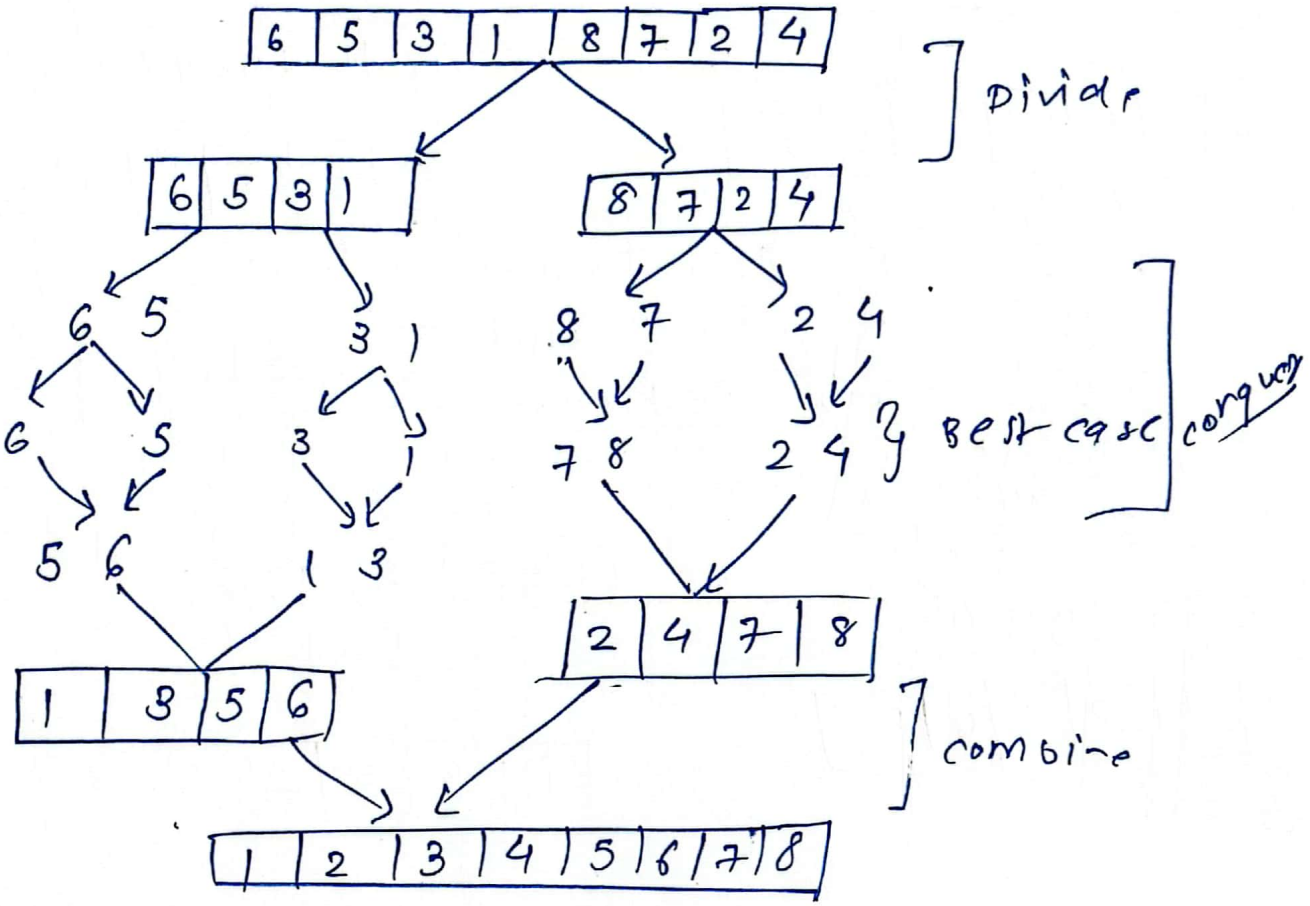
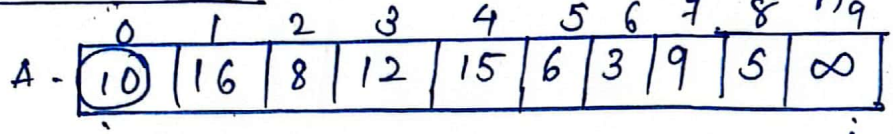
Divide and conquer :

