Multiplication of large integer numbers
(Karatsuba's Algorithm)

Page No. : 1
Date : / /

$$23 * 40 = (2\times10^1 + 3\times10^0) * (4\times10^1 + 0\times10^0)$$
$$= (2\times1)10^2 + (2*4 + 3*1)10^1 + (3*4)10^0$$
$$= 2\times10^2 + 11\times10^1 + 12\times10^0$$

$$\begin{cases} T(n) = 1 & \text{if } n=1 \\ T(n) = 4T(n/2) + kn & \text{if } n>1 \end{cases}$$

$$= \Theta(n^2)$$

$$c = a*b$$
$$= c_2 10^2 + c_1 10^1 + c_0$$

here, $c_2 = a_1 \times b_1$, $c_0 = a_0 \times b_0$

Now,

$$(a+b) * (c+d)$$
$$= [a\times c + b\times d + (a\times d + b\times c)]$$

$$c = a*b$$
$$= (a_1 10^{n/2} + a_0)(b_1 10^{n/2} + b_0)$$
$$= (a_1\times b_1)10^n + (a_1\times b_0 + a_0\times b_1)10^{n/2} + (a_0\times b_0)$$

$$= c_2 10^n + c_1 10^{n/2} + c_0$$

where $c_2 = a_1 \times b_1$ = Product of their first halves
$c_0 = a_0 b_0$ = product of their 2nd "
$c_1 = (a_1 + a_0) \times (b_1 + b_0) - (c_2 + c_0)$

$$\begin{cases} T(n) = k & \text{if } n = 1 \\ T(n) = 3T\left(\frac{n}{2}\right) + kn & \text{if } n > 1 \end{cases}$$

$$T(n) = O\left(n^{\log_2 3}\right)$$
$$= O\left(n^{1.59}\right) < O(n^2)$$

Q. $a = 2345$, $b = 6137$, multiply $a * b$.

$a_1\ a_0$ $\quad$ $b_1\ b_0$

$a = 23 \mid 45$, $\quad b = 61 \mid 37$

$$c_2 = a_1 \times b_1$$
$$= 23 \times 61 \qquad = 2|3 \times 6|1$$

$$c_2 = a_1 \times b_1 = 2 \times 6 = 12$$
$$(c_0) = 3 \times 1 = 3$$
$$c_1 = 5 \times 7 - (12 + 3) = 35 - 15 = 20$$

$$c = 12 \times 10^2 + 20 \times 10^1 + 3$$
$$= 1203$$

$$c_0 = a_0 \times b_0$$
$$= 45 \times 37$$

~~45 | 37~~

$a_1\ a_0$ $\quad$ $b_1\ b_0$

45×37

$C_2 = 4 \times 3 = 12$

$C_0 = 5 \times 7 = 35$

$C_1 = 9 \times 10 - (12 + 35) = 90 - 47 = 43$

$C_0 = 12 \times 10^2 + 43 \times 10 + 35$

$= 1200 + 430 + 35$

$= 1665$

$C_1 = (a_1 + a_0) \times (b_1 + b_0) - (C_0 + C_2)$

$= (23 + 45) \times (61 + 37) - (1403 + 1665)$

$= 68 \times 98 - 3068$

$a_1\ a_0 \quad b_1\ b_0$

$6\ 8 \times 9\ 8$

$C_2 = 6 \times 9 = 54$, $C_0 = 8 \times 8 = 64$

$C_1 = 14 \times 17 - (54 + 64)$

$= 238 - 118 = 120$

$C_1 = 54 \times 10^2 + 120 \times 10^1 + 64 - 3068$

$= 5400 + 1200 + 64 - 3068$

$= 6664 - 3068 = 3596$

Ans $= C_2 \times 10^4 + C_1 \times 10^2 + C_0$

$= 1403 \times 10^4 + C_1 \times 10^2 + C_0$

$= 14391265$

Analysis of Binary search Algorithm

* Algorithm bsearch (a, n)

```
{
    low = 0, high = n-1
    while (low <= high) do
        mid = (low + high) / 2;
        if (x = a[mid])
            return mid;
        else if (x < a[mid])
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}
```

$$T(n) = T\left(\frac{n}{2}\right) + 1 \quad \text{for } n > 1$$

$$T(1) = 1$$

$$T(n) = \Theta(\log n)$$

Decrease & conquer

Page No. : 5.
Date : / /

The Variations of Decrease & Conquer

1. Decrease by a constant (usually by 1)
   - Insertion Sort
   - DFS & BFS
   - Topological Sorting
   - Generating permutations and subsets

2. Decrease by a constant factor (usually by half)
   - Disection matrix

3. Variable size decrease
   - Euclid's algorithm
   - Computing a median
   - Searching & Sorting in a BST

★ Insertion Sort

```
Algorithm insertsort (a,n)
{   for i=1 to n-1 do
        v = a[i]
        j = i-1;
    while (j>=0 and a[j]>v) do
    {
        a[j+1] = a[j];
        j = j-1;
    a[j+1] = v;
    }.
```

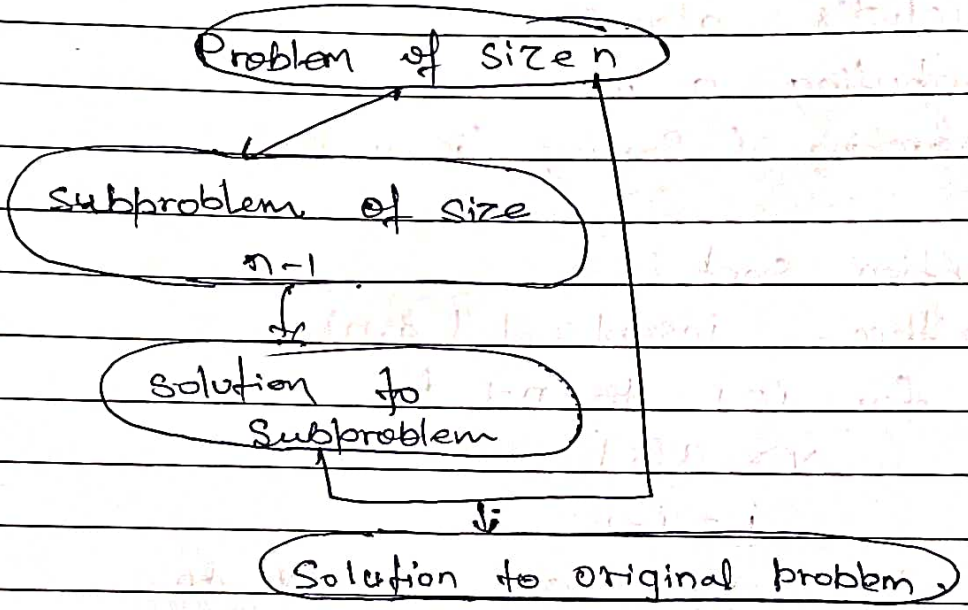
$$C_{worst}(n) = \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} 1$$

$$= \sum_{i=1}^{n-1+1} i = O(n^2)$$

$$C_{best}(n) = \sum_{i=1}^{n-1} = n-1+1-1$$

$$= n-1 = O(n)$$

(a) Decrease by constant (usually by 1)



* Compute a^n

$$a^n = (a^{n/2})^2$$

$$a^n = \begin{cases} (a^{n/2})^2 & \text{if } n \text{ is even \& +ve} \\ (a^{\frac{n-1}{2}})^2 & \text{if } n \text{ is odd \& +ve} \\ a & \text{if } n = 1 \end{cases}$$

Time Complexity = $O(\log n)$

(5) Decrease by a constant factor (usually by half)
Divide & conquer

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n > 1 \\ a & \text{if } n = 1 \end{cases}$$

* depth first search (DFS)

$G = (V, E)$

Algorithm of dfs(v)

Mark each vertex in V with 0 as 'unvisited'.

count = 0

for each vertex v in V do

if v is marked with 0

dfs(v)

dfs(v)

count = count + 1

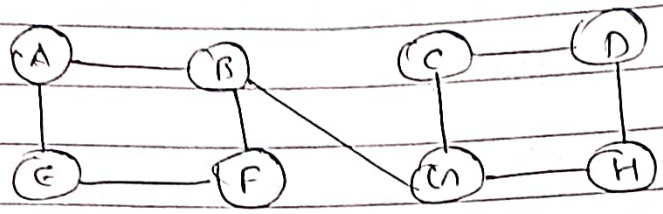
mark v with count

for each vertex is in v adjacent to v do

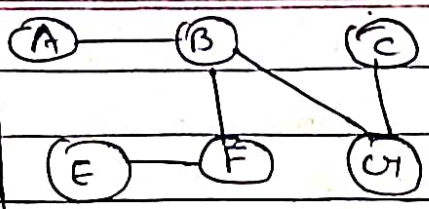
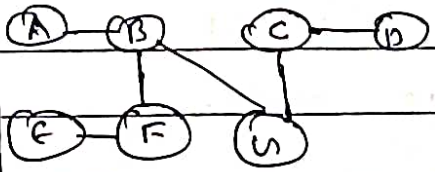
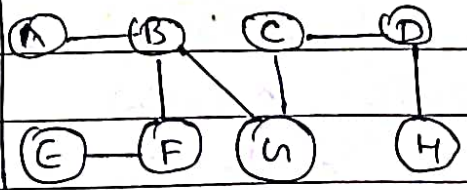
if it is marked with 0

dfs(w)

Q. Start at vertex A traverse the following graph using dfs traverse method.



Step	Graph	Remarks	Stack
1	A	insert A into stack A(1)	A
2	A — B	insert B into stack A(1), B(2)	B, A
3	A — B, B — F	insert F into stack A(1), B(2), F(3)	F, B, A
4	A — B, B — F, E — F	Push E into stack A(1), B(2), F(3), E(4)	E, F, B, A
5	No unvisited adjacent vertex, backtracking	Pop E from Stack E(4,1), F(3), A(1)	F, B, A
6	No unvisited adjacent vertex, backtracking	Pop F from stack E(4,1), F(3,2), B(2), A(1)	B, A
7	A — B, E — F, G	Push G into stack G(5), E(4,1), F(3,2) B(2), A(1)	G, B, A

8		Push C into stack C(6), G(5), E(4,1), F(3,2), B(2), A(1)	C G B A
9		Push D into stack D(7), C(6), G(5), E(4,1) F(3,2), B(2), A(1)	D C G B A
10		Push H into stack H(8) - -	H D C G B A
11	No unvisited adjacent nodes, backtracking	Pop H from stack H(8,3) - -	D C G B A
12	11	Pop D from stack D(7,4) - -	C G B A
13	11	Pop C from stack C(6,5)	G B A
14	11	Pop G from stack G(5,6)	B A
15	11	pop B from stack B(2,7)	A
16	11	pop A from stack A(1,8)	
17	Stack is empty		

* A(1,8), B(2,7), C(6,5), D(7,4), E(4,1), F(3,2), G(5,6), H(8,3)

Topological Sorting

* Sorting method to list the vertices of the graph in such an order that for every edge in the graph, the vertex where the edge starts is listed before the vertex where the edge ends.

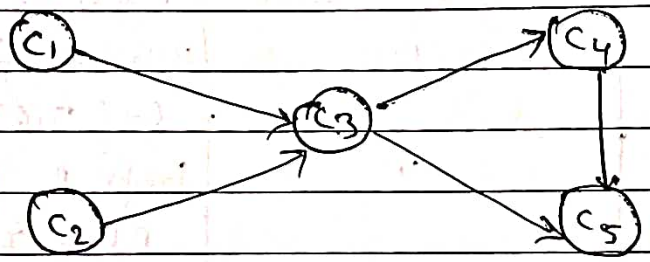
→ Applicable only when Graph must be DAG (Directed Acyclic Graph)

- If there is a cycle, topological sorting is not possible.

Methods

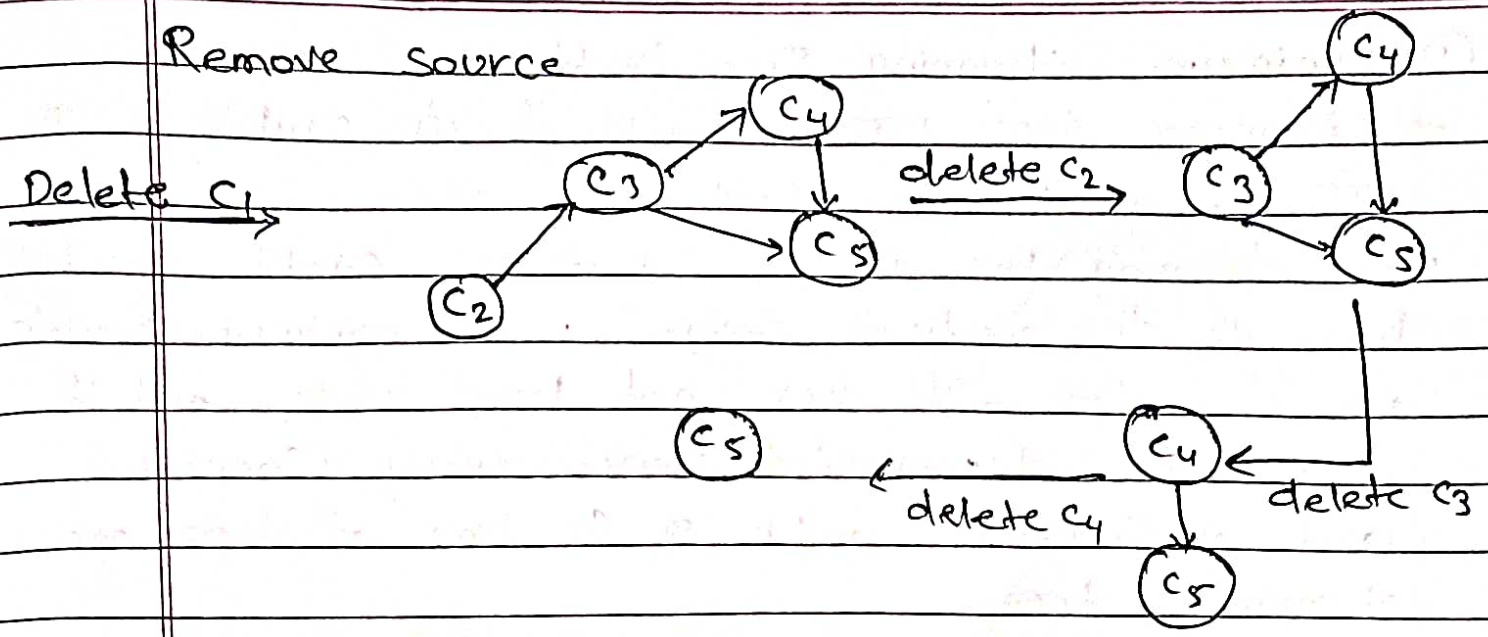
1. DFS method
2. Source Removal Method

* Apply source removal method to solve topological sorting problem for following graph.



Identify source (Indegree = 0)

* How to generate subset of a given set
* Euclid



The sequence is
→ c_1, c_2, c_3, c_4, c_5

Greedy Technique

* Greedy technique is a general algo. design strategy built on the following elements
- Configuration : different choices, values to find
- Objective fun : Some configuration to be either maximized or minimized.
- Greedy algorithm construct a sol^n to an optimization problem step by step through a sequence of choices i.e. is. terrible, logically optimal, irrevocable
- Applications of Greedy Strategy
- change making (coin change problem)
- MST
- Single Source Shortest path (Dijkstra's Algo)
- Huffman Codes
- Approximation

* Travelling Salesman Problem, Fractional Knapsack problem

Greedy Techniques

① Minimum Spanning Tree problem

* Spanning tree एक subgraph है जो graph G के सारे vertex contain करता है।

- A spanning tree is a subset of graph G, which has all the vertices covered with minimum possible no. of edges. It does not have cycles and it cannot be disconnected. Hence, every connected and undirected graph G has at least one spanning tree.

- minimum spanning tree (MST) is a subset of the edges of a connected, edge weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

② Prim's Algorithm

Algorithm Prim(G)

$V_T \leftarrow \{v_0\}$

$E_T \leftarrow \phi$ (NULL)

For (i=1 to |V|-1)

do

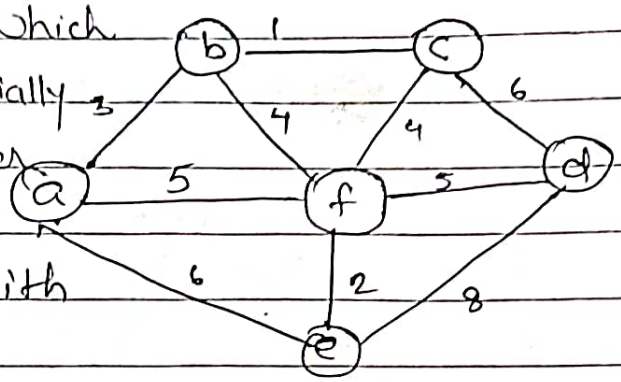
Find a minimum weight edge $e^* = (v^*, u^*)$ among all the edges (v,u) such that $v \in V_T$ and $u \in V - V_T$.

