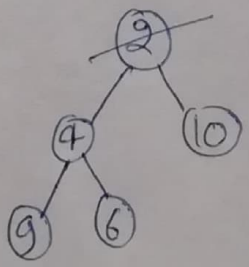
O(1) Best

O(logn) worst

Heap sort (In-place , unstable, O(nlogn))

4, 6, 10, 9, 2 → 2, 4, 6, 9, 10



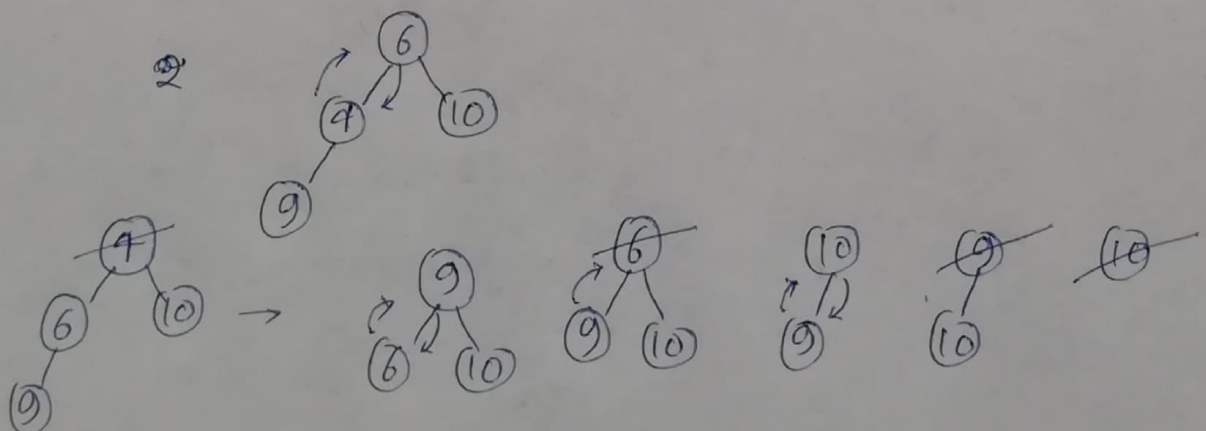


nlogn

O(n)