merge sort



Quick sort :-



i => .
i => 10
j < 10

10, 16, 8, 12, 15, 6, 3, 9, 5, ∞ . . .
i i j j

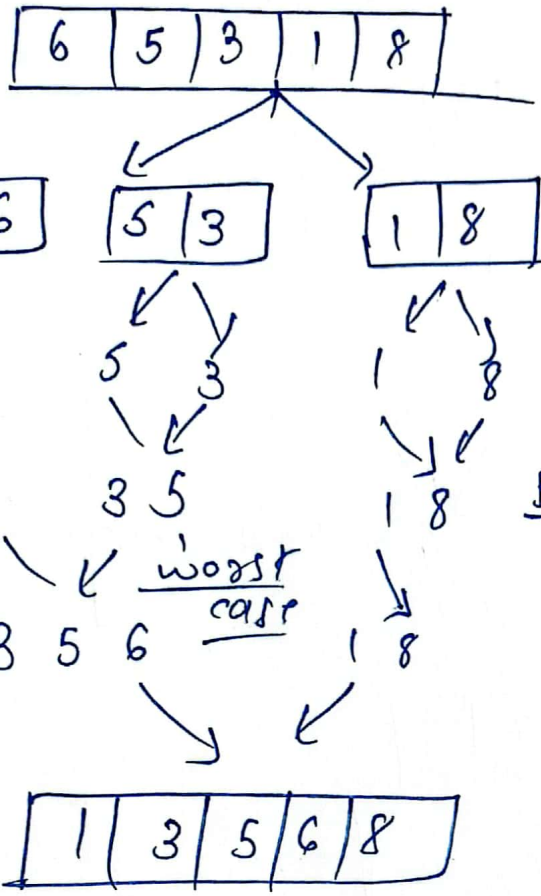
10, 5, 8, 12, 15, 6, 3, 9, 16, ∞
i j j

10, 5, 8, 9, 15, 6, 3, 12, 16
i j

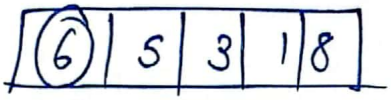
10, 5, 8, 9, 3, 6, 15, 12, 16
j i

pivot > value of pivot
pivot < value of Pivot

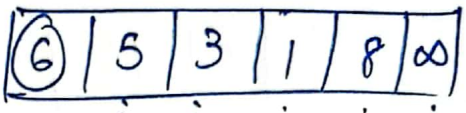
A = 6 5 3 1 8 merge sort



6 pivot :>6

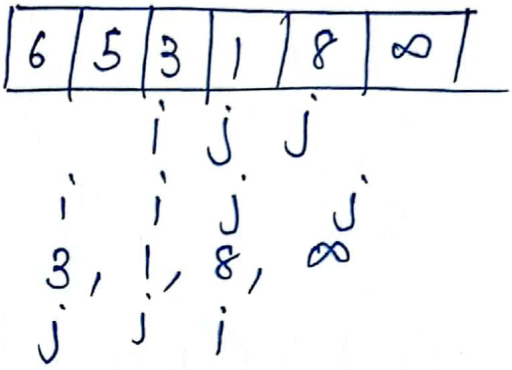


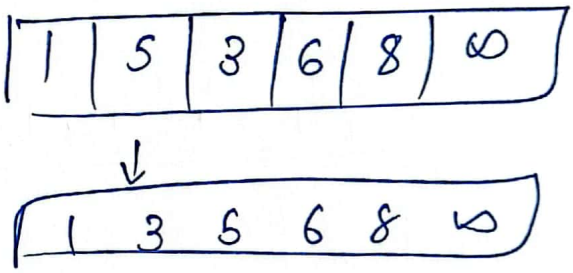
pivot element = 6



i ≥ pivot
j < pivot

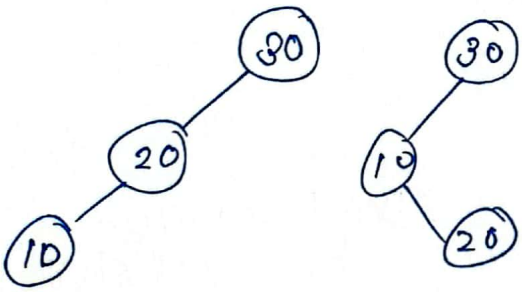
Best case





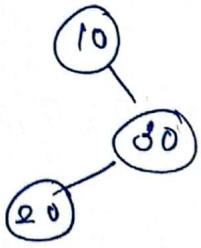
AVL Trees

keys 30, 20, 10

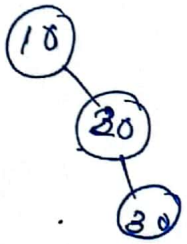


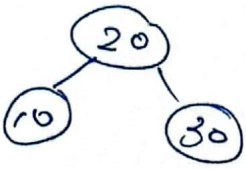
30, 10, 20

10, 30, 20



10, 20, 30





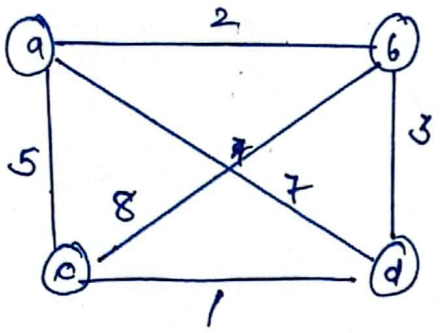
subset	Total weight	Total value
ϕ	0	0
[1]	7	42
[2]	3	12
[3]	4	40
[4]	5	25
[1,2]	10	54
[1,3]	11	Not feasible