$V_T \leftarrow V_T \cup \{u^*\}$

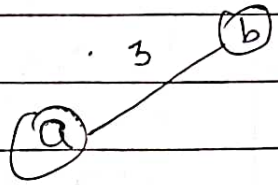
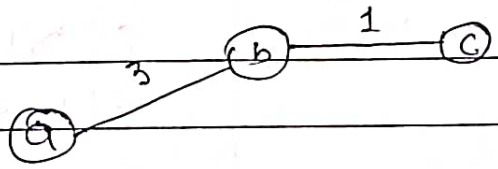
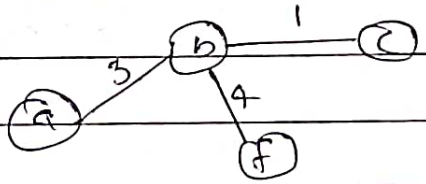
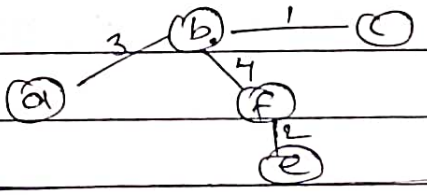
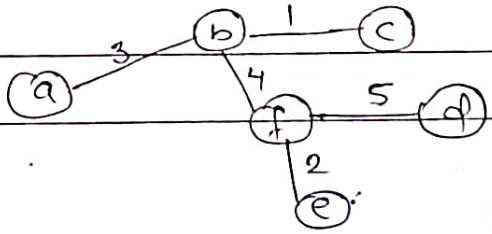
$E_T \leftarrow E_T \cup \{e^*\}$

return E_T.

Q. Apply Prim's algorithm to find a MST for the following graph.



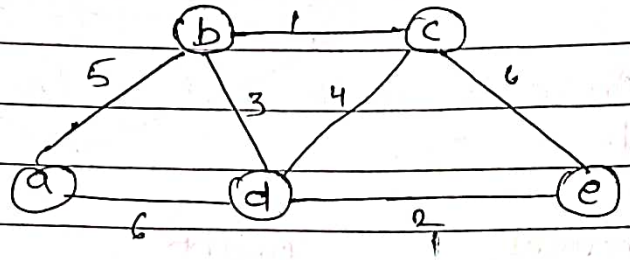
- Fringe edge → An edge which has one vertex is in partially constructed tree T_i and the other is not.
- Unseen edge → An edge with both vertices not in T_i.

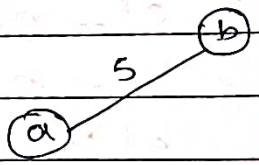
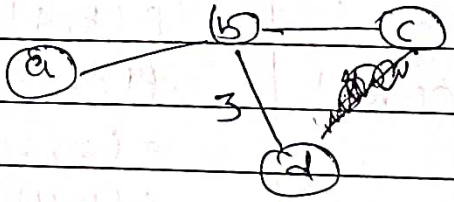
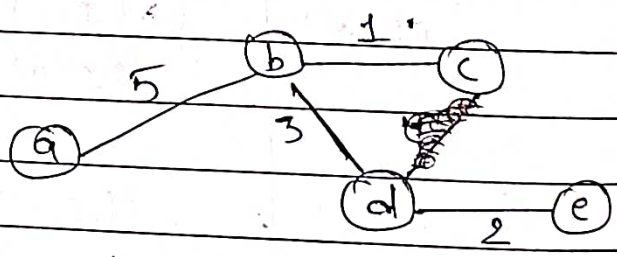
Tree Vertices	Remaining vertices	Graph
a(–,–)	b(a, 3) c(–, ∞) d(–, ∞) e(a, 6) f(a, 5)	
b(a,3)	c(b,1) d(–, ∞) e(a,6) f(b,4)	
c(b,1)	d(c,6) e(a,6) f(b,4)	
f(b,4)	d(f,5) e(f,2)	
f(f,2)	d(f,5)	

Sum of the weight of the edges
= 3+1+4+5+2 = 15

Weight of MST = 15

Q.



Tree Vertices	Remaining Vertices	Graph
a(–,–)	b(a,5) ✓ c(–,∞) d(a,6) e(–,∞)	
b(a,5)	c(b,1) ✓ d(b,3) e(–,∞)	
c(b,1)	d(b,3) ✓ e(c,6)	
d(b,3)	e(d,2)	

Weight = 5+1+3+2
= 11

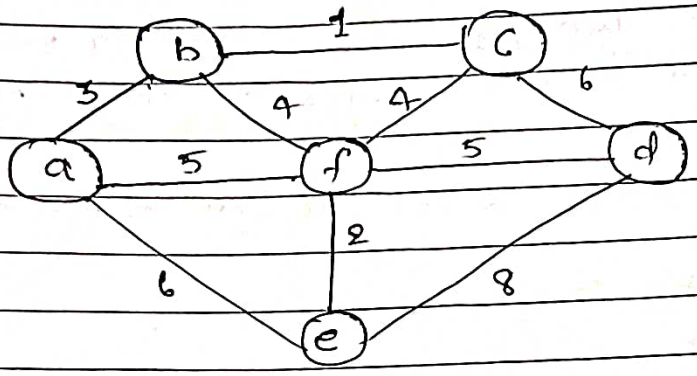
(b) Kruskal's Algo

- edges के weights को ascending order में सजा लें।
- ध्यान रखना तब है जब edges की संख्या vertices की संख्या से एक अधिक हो।
- जब भी edge draw करें तो ध्यान रखें कि cycle ना बनें।

Algorithm :-

// Input : A weighted connected graph $G = \{V, E\}$
// Output : E_T the set of edges composing a MST of G
Sort E in ascending order of the edge weight

// Initialize the set of tree edges and its size
$E_T \leftarrow \phi$
edge_counter $\leftarrow 0$
// initialize the no. of processed edges
$k \leftarrow 0$
while edge_counter $< |V| - 1$
$k \leftarrow k + 1$
if $E_T \cup \{e_{ik}\}$ is acyclic
$E_T \leftarrow E_T \cup \{e_{ik}\}$
edge_counter $\leftarrow$ edge_counter + 1
return E_T

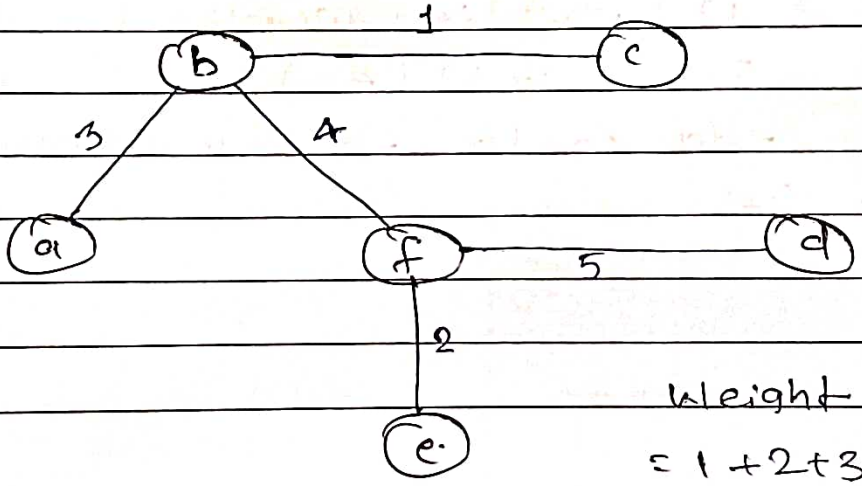


The list of edges is

Edge	ab	af	ae	bc	bf	cf	cd	df	de	ef
Weight	3	5	6	1	4	4	6	5	8	2

Sort the ~~edges~~ weight of the edges in the ascending order

Edge	bc	ef	ab	bf	cf	af	df	ae	cd	de
Weight	1 ✓	2 ✓	3 ✓	4 ✓	4 ×	5 ×	5 ✓	6	6	8



Weight of MST
$= 1+2+3+4+5 = 15$

Efficiency :- Efficiency of kruskal's algorithm is based on the time needed for sorting the edge weight of a given grt graph.
- With an efficient sorting algorithm.

Efficiency : $\Theta(E \log E)$

Conclusion :-
- Kruskal's algorithm is an "edge based algorithm".
- Prim's algorithm with a heap is faster than Kruskal's algorithm.

Prim's Algorithm Analysis

Efficiency of Prim's algorithm is based on data structure to store priority queue.

Unordered array : Efficiency : $\Theta(n^2)$

Binary heap : Efficiency : $O(m \log n)$

Min heap : Efficiency : $(n+m) \log n$

Conclusion :-
- Prim's algorithm is a vertex based algorithm
- Prim's algorithm "Needs priority queue for locating the nearest vertex" The choice of priority queue matters in prim implementation.
- Array : optimal for dense graphs
- Binary heap : better for sparse graphs
- Fibnacci heap : best in theory but not in practice

Greedy Solution for fractional Knapsack

Page No. : 19.
Date : / /

Greedy algorithm = $O(n \log n)$

Given a set of item I:

Weight	w_1	w_2	. . .	w_n
Cost	c_1	c_2	. . .	c_n

Let P be the problem of selecting items from I, with weight limit K such that the resulting cost (value) is maximum.

1. Calculate $v_i = \frac{c_i}{w_i}$ for $i = 1, 2, \ldots, n$

2. Sort the items by decreasing v_i. Let the sorted item sequence be $1, 2, \ldots, i, \ldots, n$ and the corresponding v and weight be v_i and w_i respectively.

3. Let k be the current weight limit (initially $k = K$). On each iteration we choose item i, from the head of the unselected list.

   If $k >= w_i$, we take item i, and $k = k - w_i$, then consider the next unselected item

   If $k < w_i$, we take a fraction f of item i, i.e., we only take $f = \frac{k}{w_i}$ (< 1) of item i, which weights exactly k. Then the algorithm is finished.

Q. Consider the following instance of knapsack problem, with $n = 7$, $w = 15$.

$(w_1, w_2, \ldots, w_7) = (2, 3, 5, 7, 1, 4, 1)$

$(v, v_2, \ldots, v_7) = (10, 5, 15, 7, 6, 18, 3)$

Find an optimal solution.

item	weight	value	value / weight
1	2	10	5
2	3	5	1.67
3	5	15	3
4	7	7	1
5	1	6	6
6	4	18	4.5
7	1	3	3

Sort acc^r to decreasing value/weight

item	weight	value	value/weight
5	1	6	6
1	2	10	5
6	4	18	4.5
3	5	15	3
7	1	3	3
2	3	5	1.67
4	7	7	1

Max value $= 6 + 10 + 18 + 15 + 3 + \frac{2}{3} \times 5$

$= 52 + 3.34$

$= 55.34$

Step 1:- Select item 5

$w_5 = 1$, $v_5 = 6$

now, update weight $w = w - w_5$

$= 15 - 1 = 14$

Value = 6

Step 2:- Select item 1

$w_1 = 2$, $v_1 = 10$

now, update weight $w = w - w_1$

$= 14 - 2 = 12$

Value = 6 + 10 = 16

Step 3:- Select item 6

$w_6 = 4$, $v_6 = 18$

Update weight $w = w - w_6 = 12 - 4 = 8$

Value = 16 + 18 = 34

Step 4:- Select item 3

$w_3 = 5$, $v_3 = 15$

Update weight $w = w - w_3 = 8 - 5 = 3$

Value = 34 + 15 = 49

Step 5:- Select item 7

$w_7 = 1$, $v_7 = 3$

Update weight $w = w - w_7 = 3 - 1 = 2$

Value = 49 + 3 = 52

Step 6 - Select item 2

$w_2 = 3$, $v_2 = 5$

Update weight $w = 2 - 3 < 0$

∴ Here, we can select only 2 weight

i.e. value $= \frac{2}{3} \times 5 = \frac{10}{3} = 3.34$

Total value $= 52 + 3.34$

$= 55.34$

Shortest path problem

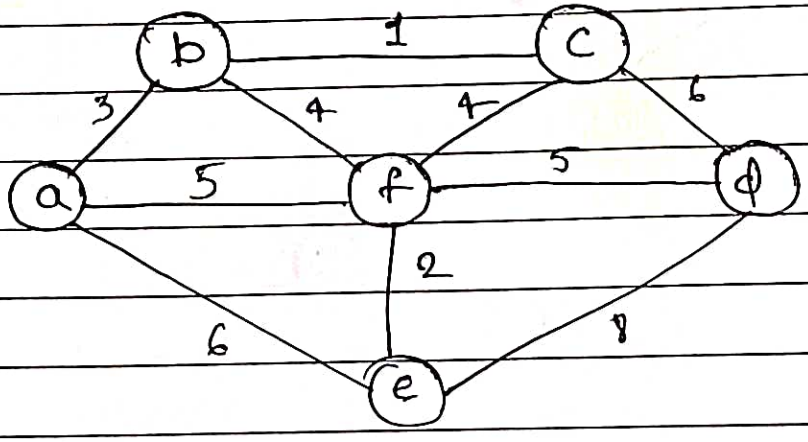
① Single source shortest path problem (Dijkstra Algorithm)
② All pair shortest path problem

Dijkstra's Algorithm

Given a weighted connected graph G(V,E) for a given vertex s, we are interested in finding shortest path from s to all other vertices.

Dijkstra's algorithm is a greedy algorithm that finds such paths when the weights are non-negative.

$$D(v) = \text{new } \{ D(v),\ D(u) + w(u,v) \}$$



Solution

Length D_v of shortest path from source (s) to vertices v and penultimate vertex P_v for every ve v in V.

Initially

$D_a = 0$, P_a = NULL

$D_b = \infty$, P_b = NULL

$D_c = \infty$, P_c = NULL

$D_d = \infty$, P_d = NULL

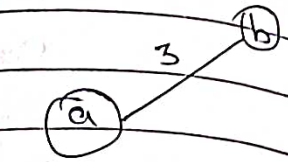
$D_e = \infty$, P_e = NULL

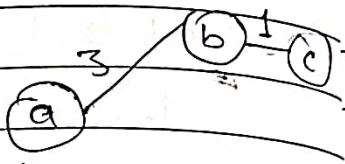
$D_f = \infty$, P_f = NULL

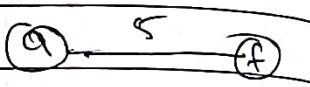
Tree Vertices	Remaining vertices	Distance & path vertex		Graph
a(-, 0)	b(a, 3)	$D_a = 0$	$P_a = a$	
	c(-, ∞)	$D_b = 3$	$P_b = [a, b]$	
	d(-, ∞)	$D_c = \infty$	P_c = null	
	e(a, 6)	$D_d = \infty$		
	f(a, 5)			

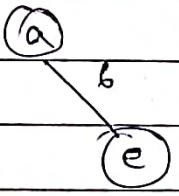
Tree Vertices	Remaining Vertices	Distance & Path Vertex	
		$D_a = 0$	$P_a = a$
	b(a,3)	$D_b = 3$	$P_b = [a,b]$
a(-,0)	c(-,∞)	$D_c = \infty$	P_c = null
	d(-,∞)	$D_d = \infty$	P_d = null
	e(a,6)	$D_e = 6$	$P_e = [a,e]$
	f(a,5)	$D_f = 5$	$P_f = [a,f]$
		$D_a = 0$	$P_a = a$
	c(b,3+1)	$D_b = 3$	$P_b = [a,b]$
	~~d(-,∞)~~	$D_c = 4$	$P_c = [a,b,c]$
b(a,3)	d(-,∞)	$D_d = \infty$	P_d = null
	e(a,6)	$D_e = 6$	$P_e = [a,e]$
	f(a,5)	$D_f = 5$	$P_f = [a,f]$
		$D_a = 0$	$P_a = a$
		$D_b = 3$	$P_b = [a,b]$
		$D_c = 4$	$P_c = [a,b,c]$
c(b,4)	d(c,4+6)	$D_d = 10$	$P_d = [a,b,c,d]$
	e(a,6)	$D_e = 6$	$P_e = [a,e]$
	f(a,5)	$D_f = 5$	$P_f = [a,f]$
		$D_a = 0$	$P_a = a$
		$D_b = 3$	$P_b = [a,b]$
		$D_c = 4$	$P_c = [a,b,c]$
f(a,5)	d(c,10)	$D_d = 10$	$P_d = [a,b,c,d]$
	e(a,6)	$D_e = 6$	$P_e = [a,e]$
		$D_f = 5$	$P_f = [a,f]$

Graph

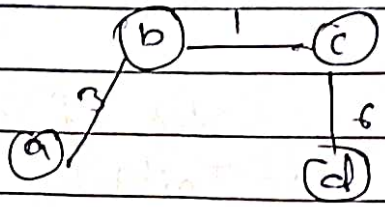








		$D_a = 0$	$P_a = a$
		$D_b = 3$	$P_b = [a,b]$
		$D_c = 4$	$P_c = [a,b,c]$
e(a,6)	d(c,10)	$D_d = 10$	$D_d = [a,b,c,d]$
		$D_e = 6$	$P_e = [a,e]$
		$D_f = 5$	$P_f = [a,f]$
d(c,10)	NIL		



Algorithm stops since no edge to scan

Conclusion

- Doesn't work with negative weights
- Applicable to both undirected and directed graphs
- Use unordered array to store the priority queue
  Efficiency = $O(n^2)$ [Array implementation]
- Use min-heap to store the priority queue
  Efficiency = $O(m \log n)$ [Adjacency list & min heap implementation]

Huffman Trees

→ invent by David. A Huffman in 1951.
→ Huffman algorithm uses a table of frequencyies of occurences to build up an optimal way of representing each character as a binary string. It achieves data compression by finding the best variable length binary encoding schemes for the symbol that occur in the file to be compressed.

Huffman codes of data compression achieves 20-90% compression.