2, 4, 6, 9, 10

2

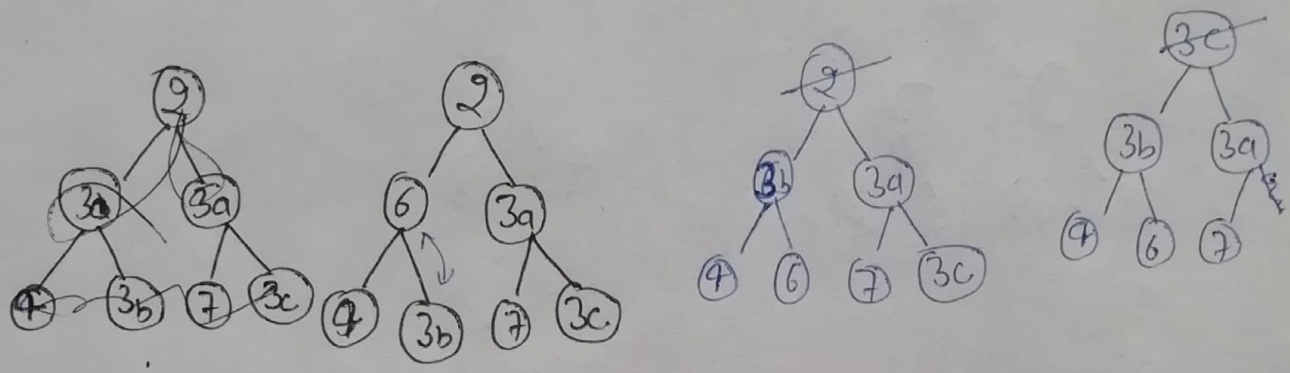


Time complexity → O(n) + nlogn

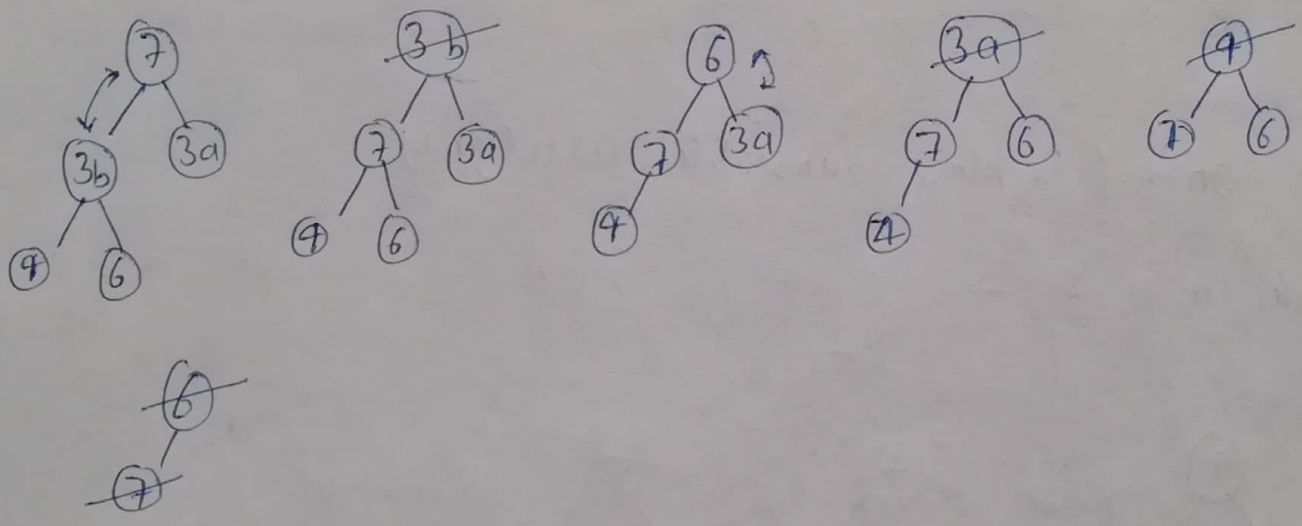
O(nlogn) Ans

Heap Sort (inplace, unstable, O(n log n)) :-

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



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

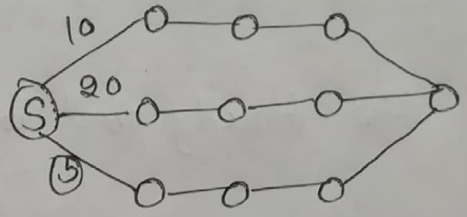


Greedy Technique (Algorithm) :-

→ follows local optimal choice of each stage with intend of finding globle optimum.

→ feasible solution (selection)

→ optimal solution



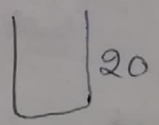
→ min cost

Application :-

1) knapsack problem → max. profit.
2) Job Sequencing → min Risk
3) minimum spaning tree
4) optimal merge pattern
5) Huffman Coding
6) Dijkstra's Algo.

Knapsack problem :-

knapsack capacity (m) = 20



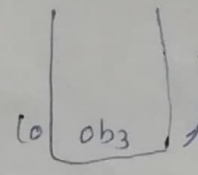
objects	ob1	ob2	ob3
profit	25	24	15
weight	18	15	10
	1.3	1.6	1.5

① Greedy about profit

$$25 + \frac{2}{15} \times 24$$

$$= 25 + \frac{48}{15}^{3.2} = 28.2$$

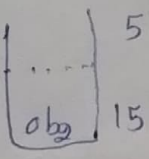
② Greedy about weight



$$15 + \frac{10}{15} \times 24$$

$$= 15 + \frac{240}{15} \quad = 31$$

③ P/w Greedy about both



$$24 + \frac{5}{10} \times 5$$

$$= 31.5$$

on [→ for i = 1 to n Calculate Profit / weight

$O(n \log n)$ → Sort objects in decreasing order of P/w Ratio

→ for i = 1 to n

if (m > 0 and $w_i \leq m$)

$m = m - w_i$;

also break;

if(m > 0)

$P = P + P_i \left[\frac{m}{w_i} \right]$

Huffman Coding :- (Greedy Technique) :-

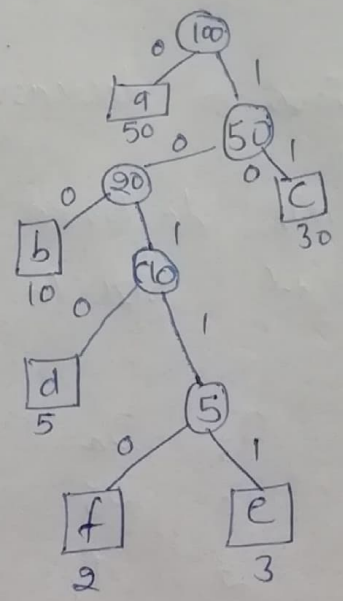
a = 50	e = 3
b = 10	
c = 30	f = 2
d = 5	

n = 100 characters

ASCII

0-127

7 × 100 = 700 bits



a = 0 = 50 × 1 = 50

b = 100 10 × 3 = 30

c = 11 = 30 × 2 = 60

d = 1010 = 5 × 4 = 20

e = 10111 3 × 5 = 15

f = 1011 4 × 5 = 10

185 bits

Average required to represent each character = $\frac{185}{100}$

= 1.85 bits / character

**

Q:- Consider the following message :

aa bbbba bbb ccc ddd eee ccc eee 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

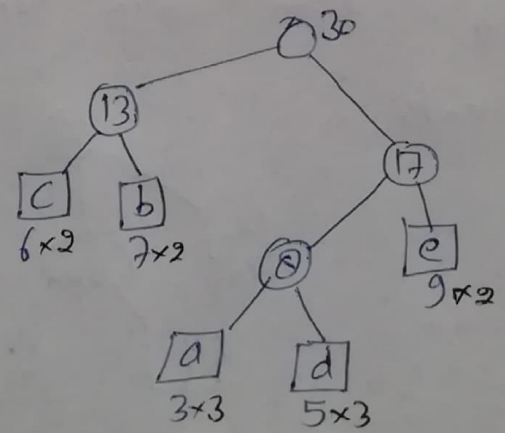
c = 6

d = 5

e = 9

13, 17

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



Job Sequencing (Greedy Technique) :-

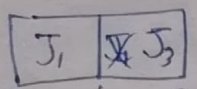
Algo:-

1) arrange all Jobs in decreasing order of profit. ($n \log n$)

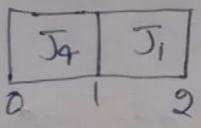
2) for each Job (mi), do linear search to find particular slot in array of size (n), where n = maximum deadline
m = total Jobs.