[1,4]	12	Not feasible
[2,3]	5	52
[2,4]	8	37
[3,4]	9	52

Not feasible

Assignment problem

	Job 1	Job 2	Job 3	Job 4
person 1	9	2	7	8
person 2	6	4	3	7
person 3	5	8	1	8
person 4	7	6	9	4

Exhaustive-search



a → d → b → c → a

l = 7 + 3 + 8 + 5 = 23

a → d → c → b → a

l = 7 + 1 + 8 + 2 = 18

a → b → c → d → a

2 + 8 + 1 + 7 = 18

a → b → d → c → a

l = 2 + 3 + 1 + 5 = 11 //

optimal

a → c → b → d → a

l = 5 + 8 + 3 + 7 = 23

a → c → d → b → a

l = 5 + 1 + 3 + 2 = 11 //

optimal

Knapsack problem using Exhaustive-search

Knapsack capacity W = 10

No of items	Weight (wi)	profit (Vi)
item 1	7	42
item 2	3	12
item 3	4	40
item 4	5	25

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

$\langle 1\ 2\ 8\ 4 \rangle = 9 + 4 + 1 + 4 = 18$

$\langle 1\ 2\ 4\ 3 \rangle = 9 + 4 + 8 + 9 = 30$

$\langle 1\ 8\ 2\ 4 \rangle = 9 + 3 + 8 + 4 = 24$

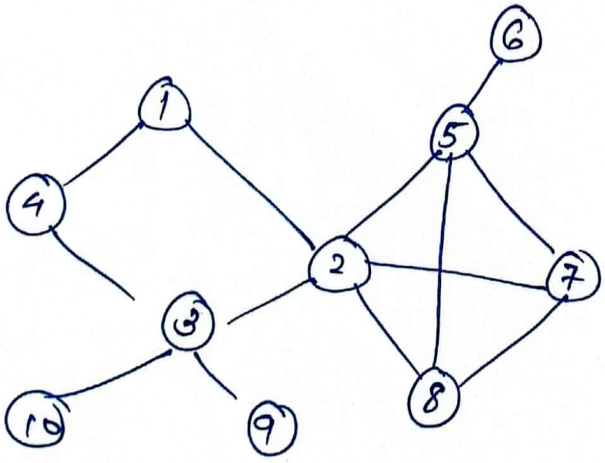
$\langle 1\ 3\ 4\ 2 \rangle = 9 + 3 + 8 + 6 = 26$