Step for constructing huffman tree

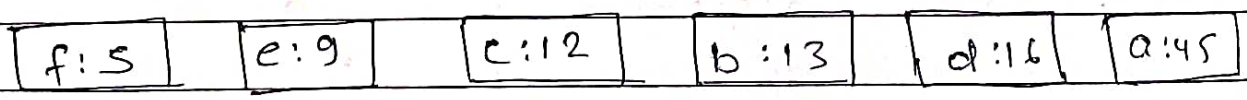
1. Initialize n one-node trees and label them with the characters of the alphabet. Record the frequency of each character in its tree's root to indicate the weight of the tree.

2. Repeat the following operation until a single tree obtains
   (i) Find two trees with smallest weight
   (ii) Make them the left and right subtree of a new tree and record the sum of their weights as the root of its new tree as its weights.

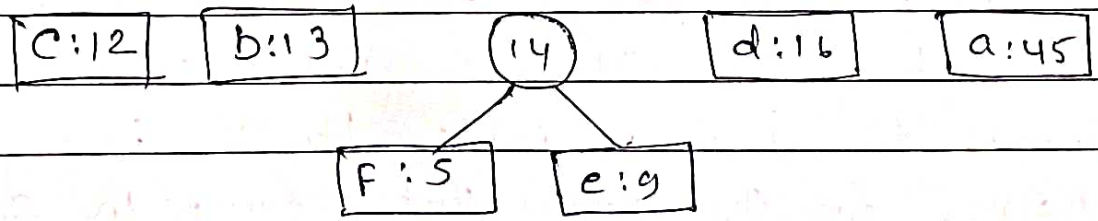
Q. Construct a huffman tree for the following data and obtain its huffman's code. (Variable length coding)

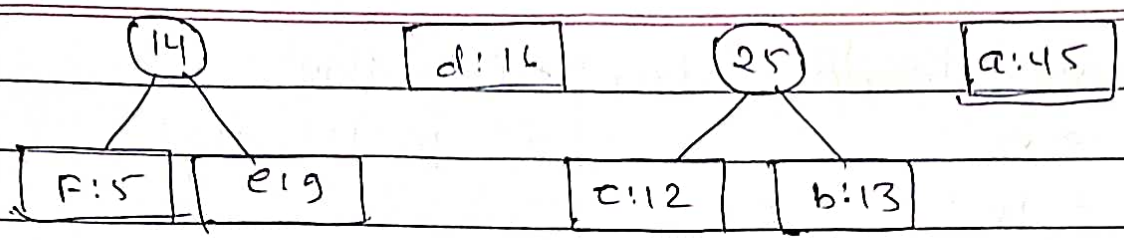
Character	a	b	c	d	e	f
Frequency of occurs or Probability	45	13	12	16	9	5

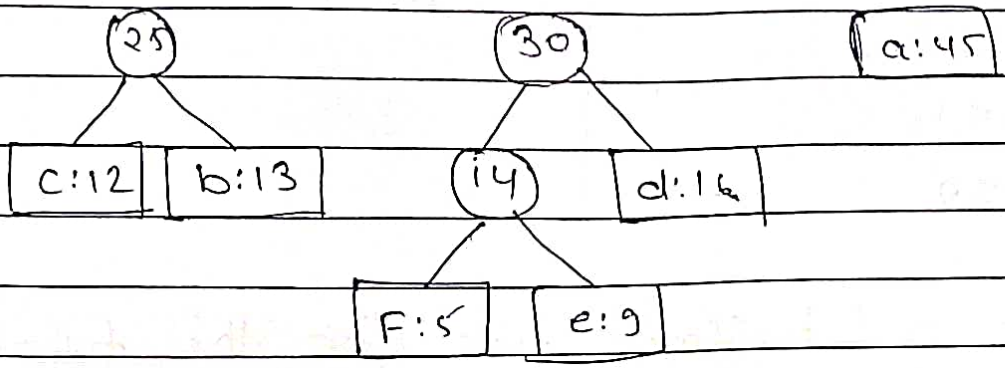
i)

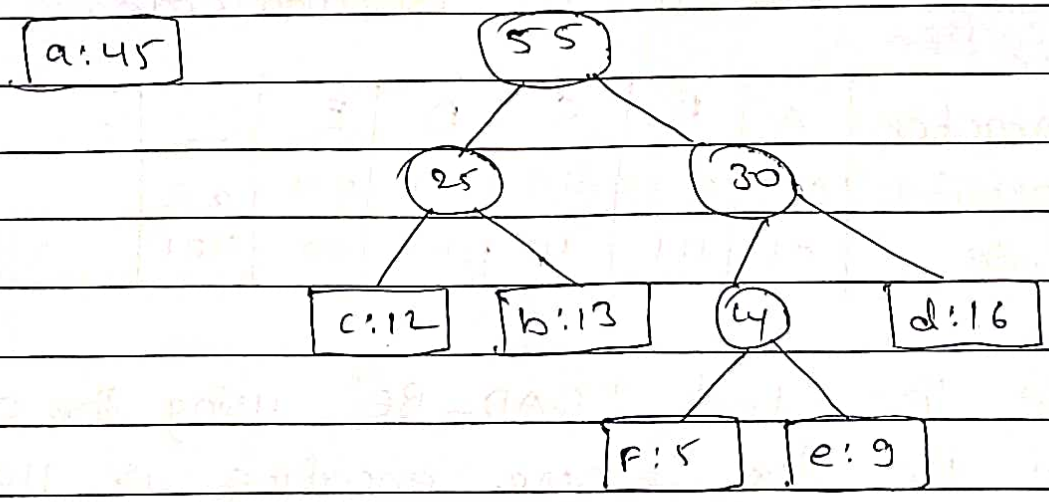


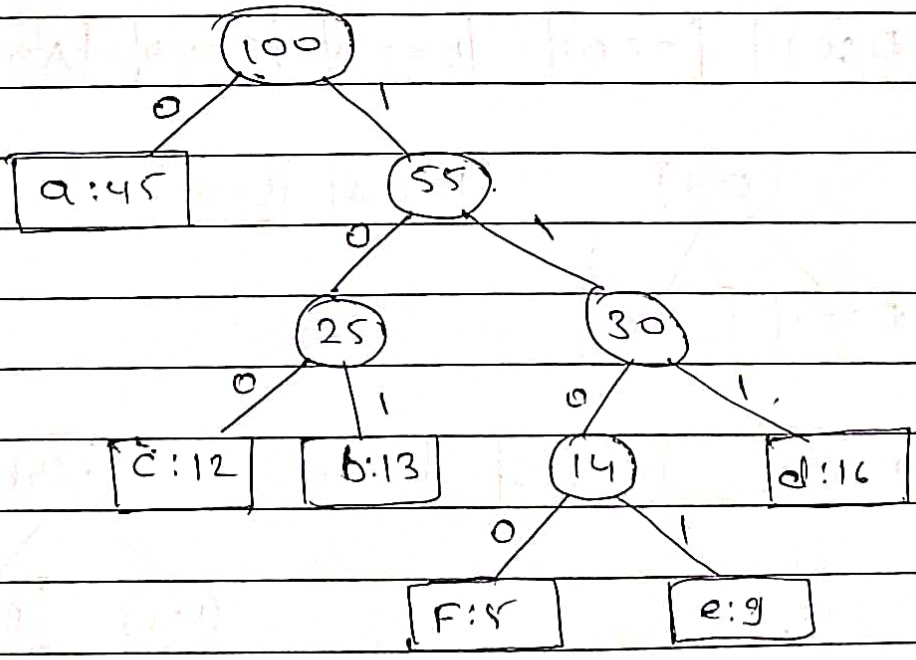
ii)











Huffman tree

Page No. : 29
Date : / /

* Variable length coding

a → 0
b → 101
c → 100
d → 111
e → 1101
f → 1100

• Decoding

(i) 100111011101
c d a e

= cdae

Q.i) Construct a huffman tree for the following data and obtain its huffman code.

Character	A	B	C	D	E	_
Probability	0.5	0.35	0.5	0.1	0.4	0.2
Code	01	111	10	1100	00	1101

ii) Encode the text "DAD_BE" using the code of Q.i)

iii) Decode the text where encoding is 1100110110.

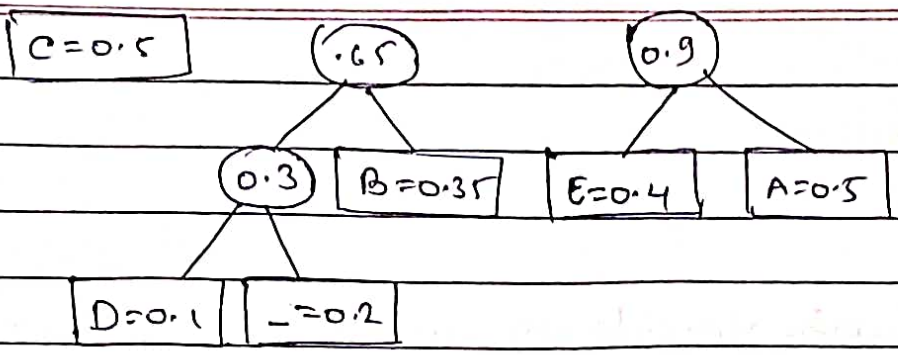
→

Step 1: [D=0.1] [_=0.2] [B=0.35] [E=0.4] [A=0.5] [C=0.5]

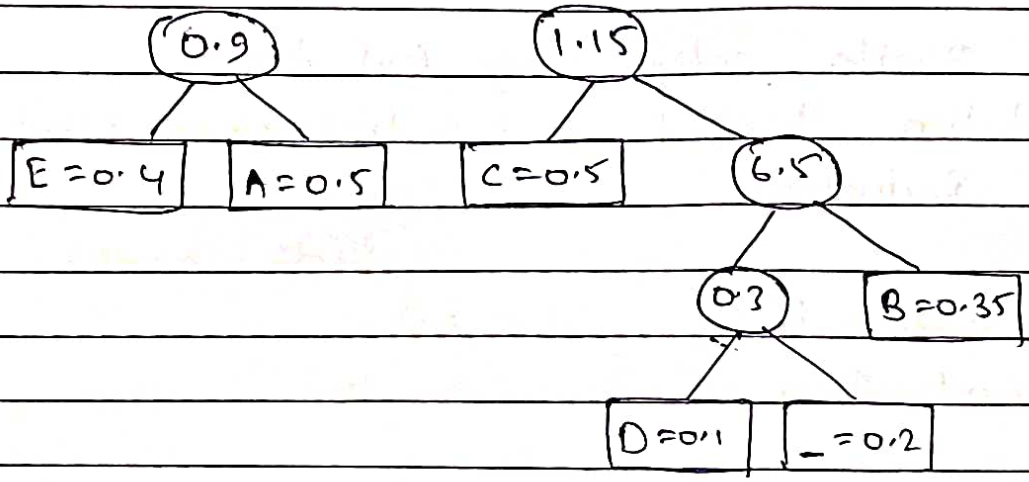
Step 2:- (0.3) [B=0.35] [E=0.4] [A=0.5] [C=0.5]
[D=0.1] [_=0.2]

Step 3:- [E=0.4] [A=0.5] [C=0.5] (0.65)
(0.3) [B=0.35]
[D=0.1] [_=0.2]

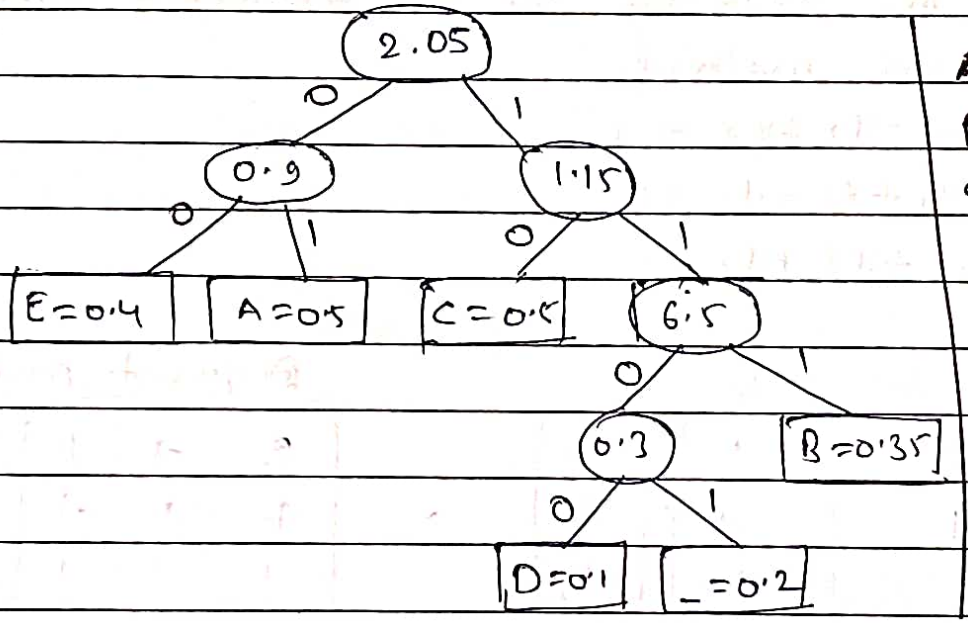
Step 4:



Step 5:



Step 6:



A = 01
B = 111
C = 10
D = 1100
E = 00
_ = 1101

ii) DAD_BE = 1100 01 1100 1101 111 00.

iii) $\frac{1100110110}{D \quad - \quad C}$ = D_C

Transform & conquer

Stages

1) Transformation
2) conquer

Major ~~Variant~~ Variations are :-

1) Instance Simple action (e.g. AVL tree)
2) Representation change (e.g. 2-3 trees, Heap & Heapsort)
3) Problem Reduction

(4) .

- Gaussian elimination
- L.U Decomposition

Q. Solve the following set of equation using Gauss elimination method.

$$2x_1 - x_2 + x_3 = 1$$
$$4x_1 + x_2 - x_3 = 5$$
$$x_1 + x_2 + x_3 = 0$$

Ans

$$Ax = b$$

$$A = \begin{bmatrix} 2 & -1 & 1 \\ 4 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Augment matrix

$$\left[\begin{array}{ccc|c} 2 & -1 & 1 & 1 \\ 4 & 1 & -1 & 5 \\ 1 & 1 & 1 & 0 \end{array}\right]$$

Ans

$$\begin{bmatrix} 2 & -1 & 1 \\ 4 & 1 & -1 \\ 0 & -3 & -1 \end{bmatrix}$$

$x_3 \rightarrow x_1 - 2x_3$

$$= \begin{bmatrix} 2 & -4 & 1 \\ 4 & 4 & -1 \\ 0 & 0 & -1 \end{bmatrix}$$

$C_2 = C_2 - 3C_3$

$$\begin{bmatrix} 2 & -1 & 1 & 1 \\ 0 & 3 & -3 & 3 \\ 0 & 0 & -1 & 4 \end{bmatrix}$$

$R_3 \to R_1 - 2R_3$
$R_2 \to R_2 - 2R_1$
$R_3 \to R_2 + R_3$

$$= \begin{bmatrix} 2 & -1 & 1 \\ 0 & 12 & -3 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 0 \end{bmatrix}$$

$R_3 \to R_2 - 2R_1$

$2x_1 - x_2 + x_3 = 0$
$12x_2 - 3x_3 = 5$
$-x_3 = 0$
$x_3 = 0$

$-1 \cdot x_3 = 1$
$\therefore x_3 = -1$

$3x_2 - 3x_3 = 3$
$x_2 - x_3 = 1$
$\therefore x_2 = 1 - 1 = 0$

$2x_1 - x_2 + x_3 = 1$
$2x_1 - 0 - 1 = 1$
$2x_1 = 2 \quad \therefore x_1 = 1$

* LU decomposition method of Solving system of linear equations.

* Given that,

$$Ax = B \quad - (1) \text{ (Matrix form)}$$

Let, Substitute

$$A = LU \quad - (2)$$

Then from (1),

$$LUx = B \quad - (3)$$

Let $Ux = y$

Then,

$Ly = B$ — find y

$Ux = y$ — find x

Here,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_1 & 1 & 0 \\ l_2 & l_3 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & u_2 & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Q. Solve the following system of equations using LU decomposition methods.