Q:

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



50 + 10 = 60



Max. 25 + 50 = 75 ✓ By

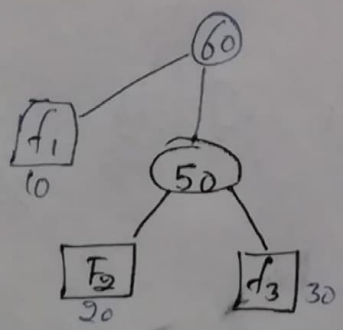
Optimal merge pattern :-

Algorithm:-

1- Create a min Heap with 'n' files

2- At each level remove two minimum

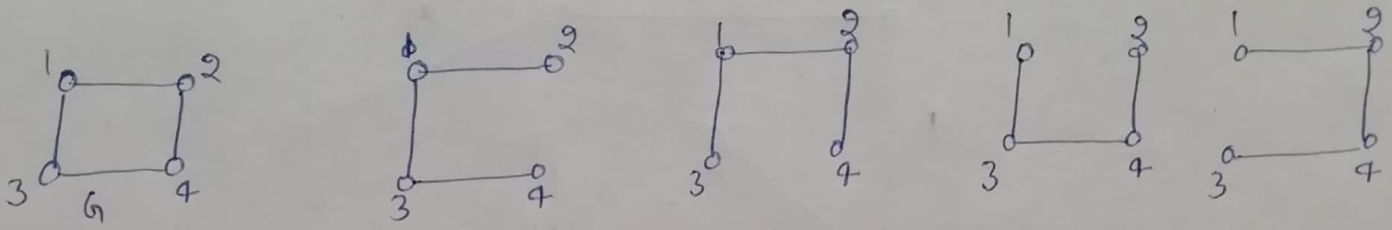
3- 'a' and 'b' from min Heap and put a+b again into Heap.



$\log n + \log n$.

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 :-

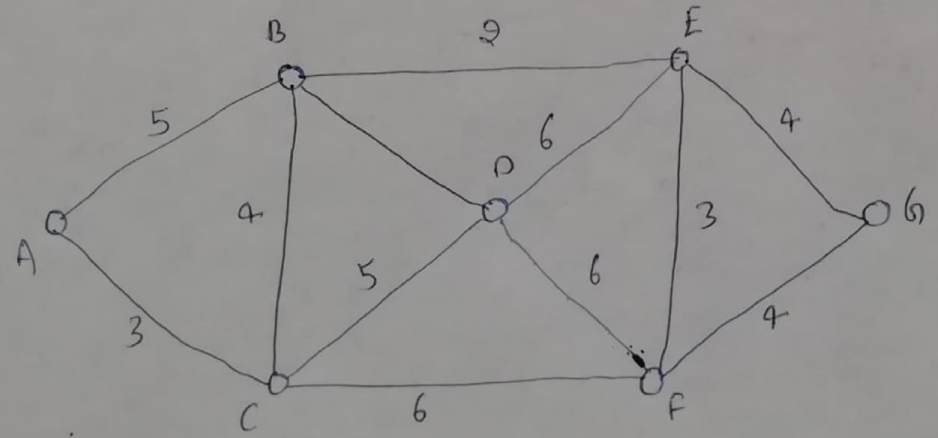
(1) construct min Heap with 'e' edge

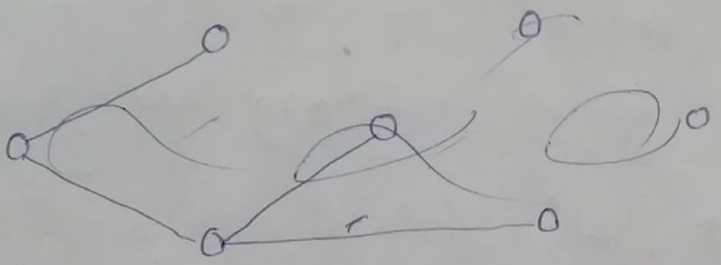
(2) Take one by one edge and add in spanning tree. (cycle should not be created).

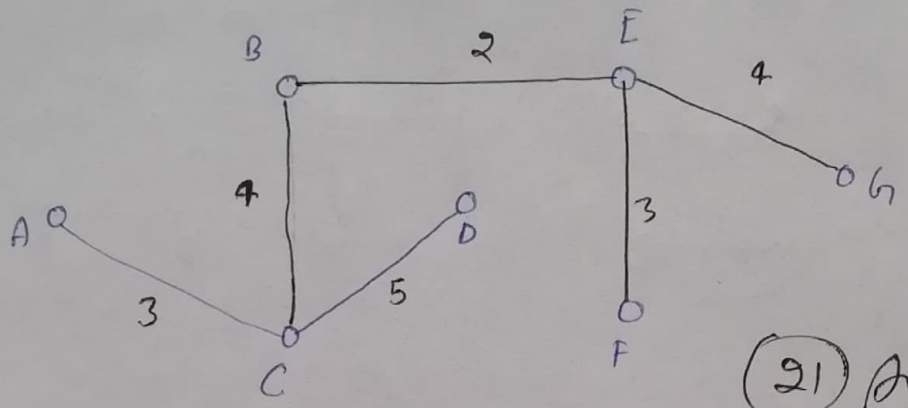
Best case (n-1) edge

Worst case 'e' edge.

(n-1) log e.