$\langle 1\ 4\ 2\ 8 \rangle = 9 + 7 + 8 + 9 = 33$ etc

$\langle 1\ 4\ 3\ 2\ 7 \rangle = 9 + 7 + 1 + 6 = 23$

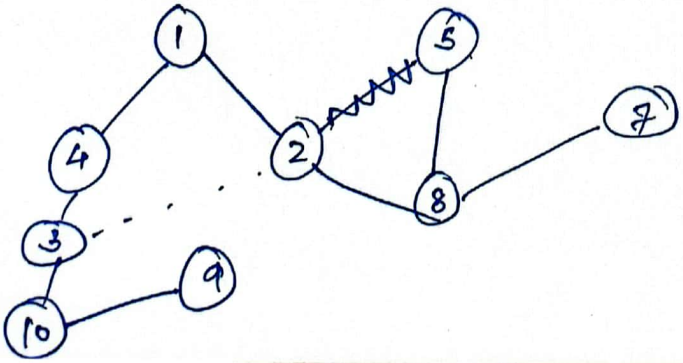
2^n complexity of time

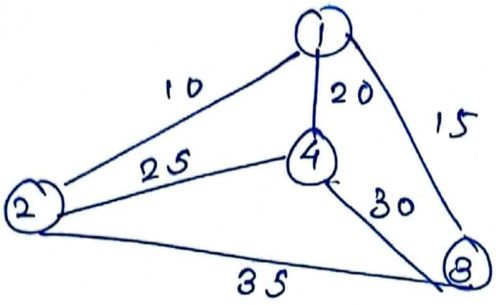
Breath First search (BFS)



Q

~~1~~
4
~~2~~
~~3~~
5
8
7
10
9





Travelling salesman problem (TSP)

$2^4 = 16$ possibl

$\langle 1\ 2\ 3\ 4 \rangle = 10 + 35 + 30$

$\langle 1\ 2\ 4\ 3 \rangle = 10 + 25 + 30$

~~$\langle 1\ 3\ 2\ 4 \rangle = 15 + 35 + 25$~~ $= 15 + 35 + 25$

$\langle 1\ 3\ 4\ 2 \rangle = 15 + 30 + 25$

$\langle 1\ 4\ 2\ 3 \rangle$ = (

$\langle 1\ 4\ 3\ 2 \rangle$ etc ✓

find optimal sol

0/1 knapsack problem using Bottom up Approach

items $n=4$ weights $w=5$

$w_1, w_2, w_3, w_4 = \{2,3,4,5\}$ $v_1, v_2, v_3, v_4 = \{3,4,5,6\}$

(i)

item	weight(w)	value
1	2	3
2	3	4
3	4	5
4	5	6

$t(i,j) = \max(T(i-1,j), (value) + T(i-1, j-(weight))$

weights → j

item \ j	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

$T(1,1) =$ $i=1$ $j=1$

$= \max(T(1-1$

$(value)_1 = (value)_1 = 3$

$(weight)_1 = (w)_1 = 2$

$T(1,1) = \max\{T(i-1, j), (value)_1 + T(i-1, j-(w)_1)\}$

$T(1,1) = \max\{T(0,2), 3 + T(0,1-2)\}$

$\max\{T(0,1), 3 + T(0,-1)\}$

6

Step 2 :. $i=1, j=2$

$(value)_2 = 4$

$(weight)_2 = 3$

$T(1,2) =$

$\max(T(i-1,j), (value)_1 + T(i-1, j-w_i)$

$\max(T(1-1,2), 3), + T(1-1, 3-3)$

3

Assignment problem

person \ Jobs	J1	J2	J3	J4
a	9	2	7	8
b	6	4	3	7
c	5	8	1	8
d	7	6	9	4

L·B = 2+3+1+4 = 10

Lets do by Row-Reduction Method!