$x_1 + x_2 - x_3 = 4$

$x_1 - 2x_2 + 3x_3 = -6$

$2x_1 + 3x_2 + x_3 = 7$

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & 3 \\ 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ -6 \\ 7 \end{bmatrix}$$

convert into upper triangular matrix,

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & 3 \\ 2 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 0 & -3 & 4 \\ 0 & 1 & 3 \end{bmatrix}$$

$R_2 \to R_2 - R_1$

$R_3 \to R_3 - 2R_1$

$$\begin{bmatrix} 1 & -4 & -1 \\ 0 & -13 & -4 \\ 0 & 0 & 3 \end{bmatrix}$$

$-4 - 3.3$

$C_2 \to C_3 - 3C_2$

$x_3 = 7,$

$-13x_2 - 4x_3 = -6$

$-13x_2 - 4 \times 7 = -6$

$$\begin{bmatrix} 1 & 1 & -1 \\ 0 & -3 & 4 \\ 0 & 1 & 3 \end{bmatrix}$$

$C_2 \to C_2 - C_3/3$

$1 - \frac{(-1)}{3}$

$-3 - \frac{4}{3}$

$$\begin{bmatrix} 1 & 4/3 & -1 \\ 0 & -13/3 & 4 \\ 0 & 0 & 3 \end{bmatrix}$$

Convert into Lower triangular matrix

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & 3 \\ 2 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 0 \\ -5 & -8 & 0 \\ 2 & 3 & 1 \end{bmatrix}$$

$R_1 \to R_1 + R_3$

$R_2 \to R_2 - 3R_3$

$$\begin{bmatrix} 3 & 0 & 0 \\ -5 & -\frac{4}{3} & 0 \\ 2 & \frac{1}{3} & 1 \end{bmatrix}$$

$C_2 \to C_2 - \frac{4}{3} C_1$

$-8 - \frac{4}{3} \times -5$

$-8 + \frac{20}{3} =$

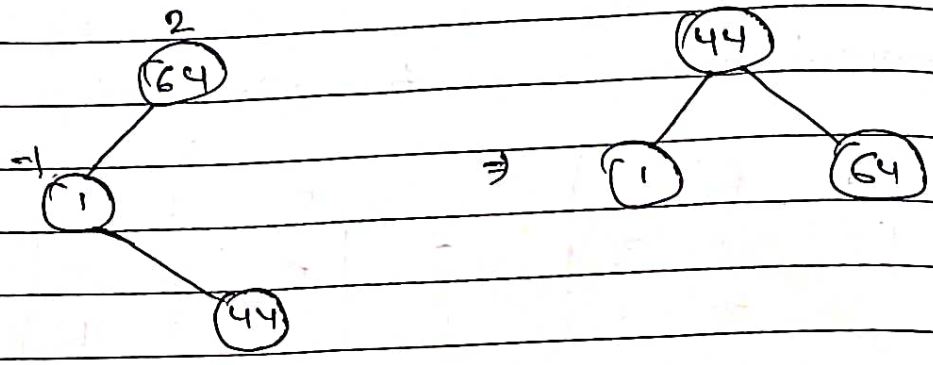
$3 - \frac{4}{3} \cdot 2$

$3 - \frac{1}{3}$

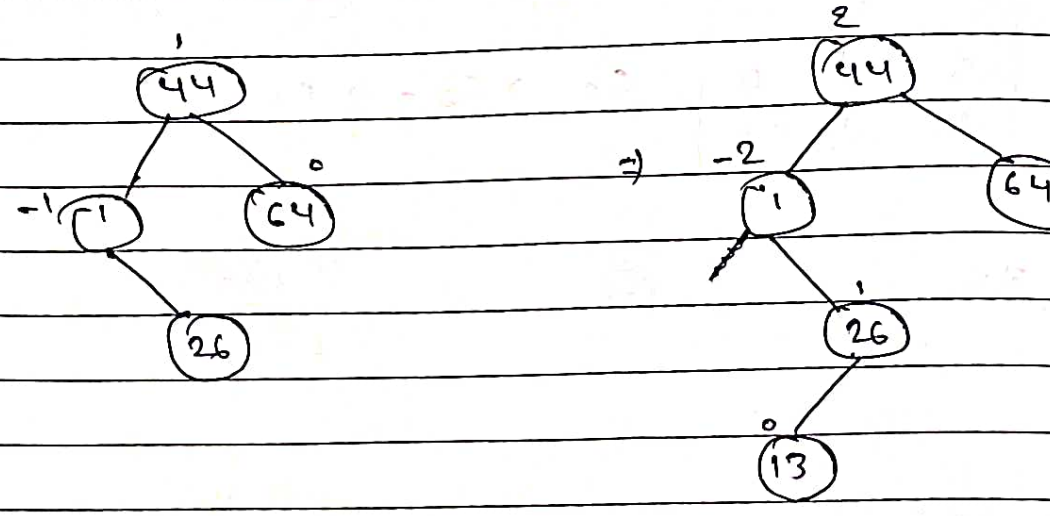
$\times 3$

Q. Construct an AVL tree for the following data items.
64, 1, 44, 26, 13, 110, 98, 85

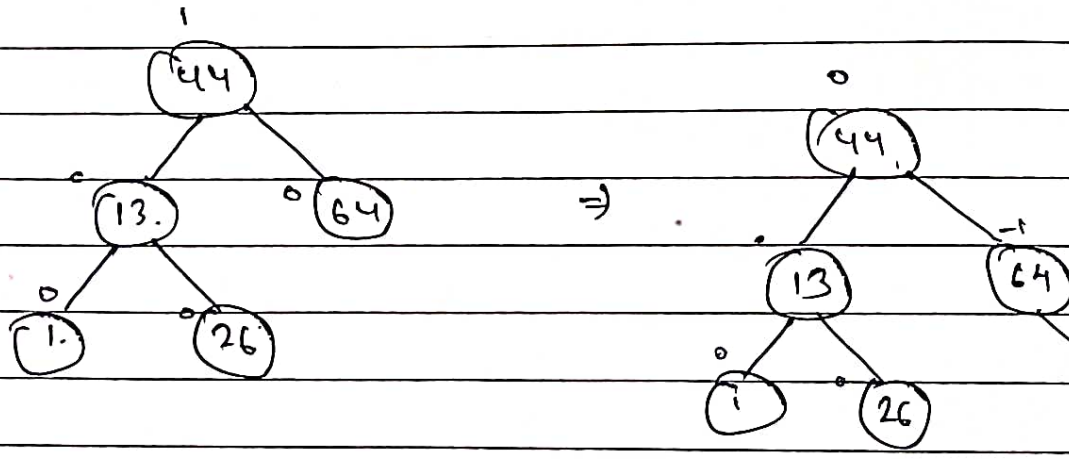
⇒

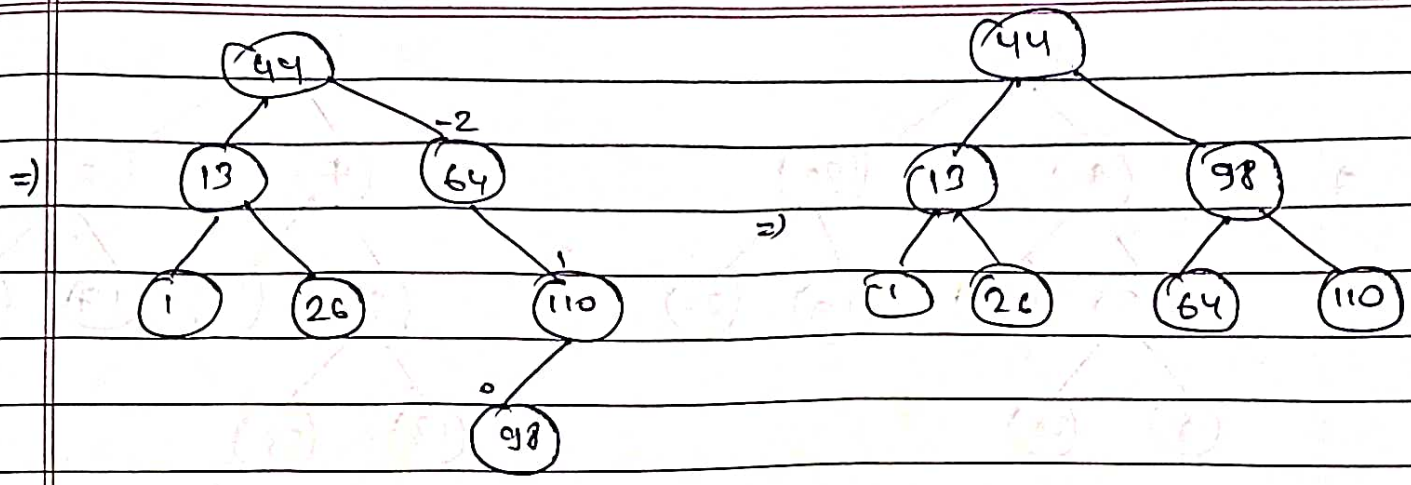


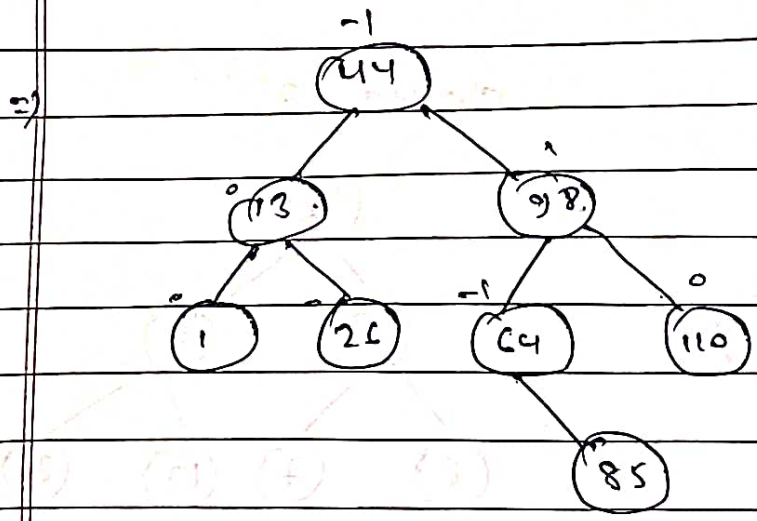
⇒



⇒



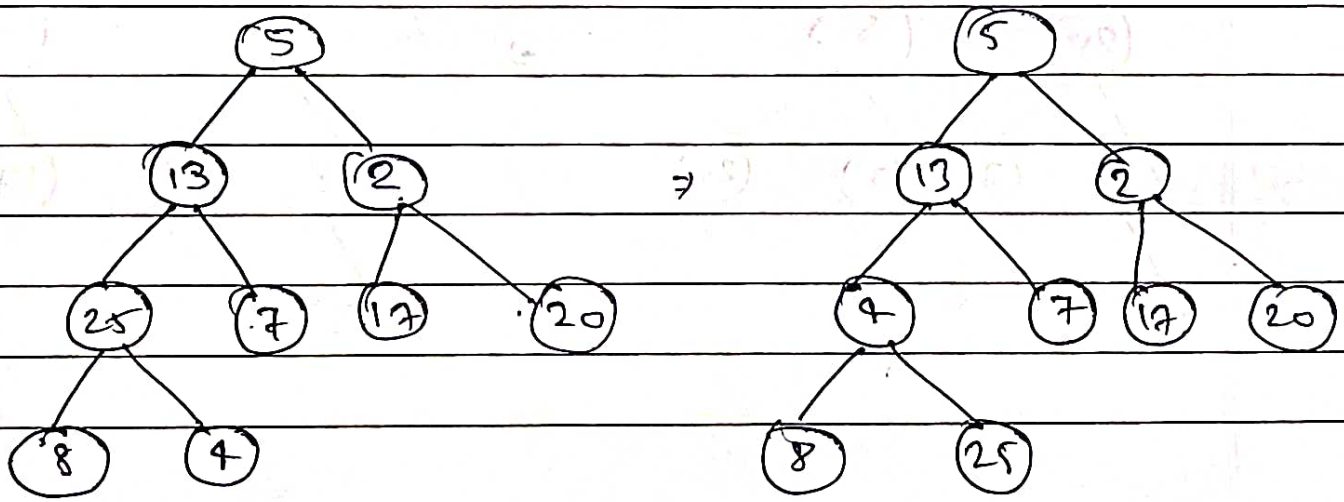


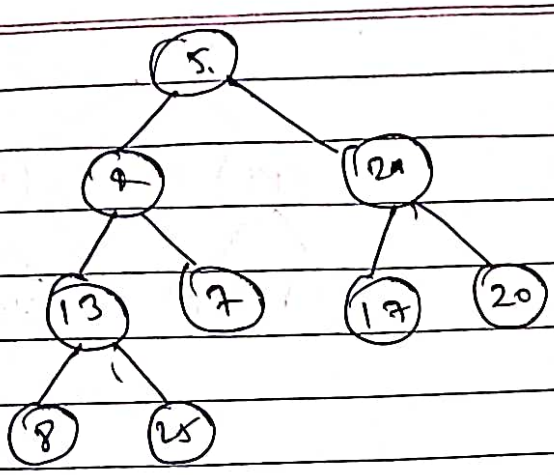


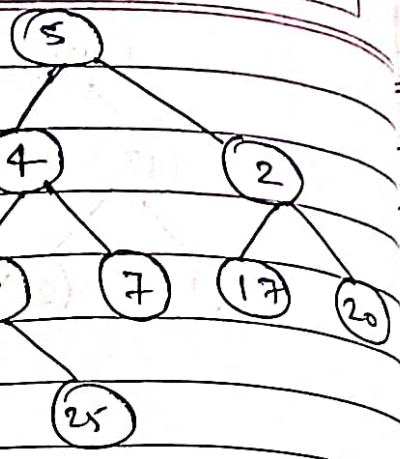
Q. Sort the following elements using heap sort.