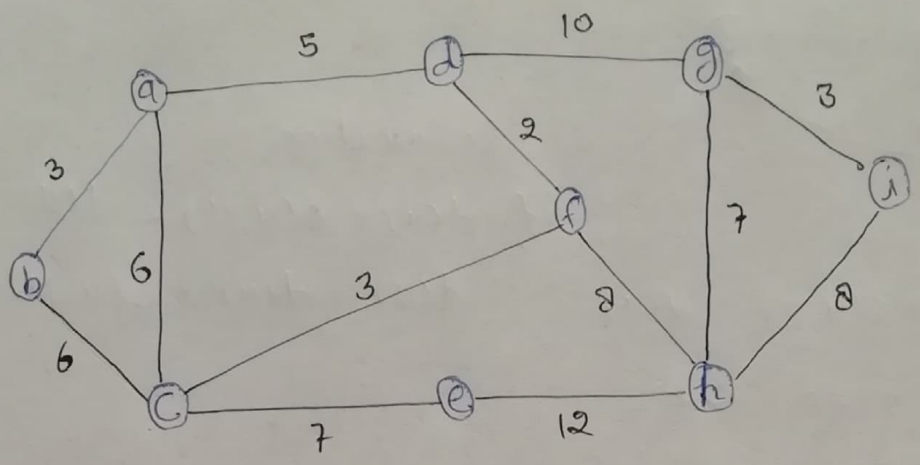


21 Ans

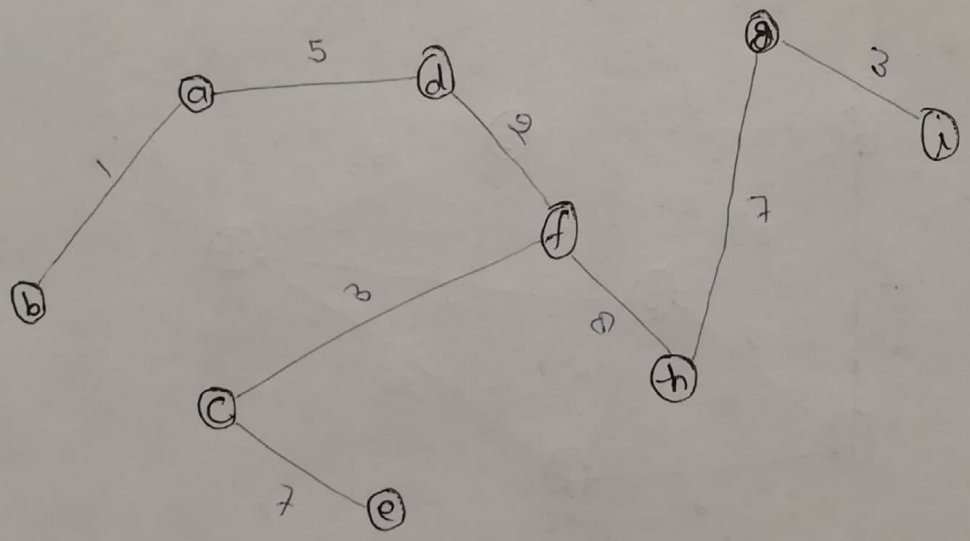
~~Prims~~

Prims Algo :-

Min cost spanning tree.

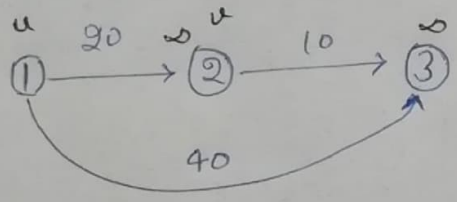


Sol:-



~~Digistra~~

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



Relaxation

if $d(u) + c(u,v) < d(v)$

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

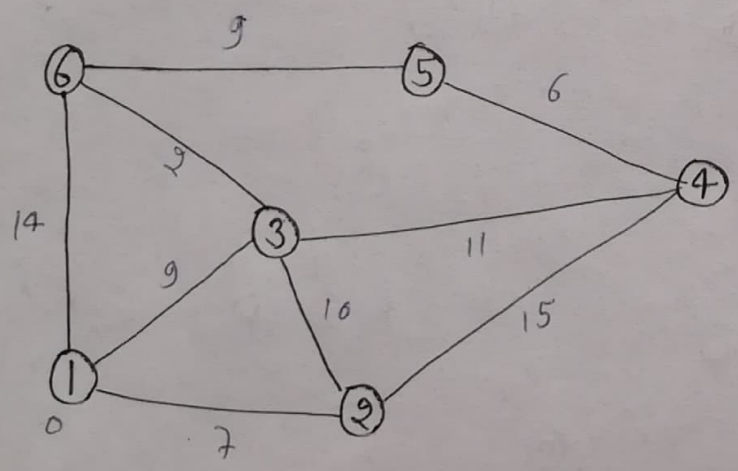
$d(u) + c(u,v)$ $d(v)$

$0 + 20 < \infty$

$0 + 40 < \infty$

$40 < \infty$

Q:-



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

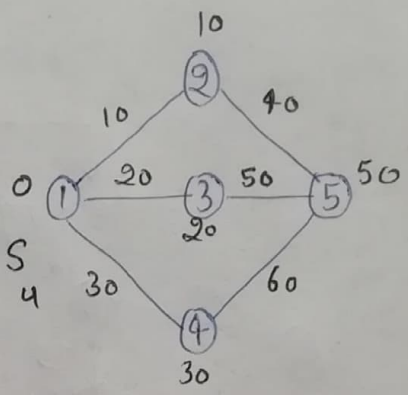
Dijkstra's Algorithm Analysis :-

Dijkstras (Graph, Source)

Create vertex set Q

for each vertex v in graph

dist [v] = ∞	O(V)
add v to Q	Build Heap
dist [source] = 0	O(V)



while Q is not empty

u = Extract-min [Q]

for each neighbour v of u

Relax (u, v)

time complexity

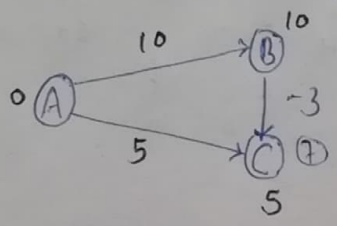
$O(V) + O(V) + V \log V + E \cdot \log V$

$= O(E \log V)$

Why does Dijkstra fail on Negative Weights ??