L·B = 2+3+1+4 = 10

a=1 | a=2 | a=3 | a=4

9+4+1+4 = 18 | 2+ | 7+3+1+ | 8+3+1+6, L·B = 18

2+3+1+4 | 7+4+5+9

9+3+1+4, L·B = 17 | L·B = 10 | L·B = 20 | a=2

b=1 | b=3 | b=4 | b=1

6+2+1+4 | 3+2+5+4 | 7+2+1+6

L·B = 13 | L·B = 19 | L·B = 17

c=3 | c=4

1+2+4
+4
1

$T(1,3) = [T(i-1, j), (value)_i + T(i-1, j-(weight)_i]$

$T(1,3) = T(0,3), (value)_1 + T(0, 3-(weight)_i]$

$(value)_1 = 3$

$(weight)_1 = 4$

$= T(0,3), 3 + T(0, 3-4)$

$= T(0,3), 3 + T(0, -1)$

$\max\{T(0,3), 3 + T(0,1)\}$

$0, 3+0$

$T(1,4) = T($

	3
$(value)_i$	$weight_i$
3	2

$= 3.$

$T(1,4) = T(1-1, 4), 3 + T(0, 4-2)$

$i \; j$

$\max(T(0,4), 3 + T(0,2))$

$0, 3+0$

$i=1 \; j=5$

$T(1,4) = 3$

$T(1,5) = T(1-1, 5), 3 + T(0, 5-2)$

$T(0,5), 3 + T(0,3)$

$0, 3+0$

$(value)_2 = 4 \quad (weight)_2 = 3$

$i=2 \; j=1$

$T(2,1) = T(2-1, 1), 4 + T(2-1, 1-3)$

$T(1,1), 4 + T(1, -2)$

$\max\{0, 4 + T(1,-2)\}$ ignore

$T(2,2)$ $\boxed{i=2 \;\; j=2}$ $\boxed{\begin{array}{l}(value)_2 = 4 \\ (weight)_2 = 3\end{array}}$

~~$= max(2$~~

$max(2-1, 2) + 4 + T(2-1, 2-3)$ ignore

$max(1,2), \; 4 + T(1, -1)$

$max(3, 4 + T(1,-1))$

$= 3$ $\boxed{i=2, j=3}$

$T(2,3) = max(2-1, 3) + 4 + T(2-1, 3-3)$

$4 + (1, 0)$

$3, \; 0 + 4 + 0$

$max(3,4) = 4$

maximum weight that we can put in knapsack = 7

Sum of subset

$S = (10, 20, 30, 40)$ — $2 \times 2 \times 2 \times 2$ — (2^n)

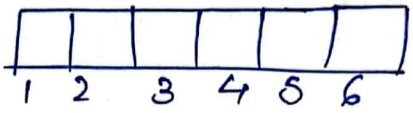
$SOS(n,m) = \begin{cases} SOS(n-1,m) \\ \text{OR} \\ SOS(n-1,wn) \end{cases}$ — True $n=0, m=0$; False $n=0, m \neq 0$; True $n \neq 0, m=0$