5, 13, 2, 25, 7, 17, 20, 8, 4

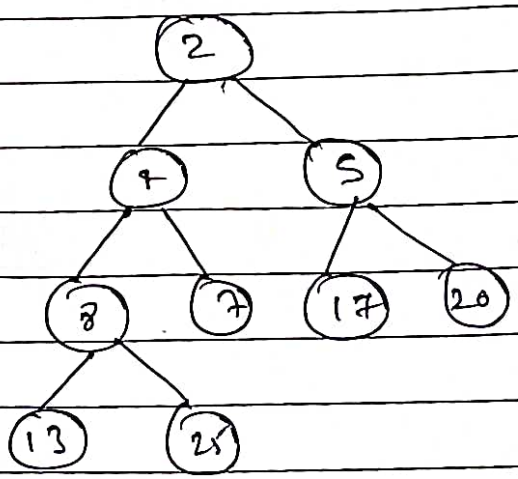
→

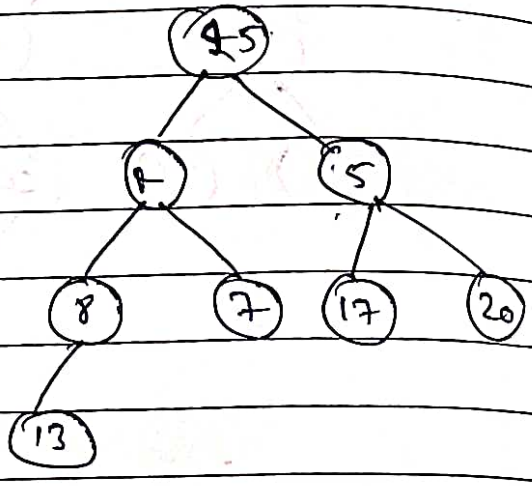


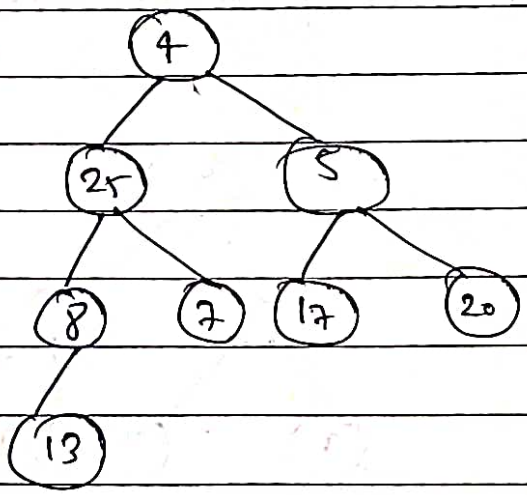

5
4
20
13
7
17
20
8
25

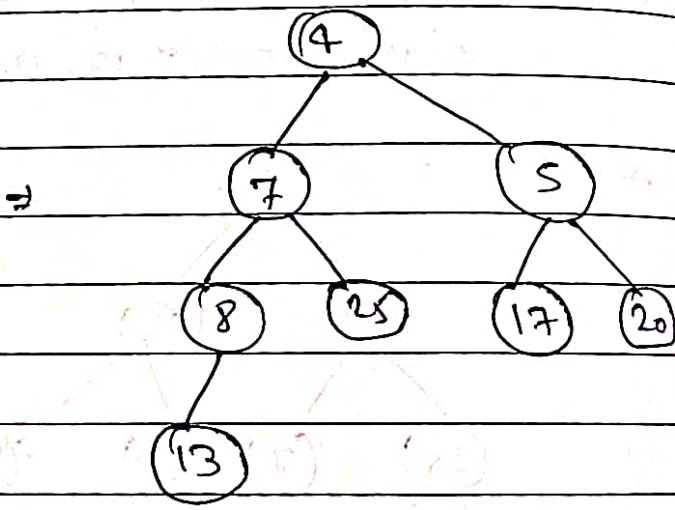

5
4
2
8
7
17
20
13
25

delete 2

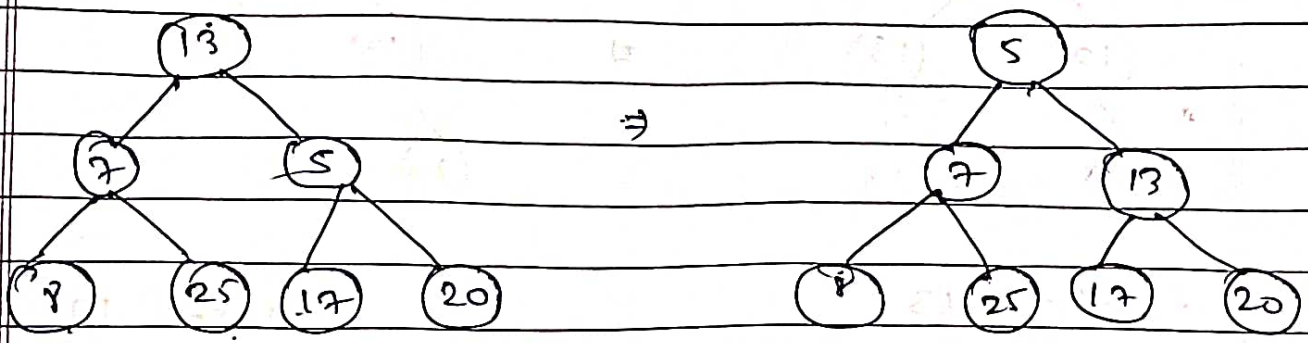

2
4
5
8
7
17
20
13
25


4
4
5
8
7
17
20
13

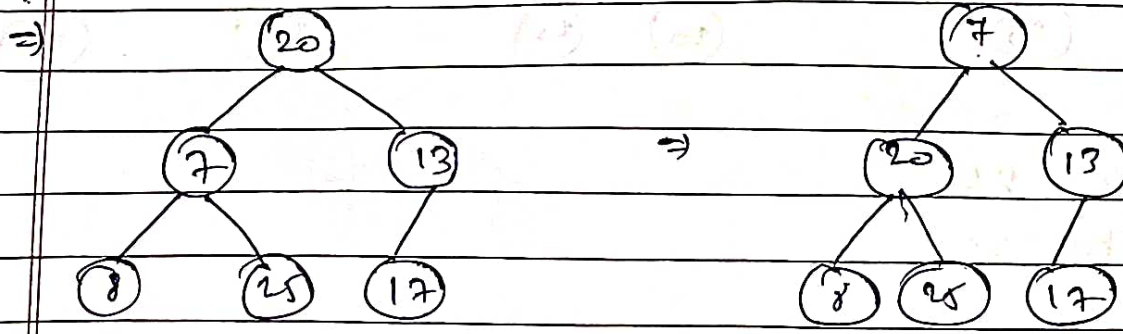

4
25
5
8
7
17
20
13


4
7
5
8
25
17
20
13

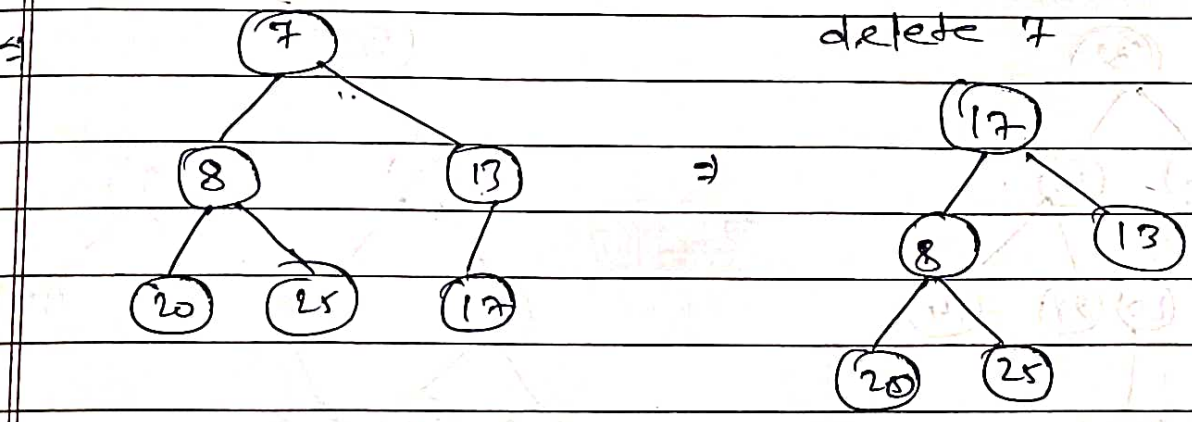
delete 4



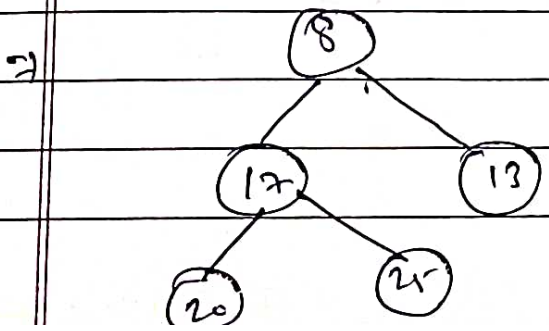
delete 5

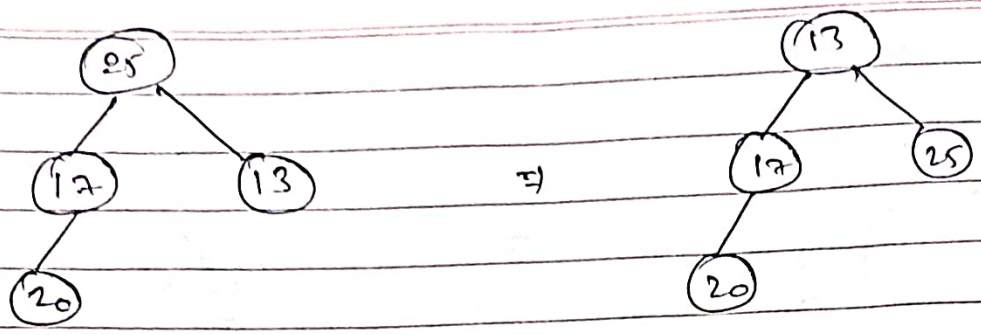


delete 7



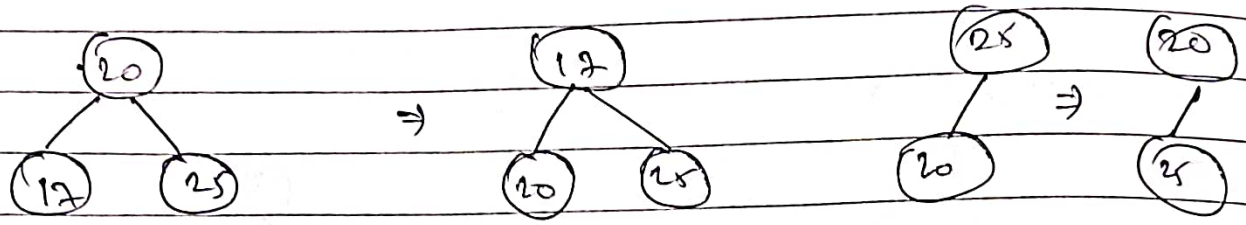
delete 8





delete 13

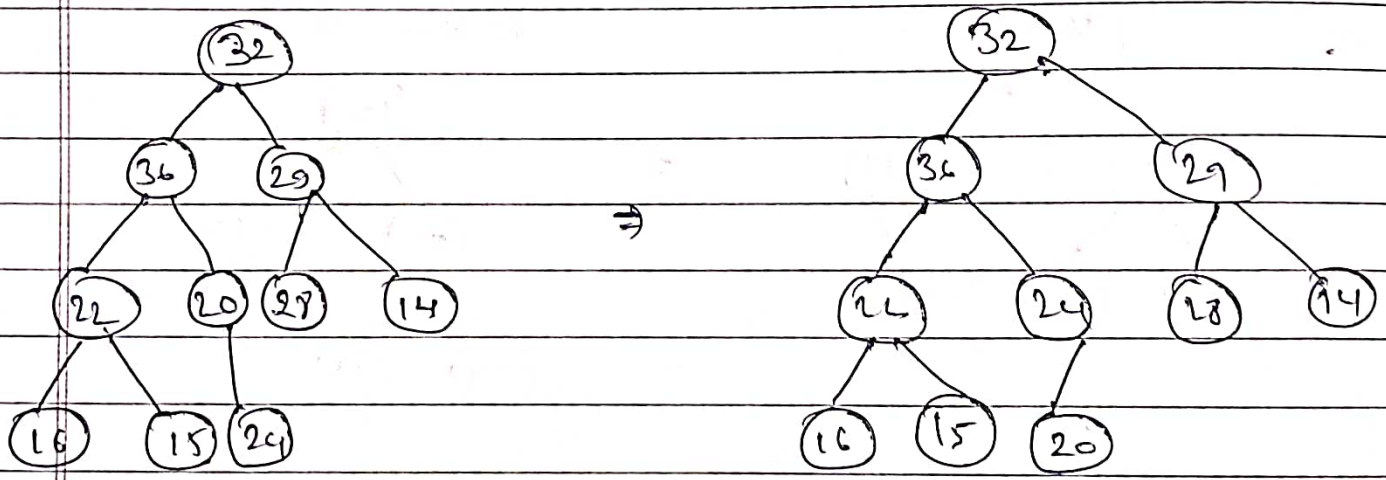
delete 17

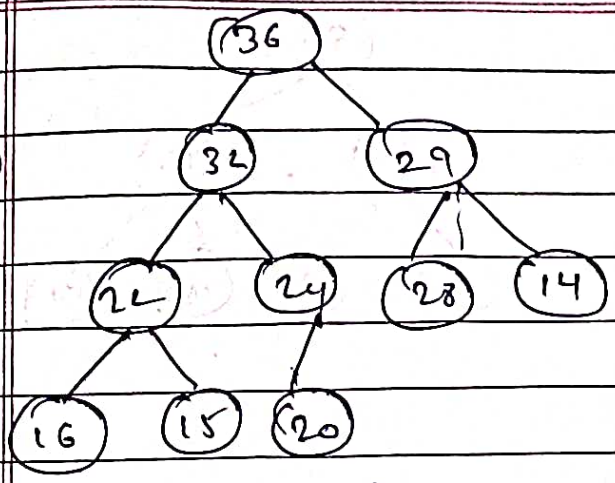


delete 20

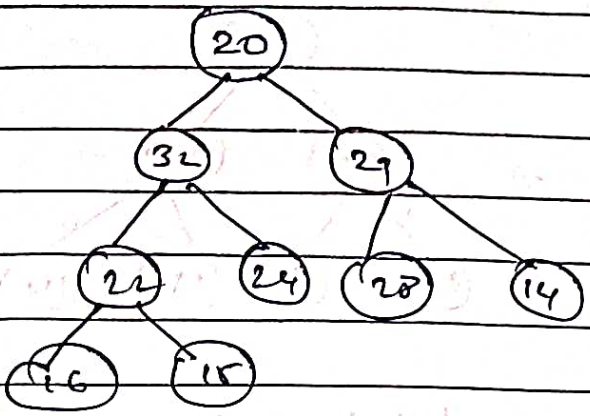
delete 25

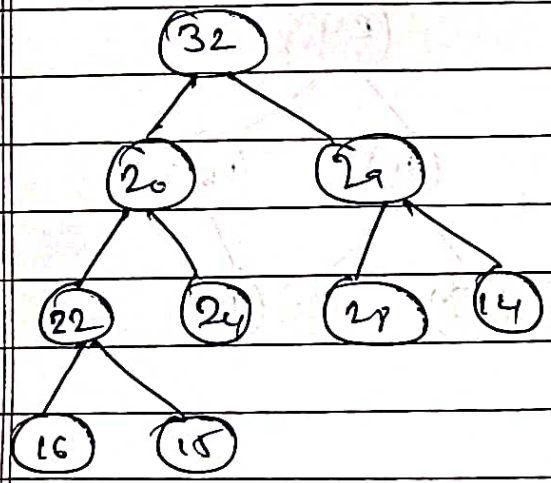
(ii) 32, 36, 29, 22, 20, 28, 14, 16, 15, 24 (max-heap)