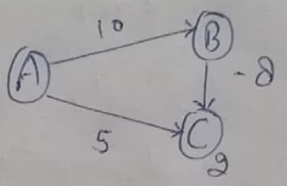
-ve

working



A	B	C
0	∞	∞
A,C	(10)	(5)
A,C,B		(5)

not working

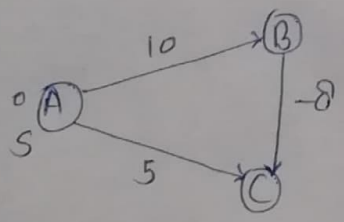


A	B	C
0	∞	∞
	10	(5) 2 ✗

A, C

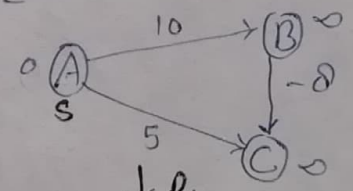
A, C, B

Bellman Ford Algorithm :-

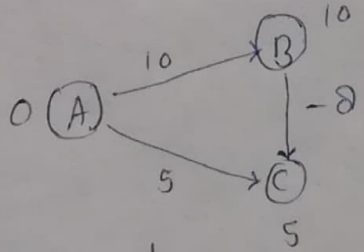


A	B	C
0	∞	∞
AC	10	(5)
ABC	(10)	(5)

3-1 = 2 time relax

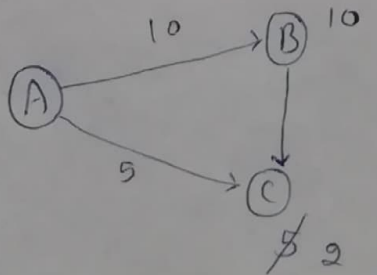


↓ R_1

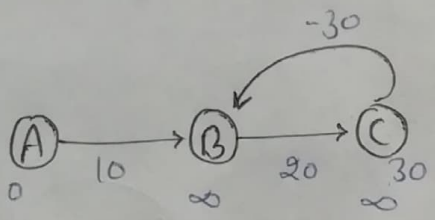


↓

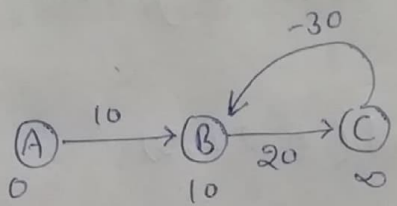
R_2

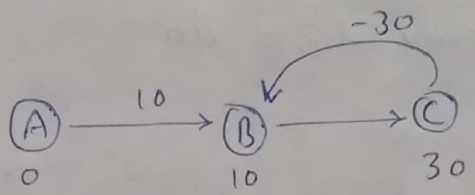


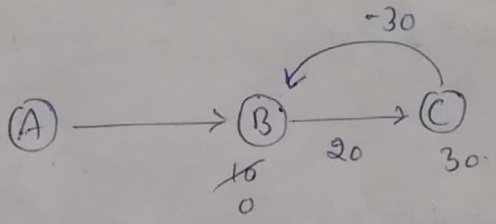
Q!-
→



v-1 +1 time





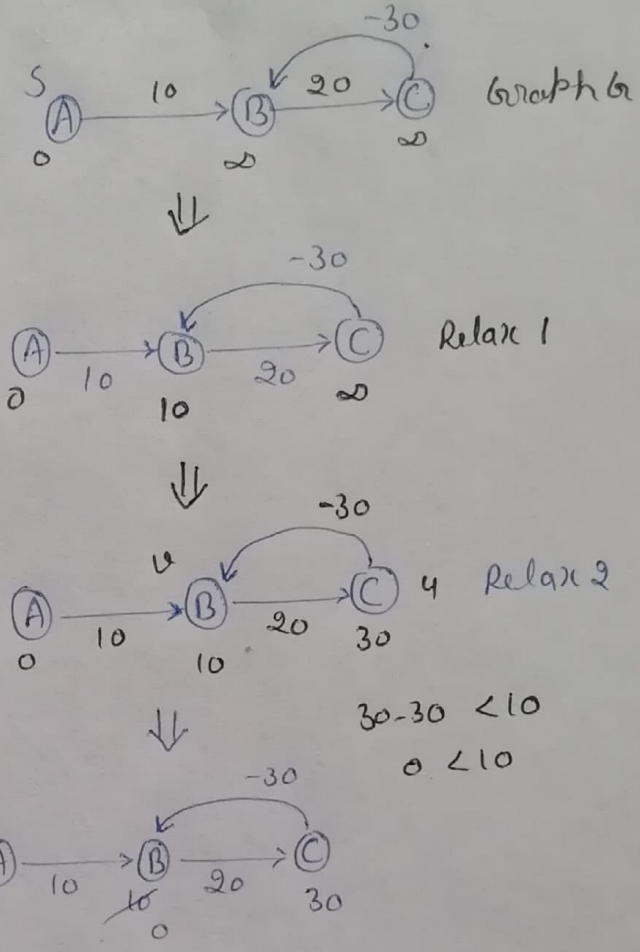


negative edge then stop.

Bellman Ford Pseudo Code :-

Bellman Ford Algorithm

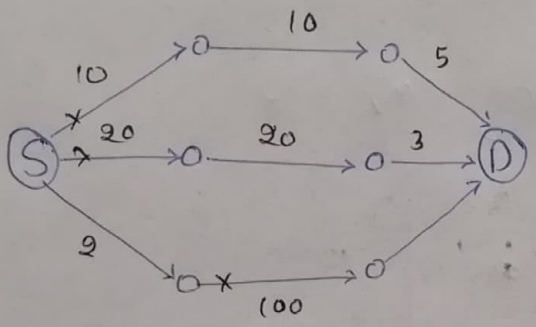
```
Bellman Ford ( G, V, E, S)
Step 1
for each vertex v ∈ G do
    dist [v] = ∞
    dist[ source ] = 0
Step 2
  for i=1 to |V| - 1
     for each edge (u v) ∈ G
        Relax (u, v, w)
Step 3
     for each edge (u, v) ∈ G
       if ( dist (u) + w(u, v) < dist [v] )
        return " Graph contain Negative
              weight cycle"
     return distance.
```



Dynamic Programming :-

It devide the problem into series of overlapping sub-problems.

Two features :- 1) optimal Substructure
2) overlapping Subproblems.



n =	0	1	2	3	4	5		
f(n) =	0	1	1	2	3	5	8	13

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

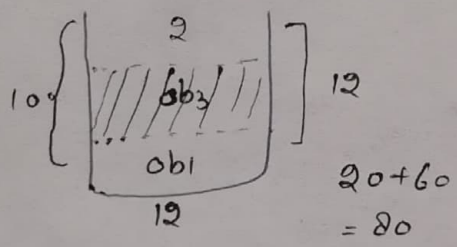
$$\begin{cases} 1 & n == 1 \\ 0 & n == 0 \end{cases}$$

0/1 Knapsack failed using Greedy approach :-

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

Object	ob_1	ob_2	ob_3
weight	2	4	8
profit	20	25	60

Knapsack capacity (M) = 12



1	2	3	4	5	6	...	n
↓	↓	↓					
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	-	85 ✓ Ans
1	0	0	-	20
1	0	1	-	80
1	1	0	-	45
1	1	1	-	NP

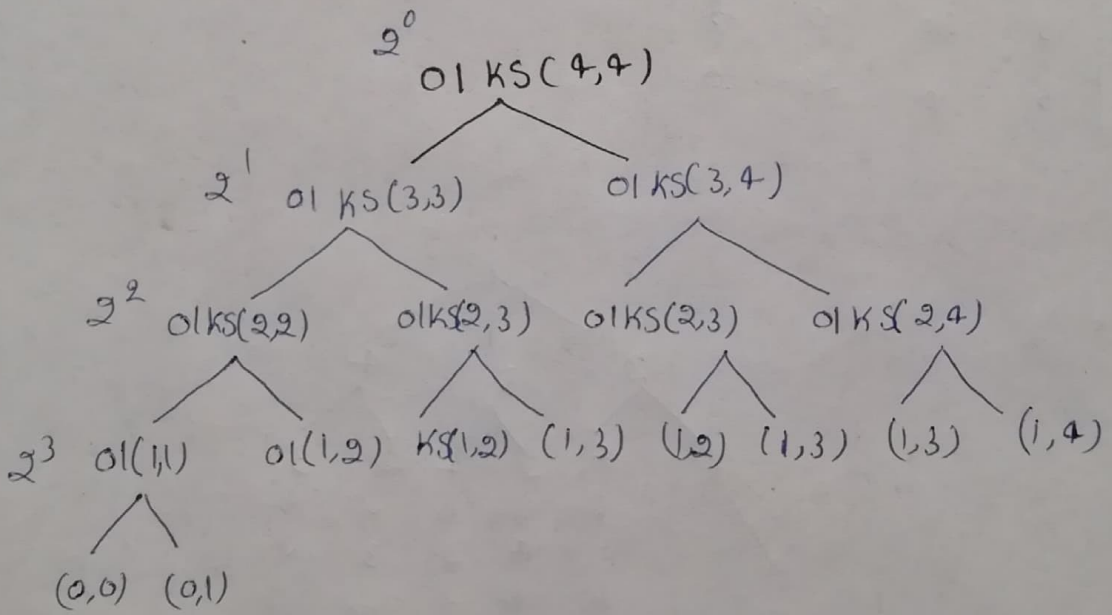
0/1 Knapsack Recursive eqn. :-

1	2	3	4	5	----	n
2 ×	2 ×	2 ×	2 ×	2	----	2	2^n

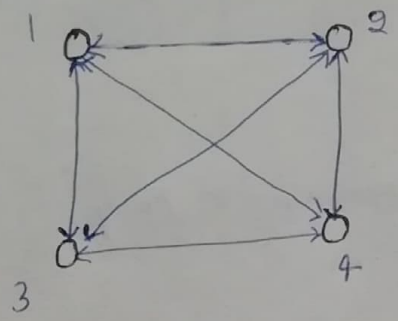
0/1 Knapsack (n, m) =

$$\begin{cases} \max \begin{cases} 0/1\,KS(n-1), (m-w_n) + p_n \\ 0/1\,KS(n-1), m \end{cases} \\ 0/1\,KS(n-1, m) \quad w_n > m \\ 0 \quad n = 0 \text{ or } m = 0 \end{cases}$$

Q) 0/1 KS (4,4)