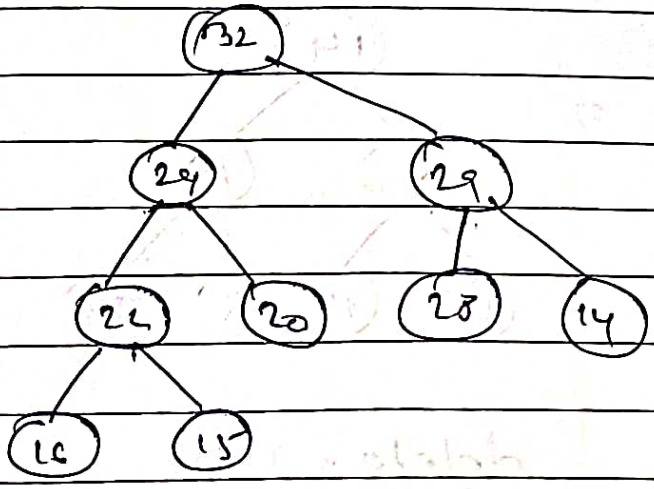




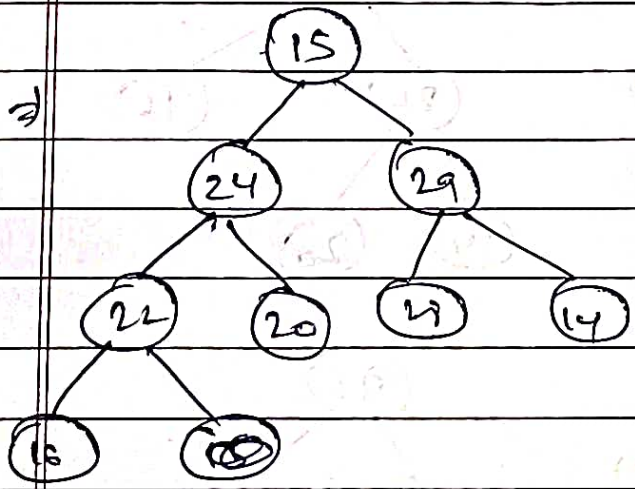
delete 32

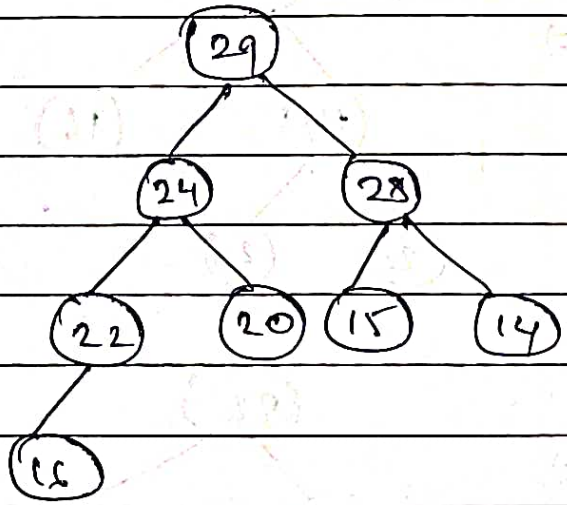




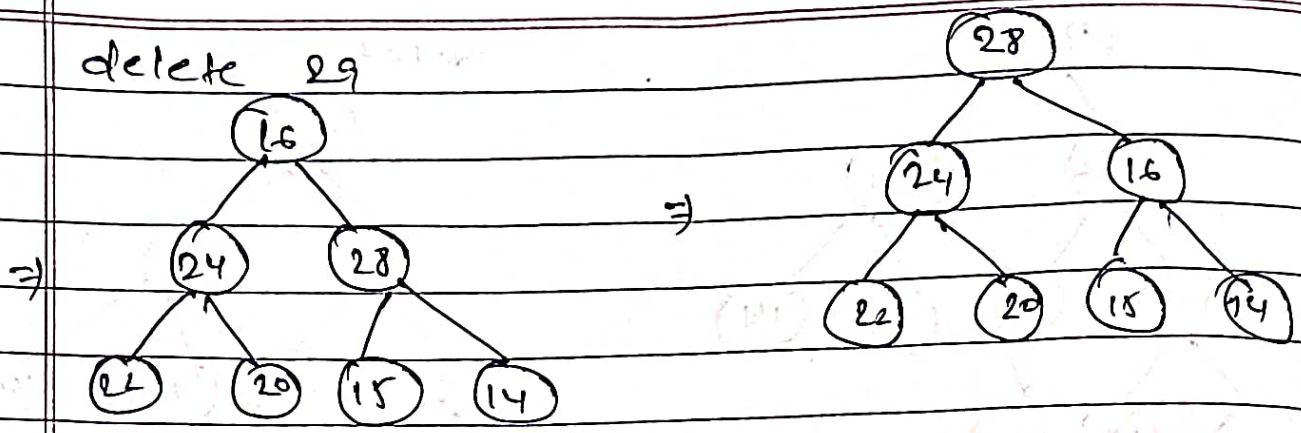


delete 32

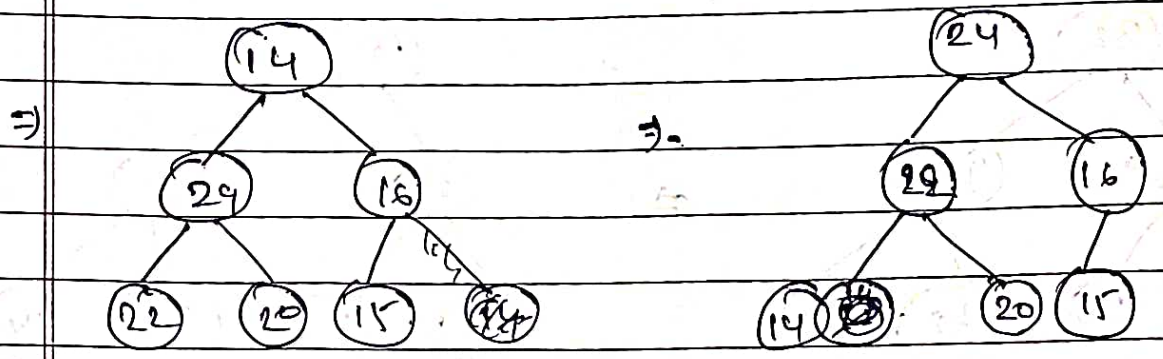




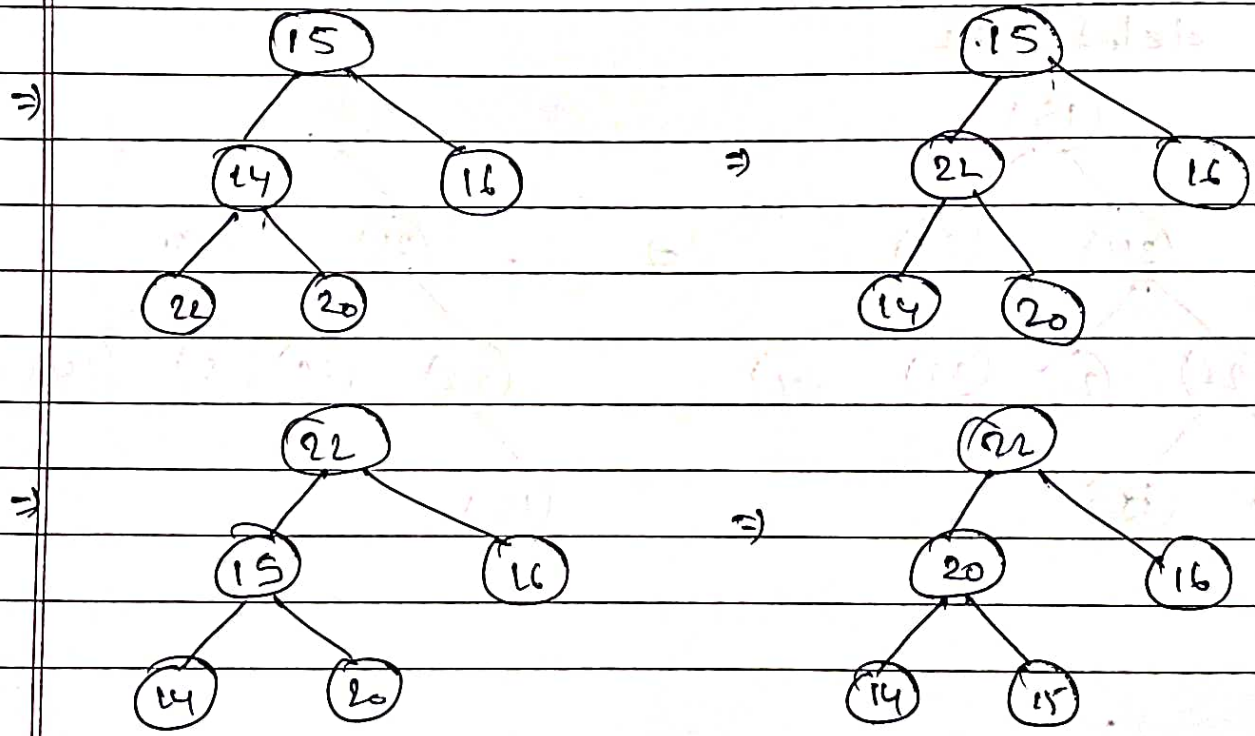
delete 29



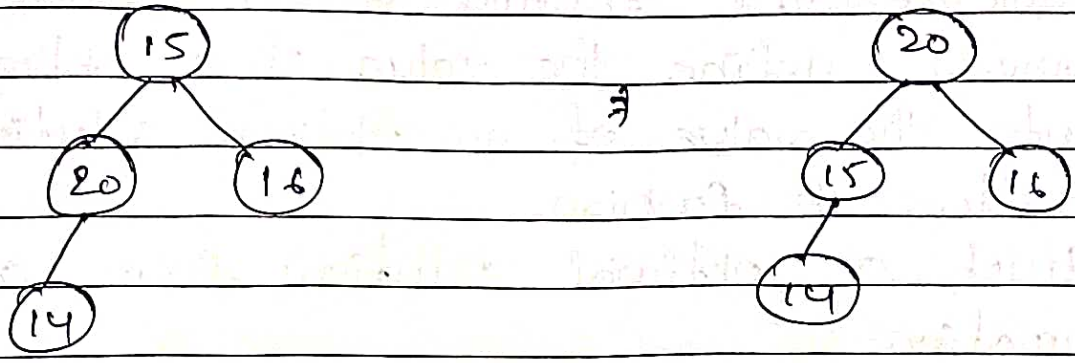
delete 24



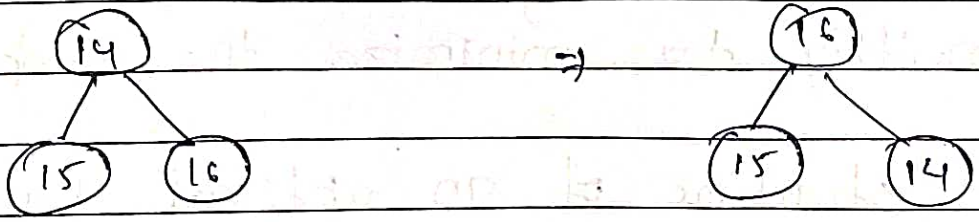
delete 24



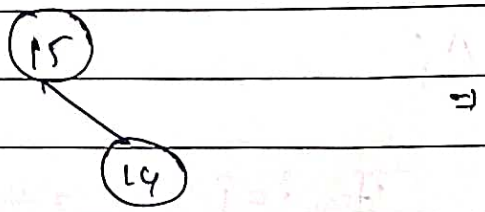
delete 22



delete 20



delete 16



delete 15

delete 14

Dynamic Programming

Steps of dP algorithm

1. Characterise the structure of an optimal soln.
2. Recursively define the value of an optimal soln.
3. Compute the value of an optimal solution in a bottom up fashion.
4. Construct an optimal solution from compute information.
5. Using DP method to find the sequence in which following chain of matrices should be multiply to minimize the computation time

Step 1 :- The structure of an optimal parenthesization.

Step 2 :- Recursively define the value of an optimal solution.

$A_i \ldots A_{i+1} \ldots A_j$

$$m[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min\{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \} & \text{if } i<j \\ & i \le k < j \end{cases}$$

Step 3 :- Compute the value of an optimal solution.

$m[1,n] \rightarrow$ ~~min~~ minimum $A_i \ldots\ldots A_n$.

Step 4 :- Construct an optimal soln.

ex- $A_1 = (10 \times 20)$, $A_2 = (20, 50)$, $A_3 = (50 \times 1)$
$A_4 = (1 \times 100)$

case 1 - $A_1 (A_2 (A_3 A_4))$

→ $A_3 A_4 = (50 \times 1)(1 \times 100) = 5000$
$= (50 \times 100)$ (order)

$A_2 (A_3 A_4) = (20 \times 50)(50 \times 100) = (20 \times 100)$ order
$= 20 \times 50 \times 100$
$= 100000$

$A_1 (A_2 (A_3 A_4))$ order
$= (10 \times 20)(20 \times 100)$ (10×100)
$= 10 \times 20 \times 100$ $= 20000$

Total $= 5000 + 10,00,00 + 20,000$
$= 125000$

case 2 - $(A_1 (A_2 A_3)) A_4$
$= ((10 \times 20) \times (20 \times 1))(1 \times 100)$
$= (10 \times 1) \times (1 \times 100)$
$= 10 \times 100$ (order)

Total $= 20 \times 50 + 10 \times 20 + 10 \times 100$
$= 1000 + 200 + 1000$
$= 2200$

Here, we get the minimum cost.

when K=1 when K=2

$M[1,3]$ $K=1,2$ $= \min\{ m[1,1]+m[2,3]+p_0.p_1.p_3 , m[1,2]+ m[3,3]+p_0.p_2.p_3 \}$

$= \min\{ 0+1000+10.20.1 , 10000+0+10.50.1 \}$

$= \min\{ 1200, 10500 \}$

$= 1200$

when K=2 when K=3

$M[2,4]$ $K=2,3$ $= \min\{ m[2,2]+m[3,4]+p_1.p_2.p_4 , m[2,3]+m[4,4] + p_1.p_3.p_4 \}$

$= \min\{ 0+5000+20.50.100 , 1000+0+20.1.100 \}$

$= \min\{ 105000, 3000 \}$

$= 3000$

when K=1

$M[1,4]$ $K=1,2,3$ $= \min\{ \cancel{\min\{1,}$ $m[1,1]+m[2,4]+p_0.p_1.p_4$ ($K=2$), $m[1,2]+m[3,4]+p_0.p_2.p_4$, $m[1,3]+m[4,4]+p_0.p_3.p_4$ ($K=3$)

$= \min\{ 0+3000+10.20.100 , 10000+5000+10.50.100 , 1200+0+10.1.100 \}$

$= \min\{ 23000, \cancel{110000}\ 65000, 2200 \}$

$= 2200$

Knapsack Problem

→ Maximization problem
→ 0/1 knapsack

Step 1 :- Identify the smaller sub-problem. $S_k = \{1, 2, \ldots k\}$

Step 2 :- Recursively, define the value of an optimal soln in terms of soln of subproblems.

Initial conditions :-

$V[0, j] = 0$ for all $j \geq 0$
$V[i, 0] = 0$ for all $i \geq 0$

- V is 2-d array denoting value

Recursive step :-

$$V[i][j] = \begin{cases} \max \{ V[i-1, j], v_i + V[i-1, j-w_i] \} & \text{if } j - w_i \geq 0 \\ V[i-1, j] & \text{if } j - w_i < 0 \end{cases}$$

Step 3 :- Bottom up computation using iteration.
The goal is to find V and W, the maximal value of a subset of n given items that fit into the knapsack of capacity W and an optimal subset itself.

Q. Apply bottom-up DP to the following instance of knapsack problem with capacity W=5

item	Weight (kg)	Value (Rs)
1	2	3
2	3	4
3	4	5
4	5	6

i \ V	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$V[1][2] = \max\{V[0,2], 3 + V[0,0]\}$ { j-wi : 2-2 = 0

$= \max\{0, 3\} = 3$

$V[1][3] = \max\{V[0,3], 3 + V[0,1]\}$ {3-2 = 1 ≥ 0

$= \max\{0, 3\} = 3$

$V[2][1] = V[1,1] = 0$ { 1-3 < 0

$V[2][2] = V[1,2] = 3$ { 2-3 < 0

$V[2][3] = \max\{V[1,3], 4 + V[1,0]\}$ { ~~3-4 ≥ 0~~ 3-3 = 0

$= \max\{3, 4\} = 4$

Optimal Value = $v[4,5] = 7$

Now,

$v[4,5] = v[3,5]$

⇒ item 4 not included in the subset

$v[3,5] = v[2,5]$

⇒ item 3 not included in the subset

$v[2,5] \neq v[1,5]$

⇒ item 2 included in the subset

Remaining capacity = 5 - 3 = 2 Kg

$v[1,2] \neq v[0,2]$

⇒ item 1 included in the subset

∵ weight - weight selected

⇒ 2 - 2 = 0

∴ No items left to select.

Optimal Subset = {1, 2}

Value = 7

1Q. Solve the following 0/1 knapsack problem using dynamic programming.

item	weight	value
1	2	11
2	11	21
3	22	31
4	15	33

item	weight	value
1	1	1
2	2	6
3	4	4

capacity = 3

2Q. Matrix chain multiplication by DP.

~~A(2~~ A(4x5), B(5x3), C(3x2), D(2x7), E(7x2)

Answer

(1)

i \ j	0	1	2	3	4
0	0	0	0	0	0
1	0				
2	0				
3	0				

2. $P_0 = 4, P_1 = 5, P_2 = 3, P_3 = 2, P_4 = 7, P_5 = 2$

	1	2	3	4	5
1	0	60	70	126	110
2		0	30	100	70
3			0	42	40
4				0	28
5					0

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

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

$k=1 \quad = \min\{ 0 + 4 \times 5 \times 3 \} = 60$

$M[1,3] = \min\{ m[1,1] + m[2,3] + P_0 \cdot P_1 \cdot P_3, m[1,2] + m[3,3] + P_0 P_2 P_3 \}$

$= \min\{ 0 + 30 + 4 \cdot 5 \cdot 2, 60 + 0 + 4 \cdot 3 \cdot 2 \}$

$= \min\{ 70, 84 \} = 70$

$k = 1,2,3$

$M[1,4] = \min\{ m[1,1] + m[2,4] + P_0 \cdot P_1 \cdot P_4, m[1,2] + m[3,4] + P_0 \cdot P_2 \cdot P_4, m[1,3] + m[4,4] + P_0 \cdot P_3 \cdot P_4 \}$

$= \min\{ 0 + 100 + 4 \cdot 5 \cdot 7, 60 + 42 + 4 \cdot 3 \cdot 7, 70 + 4 \cdot 2 \cdot 7 \}$

$= \min\{ 240, 186, 126 \} = 126$

$k = 1,2,3,4$

$M[1,5] = \min\{ m[1,1] + m[2,5] + P_0 \cdot P_1 \cdot P_5, m[1,2] + m[3,5] + P_0 \cdot P_2 \cdot P_5, m[1,3] + m[4,5] + P_0 \cdot P_3 \cdot P_5, m[1,4] + m[5,5] + P_0 \cdot P_4 \cdot P_5 \}$

$= \min\{ 70 + 4 \cdot 5 \cdot 2, 60 + 40 + 4 \cdot 3 \cdot 2, 70 + 28 + 4 \cdot 2 \cdot 2, 126 + 4 \cdot 7 \cdot 2 \}$

$= \min\{ 110, 124, 114, 182 \}$

$= 110$

Final ans :- $A(B(C(DE)))$

(5) Shortest Path Problem

All pair shortest path problem
(Floyd's Algorithm)

Given a weighted graph G(V, E)w, the all pairs shortest path problem is to find the shortest path b/w every pair of vertices (Vi, Vj)∈V.

Underlying idea of Floyd's algorithm

- Let W denote the initial weight matrix
- Let D(K) [i,j] denote cost of shortest path from i to j whose intermediate vertices are a subset of {1,2,....K}.

- Recursive definition

Case 1: A shortest path from Vi to Vj restricted to using only vertices from {V1, V2, — VK} as intermediate vertices does not use VK. Then

$$D(K)[i,j] = D(K-1)[i,j]$$

Case 2:- A shortest path from Vi to Vj restricted to using only vertices from {V1, V2, ---- VK} as intermediate vertices to use VK. Then

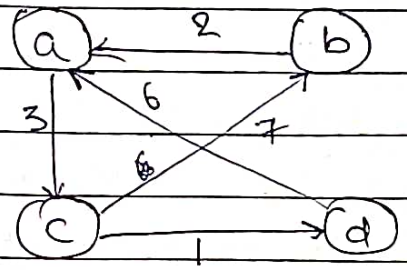
$$D(K)[i,j] = D(K-1)[i,k] + D(K-1)[k,j]$$

We conclude :

$D(k)[i,j] = \min \{ D(k-1)[i,j], D(k-1)[i,k] + D(k-1)[k,j] \}$

* Algorithm Floyad
// Input : Weight matrix W
// Output : Distance matrix of shortest path's length

```
D ← W
for K = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            D[i,j] = num{ D[i,j], D[i,k] + D[k,j] }
return D
```



$W = D^0$

	a	b	c	d
a	0	∞	3	∞
b	2	0	∞	∞
c	∞	7	0	1
d	6	∞	∞	0

$P =$

	a	b	c	d
a	a	b	c	d
b	a	b	c	d
c	a	b	c	d
d	a	b	c	d

$D^{(1)} =$

	a	b	c	d
a	0	∞	3	∞
b	2	0	5	∞
c	∞	7	0	1
d	6	∞	9	0

$D'_{bc} = \min \{ D^0[b,c] ; D^0[b,a] + D^0[a,c] \}$
$= \min \{ \infty, 2+3 \}$
$= 5$

$D'_{bd} = \min \{ D^0[b,d], D^0[b,a] + D^0[a,c] + D^0[c,d] \}$
$= \min \{ \infty, 2+3+1 \}$
$= \min \{ \infty, 6 \}$
$= 6$

We cannot take it because it has 2 intermediate vertex.

$D'[c,b] = \min \{ D^0[c,b], \infty \}$
no any intermediate vertex

$D'[d,c] = \min \{ \infty, 6+3 \}$
$= 9$

Now, $P =$

a	b	c	d
a	b	a	d
a	b	c	d
a	b	a	d

$D^{(1)}$ =

	a	b	c	d
a	0	∞	3	∞
b	2	0	5	∞
c	9	7	0	1
d	6	∞	9	0

$D^2[c,a] = \min\{\infty, 7+2\}$

$= 9$

Now,

$P =$

a	b	c	d
a	b	a	d
b	b	c	d
a	b	a	d

D^3 =

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	9	7	0	1
d	6	16	9	0

D_4 =

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	7	7	0	1
d	6	16	9	0

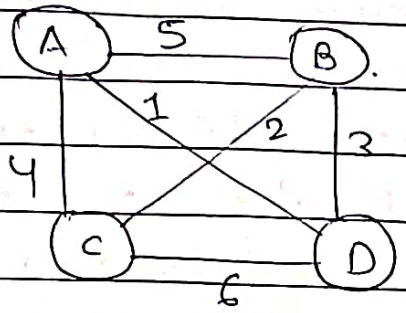
$P =$

	a	b	c	d
a	a	c	c	c
b	a	b	a	c
c	a	b	c	d
d	a	a	a	d

- $d \to b$

→ $d \to a \to c \to b$

Q.