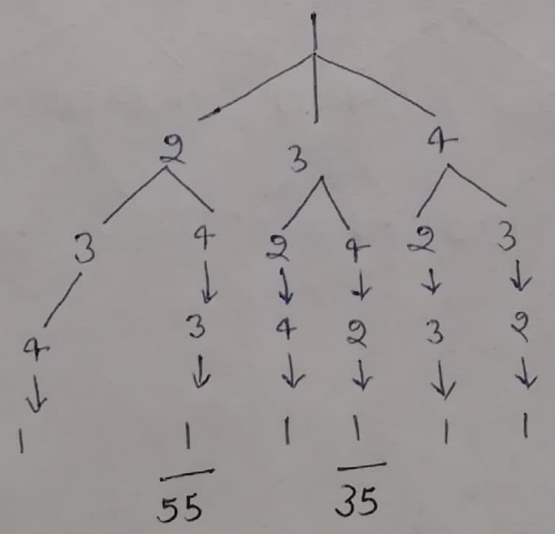
Travelling Salesman Problem :-



1 → 2 → 4 → 3 → 1

55

	1	2	3	4
1	0	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" :-

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

$$m = 50$$

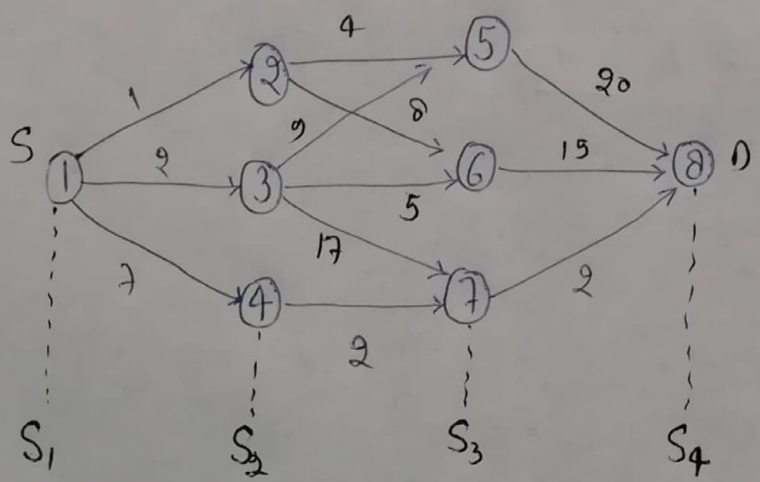
$$SoS(n,m) = \begin{cases} SoS(n-1, m) \\ SoS(n-1, m-w_n) \end{cases} \quad SoS(n-1, m)\ w_n > m$$

True $n=0$, $m=0$

False $n=0$, $m \neq 0$

True $n \neq 0$, $m=0$

Multistage Graph :-



1 — 2 — 5 — 8 (25)

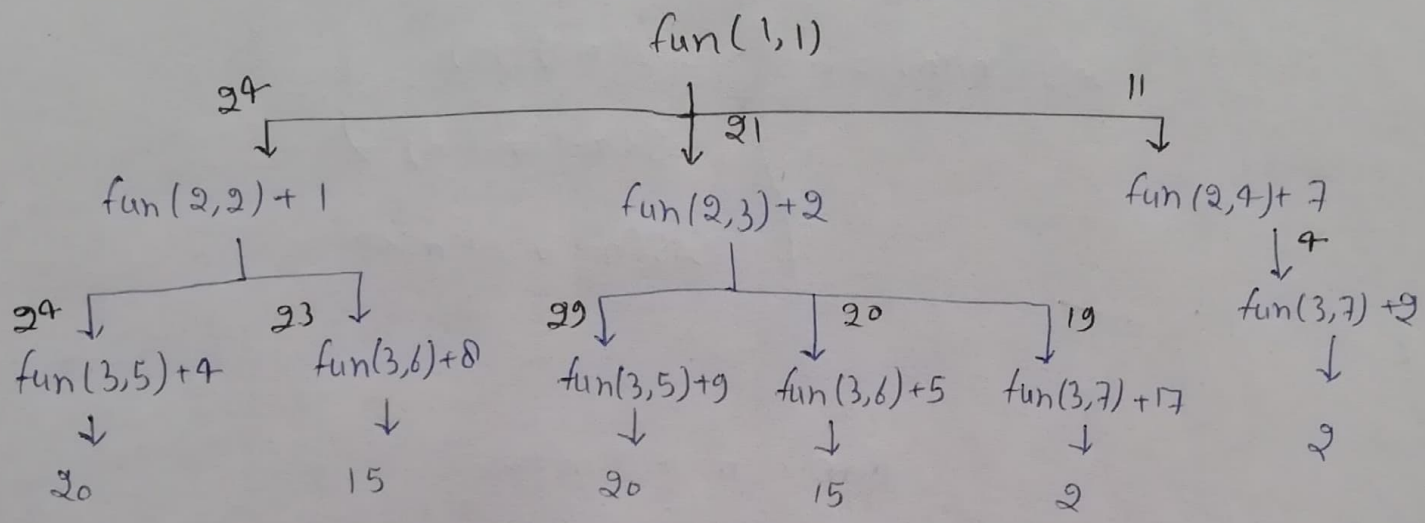
1 9 20

fun (S_i, V_j)

fun (2, 3)

$$\min \begin{cases} fun(S_{i+1}, K) + C(V_J, K) \\ \boxed{C(V_J, D) \quad S_J = S_i - 1} \end{cases}$$

(11) Ans

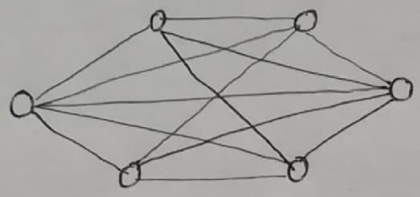


time complexity

$O(V + E)$

$O(E)$

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



$VE \log V$

$V.E$

$V.V.E$!

$V^2 E$

$V^2 . V^2$

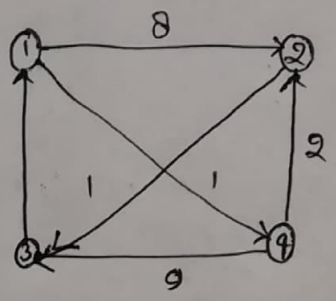
V^4

$VE \log V$

$V.V^2 \log V$

$V^3 \log V$ dijkstra

Floyd Warshall working with Ex.:-



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

$$D' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & (12) & 0 & (5) \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

$$D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & (9) & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & (3) & 0 \end{bmatrix} \end{array}$$

$D^3 =$

	1	2	3	4
1	0	8	9	1
2	(5)	0	1	(6)
3	4	12	0	5
4	(7)	2	3	0

$D^4 =$

	1	2	3	4	
1	0	(3)	(4)	1	n^2
2	5	0	1	6	
3	4	(7)	0	5	$n^2 \times n$
4	7	2	3	0	

Time complexity = $n^2 \times n$

$O(n^3)$

Space complexity = $n^2 \times 2$

$O(n^2)$

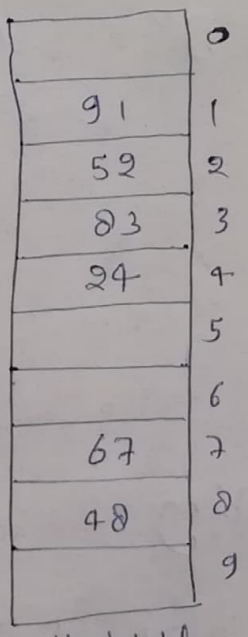
Hashing (Storing and Retrieving Data in O(1) time) :-