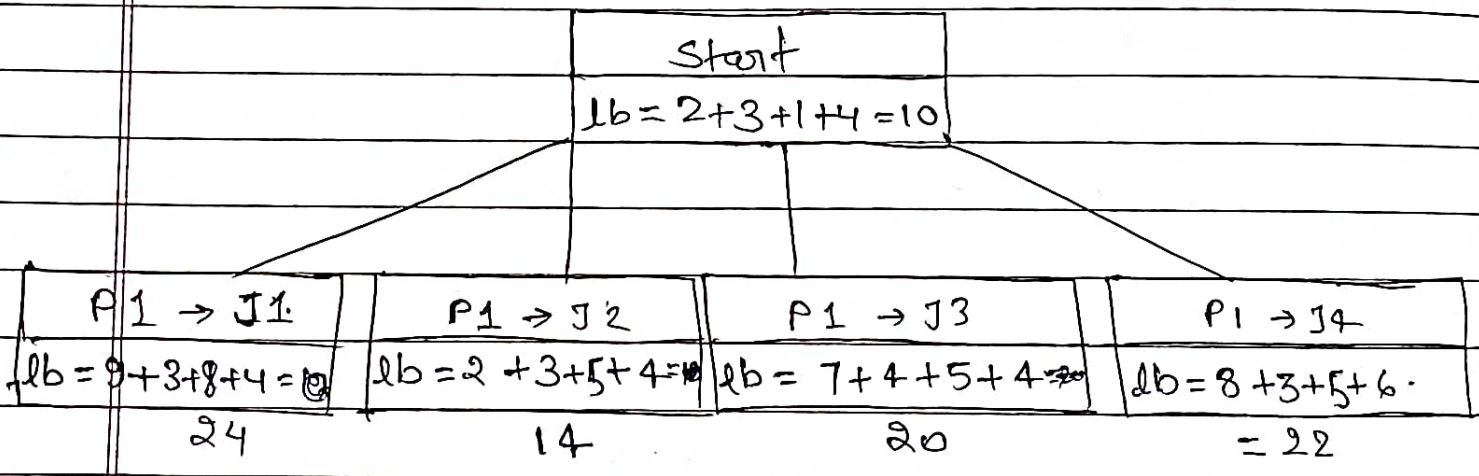
Branch and Bound

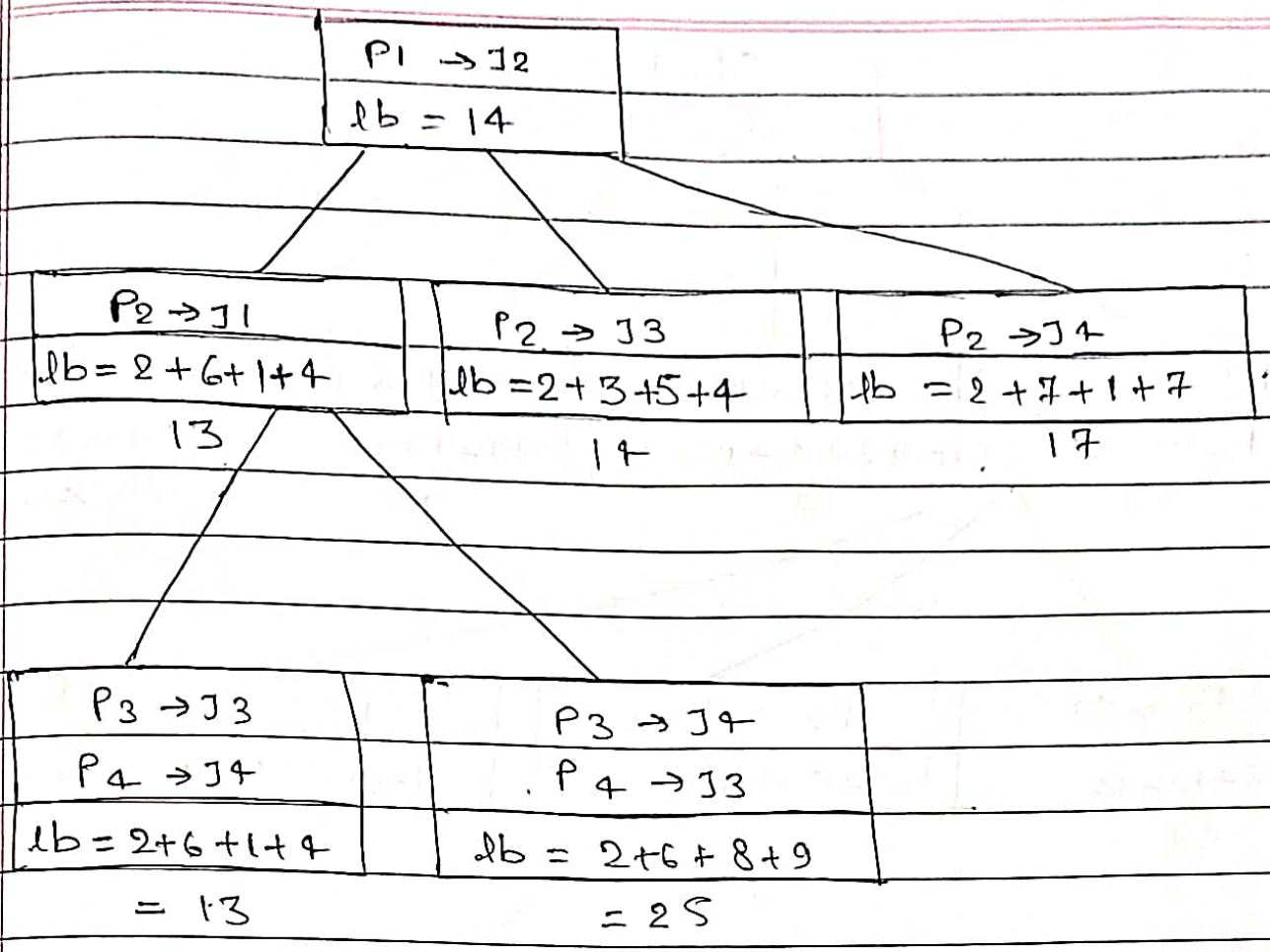
It is an algorithm design paradigm which is generally used for solving combinational problems. These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in worst case. This technique solve these problems relatively quickly.

Problems :- 0/1 Knapsack
TSP
Assignment problem.

Q. Solve the following assignment problem using branch and bound technique.

Persons \ Jobs	J_1	J_2	J_3	J_4
P_1	9	(2)*	7	8
P_2	6*	4	(3)	7
P_3	5	8	(1)*	8
P_4	7	6	9	(4)*



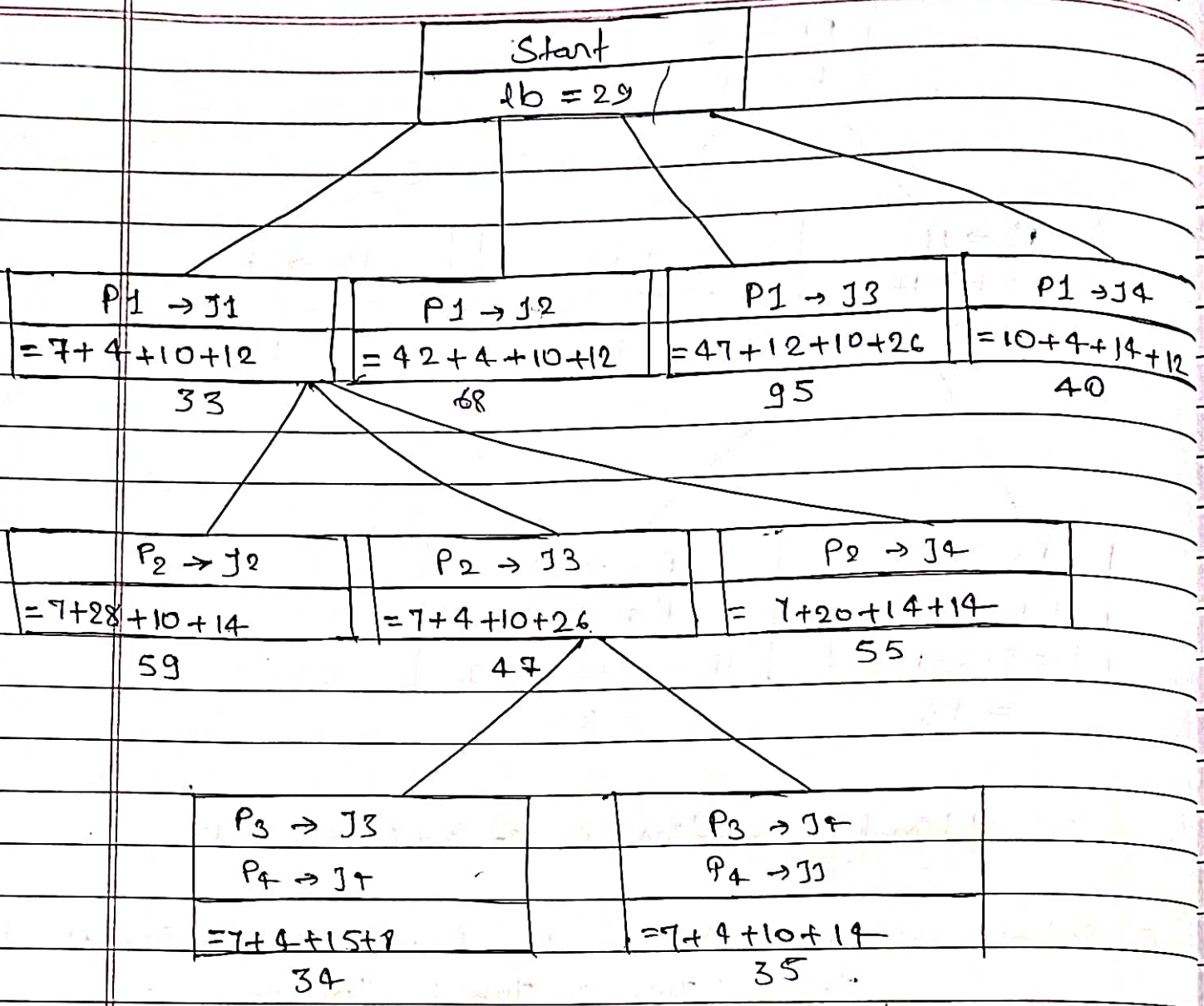


∴ Optimal Solution is :- 2+6+1+4 = 13

Q. Solve assignment problem by branch & bound method.

	Job 1	Job 2	Job 3	Job 4
Person A	(7)*	42	47	10
Person B	12	28	(4)*	20
Person C	34	14	15	(10)
Person D	12	26	14	(8)

Start
lb = 7+4+10+8 = 29



∴ optimal Soln lb :- 7+4+15+8

= 34

2) 0/1 knapsack problem

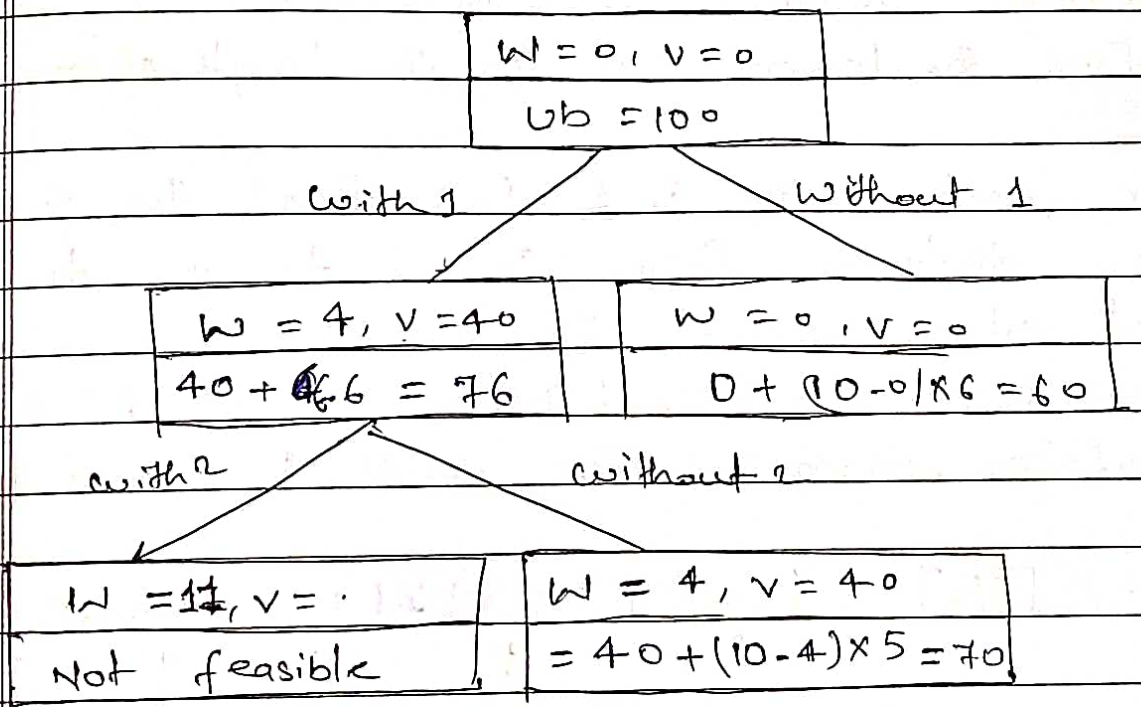
item	weight	value	value/weight
1	4	40	40/4 = 10
2	7	42	42/7 = 6
3	5	25	25/5 = 5
4	3	12	12/3 = 4

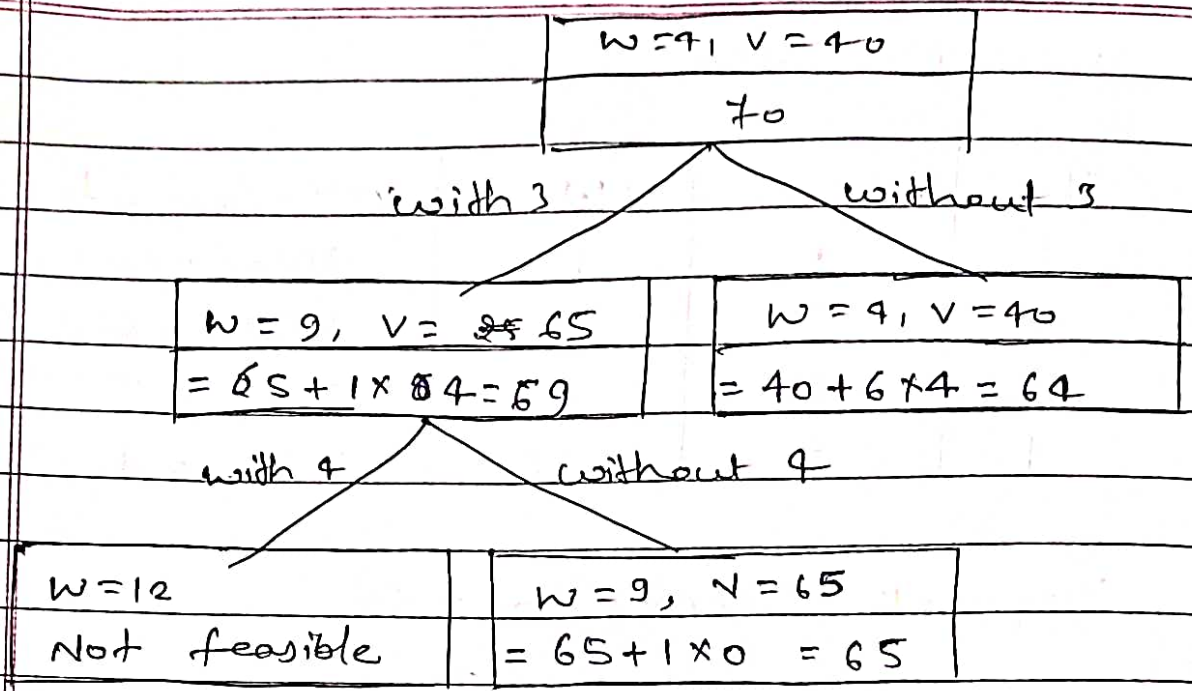
Knapsack Capacity (w) = 10

- Arrange in descending order
- Upper bound = $v + (w - w)(v_{i+1}/w_{i+1})$

Now,

ub = 0 + (10-0) · 10 = 100





optimal subset = {1, 3}
optimal solution = 65

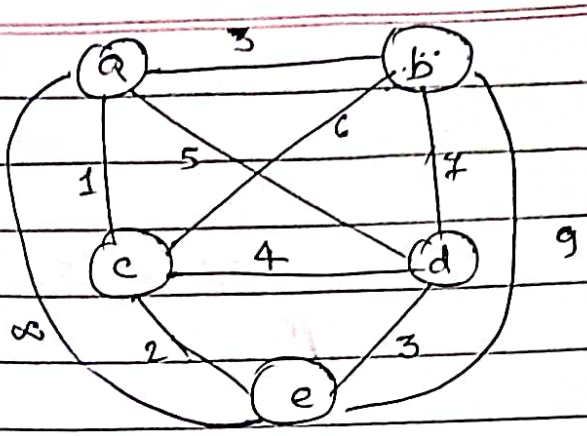
(3) Travelling Salesman problem (TSP)
(To find the lower bound on the length of any tour)

For each city i, $1 \leq i \leq n$, find the sum s_i of the distances from city i to the two nearest cities. Compute the sum (s) of these n numbers, divide the result by 2. If all the distances are integers, round up the result to nearest integer.

$$lb = \lceil s/2 \rceil = \lceil \{(1+3) + (3+6) + (1+2) + (3+4) + (2+3)\}/2 \rceil$$