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

K mod 10

24 mod 10 = 4

∴ Hash value = 4



Collision in Hashing :-

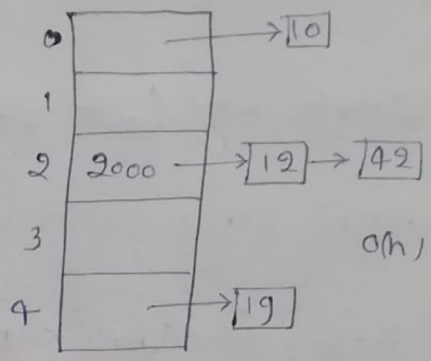
Collision Resolusion Techniques

- Chaining (open Hashing)
  - h (value) = 0
  - 0 + 1² mod 6 = 1
  - 0 + 2²
  - 4 mod 6 = 4
- $R + i^2$ mode
  - 30 mod 6 = 0
- open Addressing (closed Hashing)
  - → Linear probing
  - → Quodaratic probing
  - → Double Hashing

Chaining :- (Open Hashing)

Keys : 42, 19, 10, 12

K mod - 5



O(n)

Advantages :- Deletion is easy
O(1) Insert

Disadvantages :- O(n)
Extra space लेता है।

Load factor :-

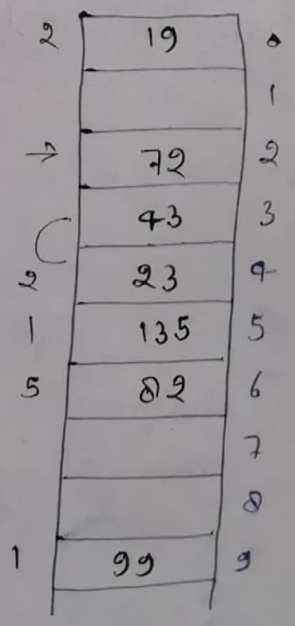
$$\alpha = \frac{\text{elements / key}}{\text{slots}} \qquad \frac{4}{5}$$

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 ε O(n)
Deletion Difficult
Primary clustering
Secondry clustering

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?

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

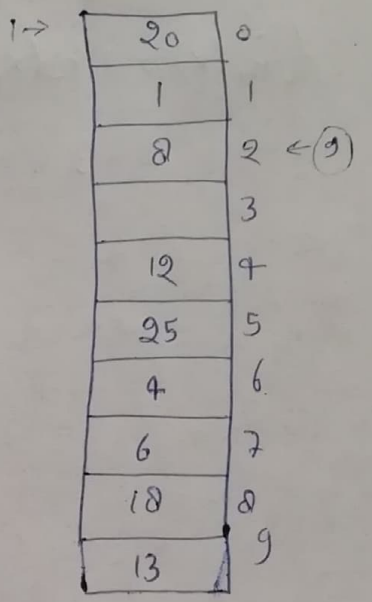
$12^2 \bmod 10$ $\quad$ $1^2 \bmod 10$

$144 \bmod 10$ $\quad$ $3^2 \bmod 10$

$9 \bmod 10$

$8^2 \bmod 10$

$64 \bmod 10$



Probe value of $8 \Rightarrow 9$

maximum probe value of $9 \rightarrow 8$

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

$6+1^2 = 7$

$6+2^2 = 10$

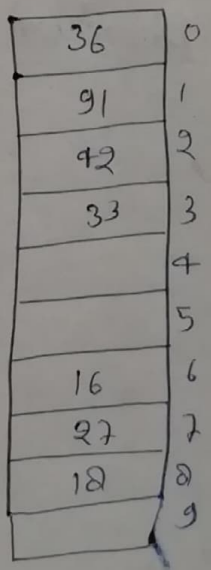
$2+1^2 = 3$

$2+2^2 = 6$

$2+3^2 = 11$

$2+4^2 = 18$

$2+5^2 = 27$



Advantage :- No extra space

Disadvantage :- Search $O(n)$

* Secondry clustering
* No guarentee of finding slot.

Double Hashing :-

$h_1(k) = k \bmod 11$

$h_2(k) = 8 - (k \bmod 8)$

$(h_1(k) + i\,h_2(k)) \bmod 11$

Keys: 20, 34, 45, 70, 56

$f(k) + i$

$f(k) + i^2$

$1 + 3 = 4$

56 → 1

$8 - (56 \bmod 8)$

$8 - (45 \bmod 8)$

$8 - 5$

3

$8 - (70 \bmod 8)$

6

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

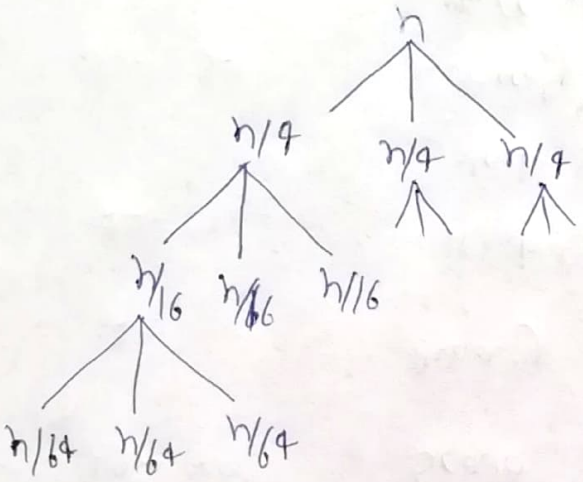
Advantage :- No extra space

No Primary Clustering

No Secondry " "

Disadvantage :- $O(n)$

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



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

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

$$1 + x + x^2 + x^3 + \cdots$$

$$cn^2\left[\frac{1}{1-\frac{3}{16}}\right]^{1/1-x}$$

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