(a,c) & (a,b) (b,a) & (b,c) c d e

$$= 28/2 = 14$$



0

a
lb = 14

1	2	3	4
a, b	a, c	a, d	a, e
lb = 14		lb = 16	lb = 19

b is not before c

for a,d = $[\{(1+5)+(3+6)+(1+4)+(3+5)+(2+3)\}/2]$

= $\lceil 31/2 \rceil$ = 16

for a,e = $[\{(1+8)+(3+6)+(1+2)+(3+4)+(2+8)\}/2]$

= 38/2 = 19

a, b
lb = 14

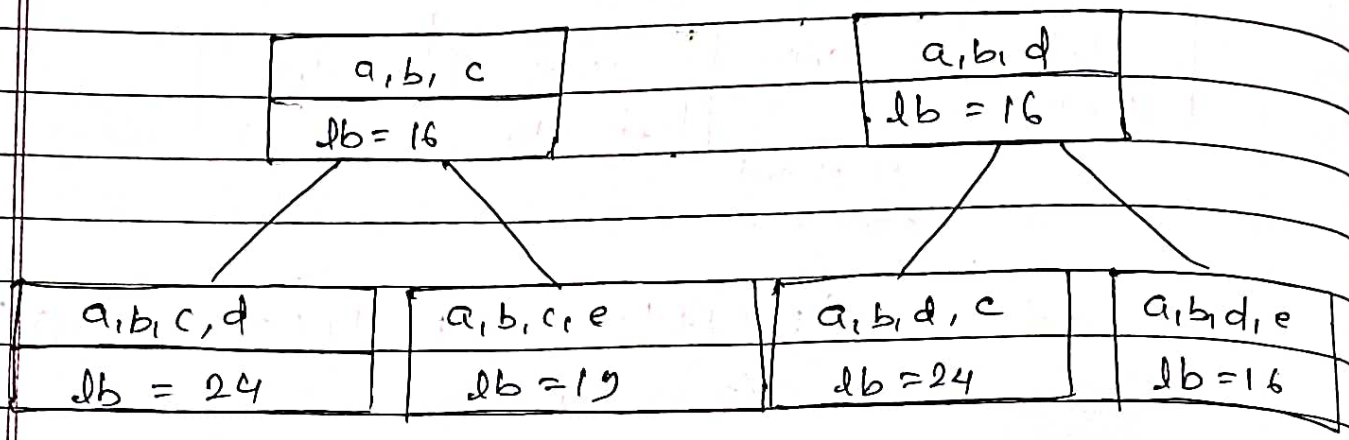
a, b, c	a, b, d	a, b, e
lb = 16	lb = 16	lb = 19

for a, b, c = $[\{(1+3)+(3+6)+(1+6)+(3+4)+(2+3)\}/2]$

= 32/2 = 16

for a,b,d = [{(1+3)+(3+7)+(1+2)+(9+7)+(2+3)}/2]
= 32/2 = 16

for a,b,e = [{(1+3)+(9+3)+(1+2)+(3+4)+(2+9)}/2]
= ⌈37/2⌉ = 19



~~for a,b,c,d = [{(1+3)+(3+6)+(6+4)+(4+3)+(2+3)}/2]~~
=

for a,b,c,d = [{(8+3)+(3+6)+(6+4)+(4+3)+(2+3)}/2
= 48/2 = 24

for a,b,c,e = [{(3+5)+(3+6)+(6+2)+(2+3)+(5+3)}/2]
= 38/2 = 19

for a,b,d,c = [{(8+3)+(3+7)+(7+4)+(4+2)+(2+3)}/2]
a b d c e

= 48/2 = 24

for a,b,d,e = [{(1+3)+(3+7)+(7+3)+(2+3)+(3+2)}/2]
a b d e c

= 32/2 = 16

Solution → a → b → d → e → c → a

Cost → 3 + 7 + 3 + 2 + 1 = 16

Q.

item	Value	weight
1	50	10
2	48	6
3	30	5
4	12	3

Knapsack Capacity (W) = 12

Q.

	1	2	3	4	5
1	∞	17	7	35	18
2	9	∞	5	14	19
3	29	24	∞	30	12
4	27	21	25	∞	48
5	15	16	28	18	∞

Backtracking

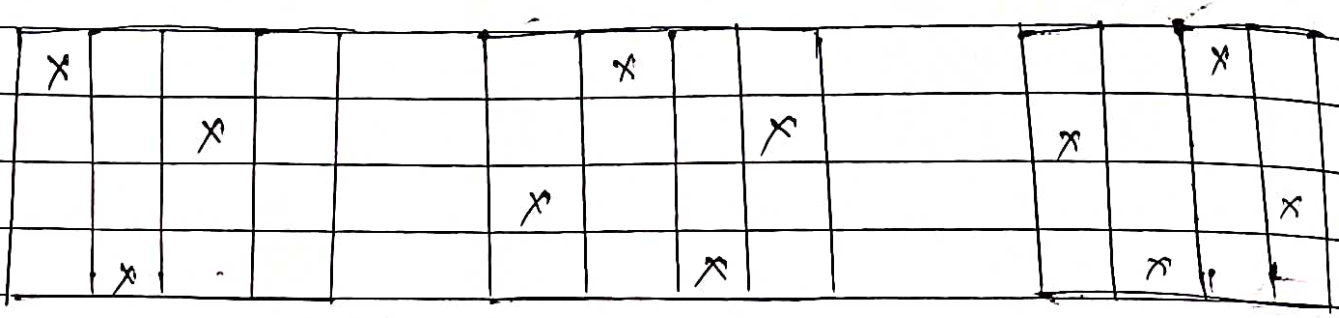
Page No. : 61
Date : / /

It is an algorithm design technique to solve problems by an incremental way. It uses recursive approach to solve problems. ~~by an incremental way. It uses recursive approach to.~~ It is used to find all possible ~~co~~ combinations to solve an optimization problem.

Problems

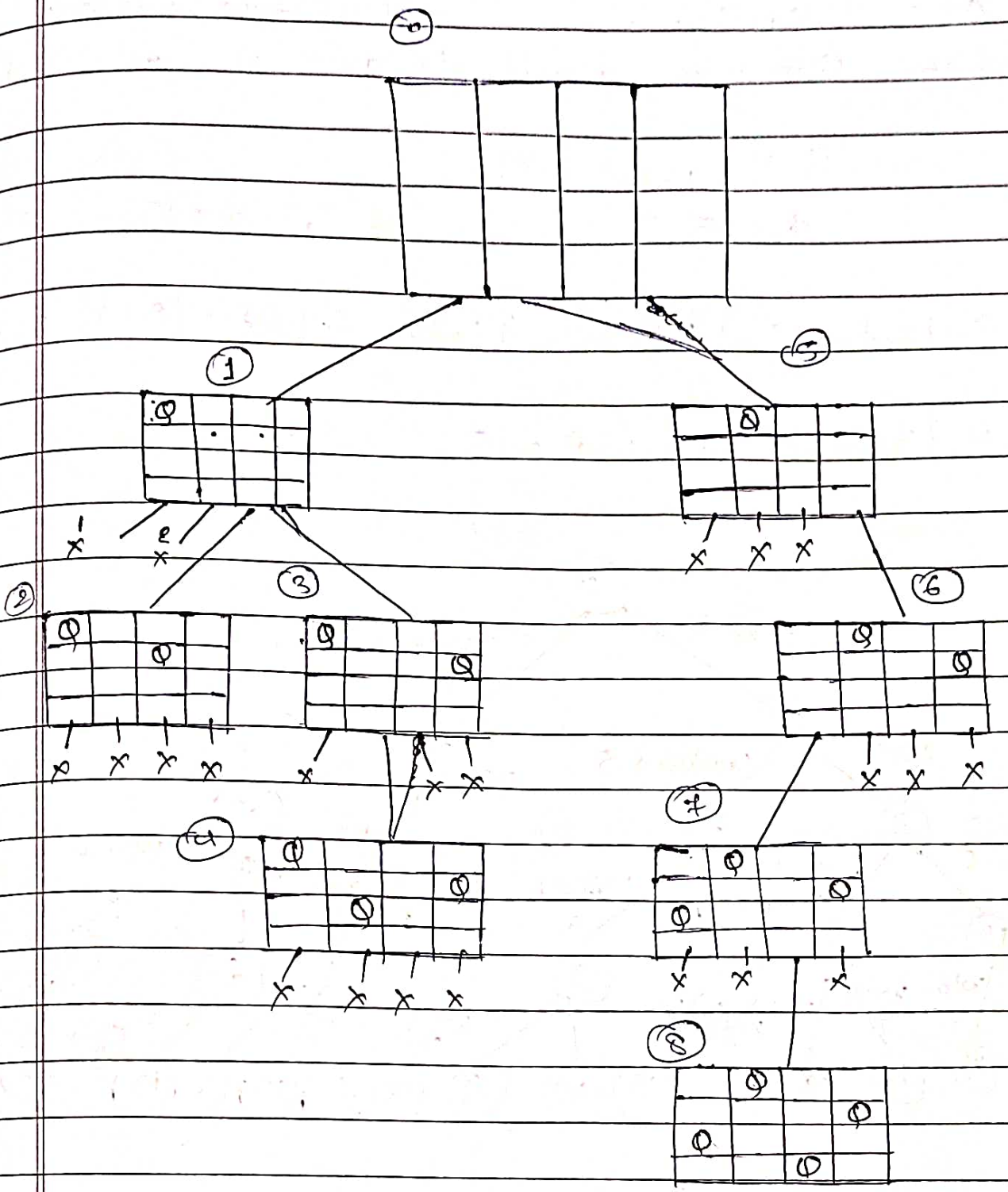
N-queen Problem
Sum of Subset Problem
Hamiltonian Circuit

N-queen Problem



<2,4,1,3>

<3,1,4,2>



Another solution : <3,1,4,2> <2,4,1,3>

↓

is achieved when nc. place. 1st queen 3rd column of 1's row & proceed.

Sum of Subset Problem

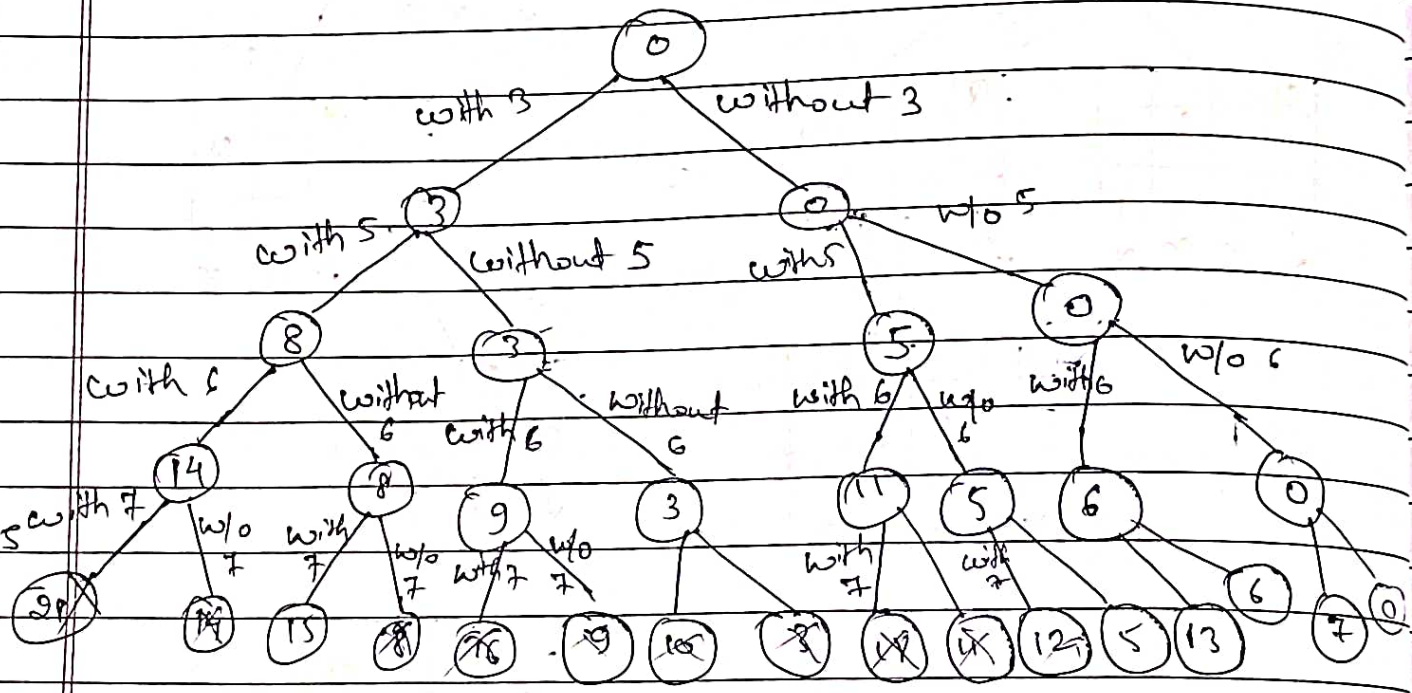
$A = \{a_1, a_2, \ldots, a_n\}$ of n +ve integers, whose sum is equal to given +ve integer d.

$S = \{1, 2, 5, 6, 8\}$, $d = 15$

$d = 9$

Subset = {1,2,6}, {1,8} Subset = {2,5,8}, {1,6,8}

Q. $A = \{3, 5, 6, 7\}$, $d = 15$



$s + a_{i+1} > d$

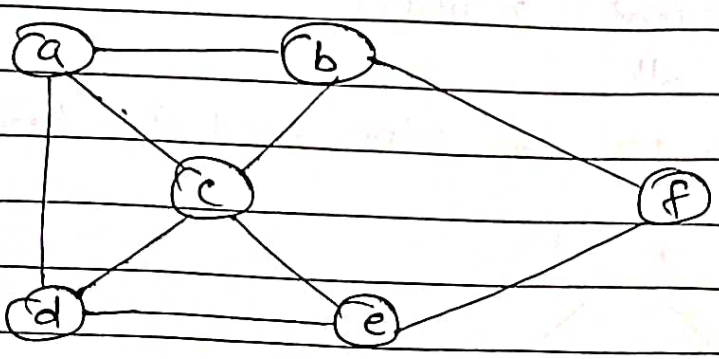
$s + \sum_{j=i+1}^{n} a_j < d$

} stop searching

Hamiltonian Circuit Problem

Page No. : 68.
Date : / /

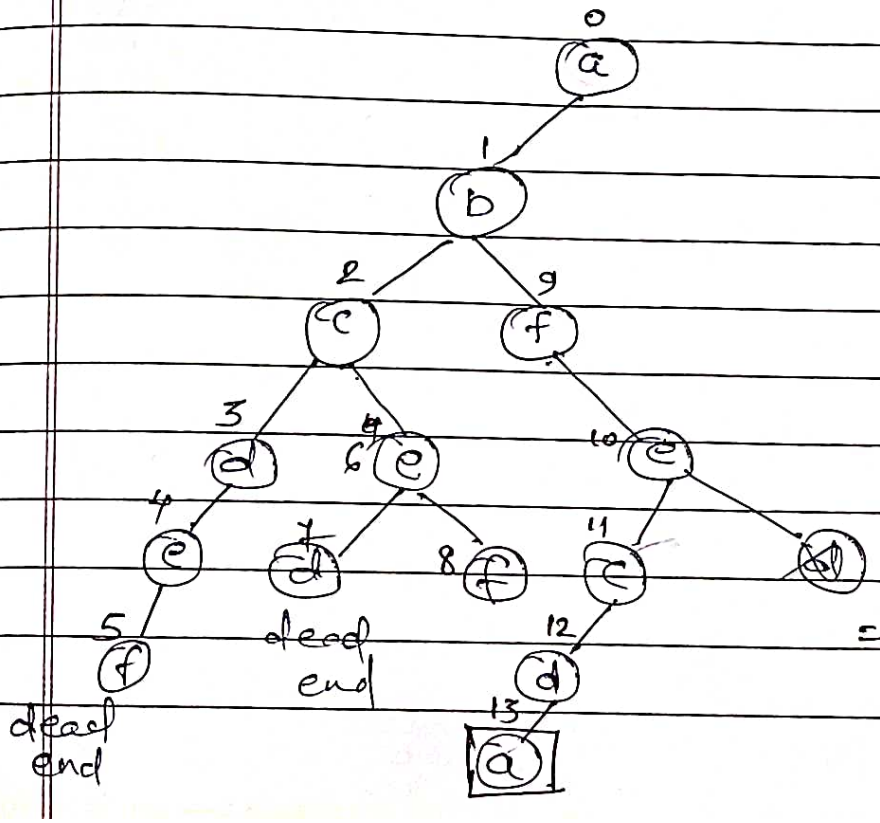
complete graph → A graph is complete if there is exactly one edge b/w every pair of vertices.



check करना है कि Graph में Hamiltonian Circuit है कि नहीं, यदि बिना पेन उठाये सारे vertices cover हो जाये बिना edge repeat किये तो hamiltonian circuit होगा।

* Backtracking method

State space tree for finding hamiltonian circuit.



Now, we reached At 'A'.

Soln

= {a → b → f → e → c → d → a}

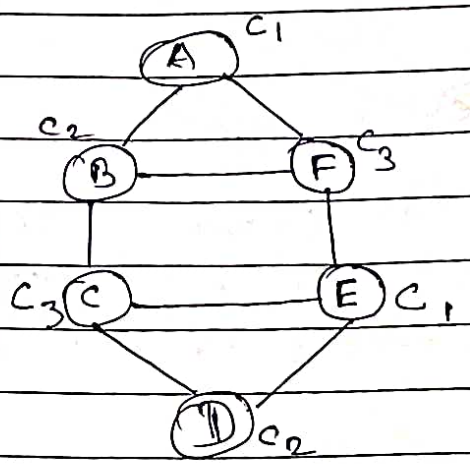
Graph coloring Problem

Proper coloring → The vertices of graph should be coloured in such a way that two adjacent vertices should not be of same color.

$\lambda(G)$ = Chromatic number

⇓

The min. no. of color used for proper coloring



Here, $\lambda(G) = 3$