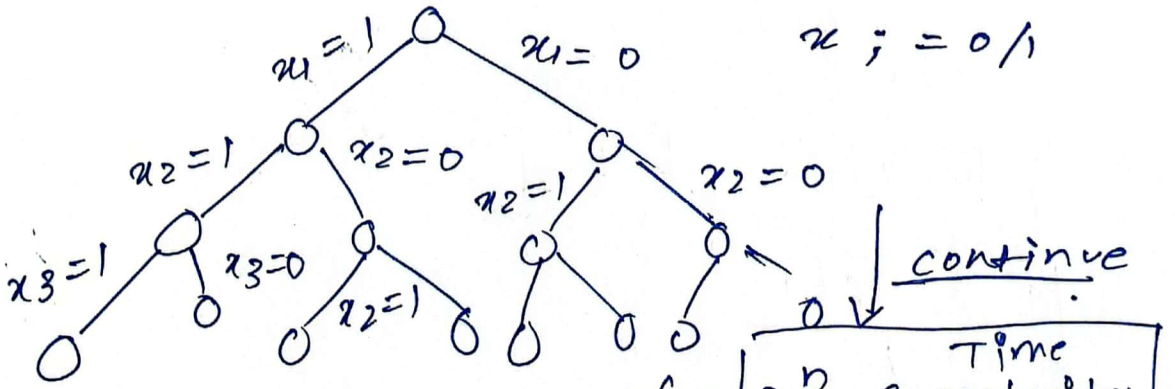
$SOS(n-1,m)$ $wm > m$

n m

$w[1:6] = \{5, 10, 12, 13, 15, 18\}$

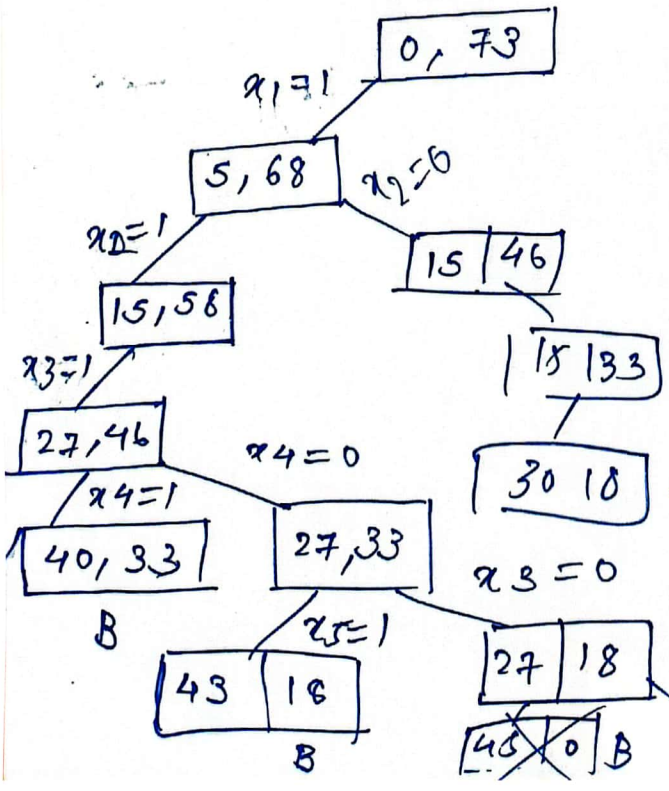
$n=6$ $m=30$





$x_i = 0/1$

$2^6 =$ Time complexity 2^n

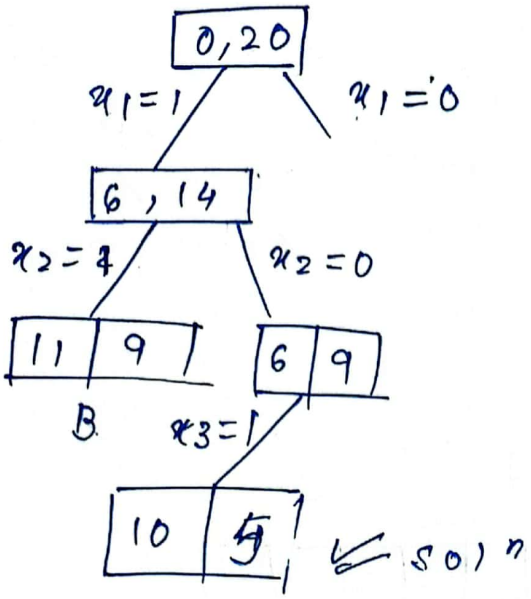


$$\sum_{i=1}^{k} w_i x_i + w_{k+1} \leq M$$

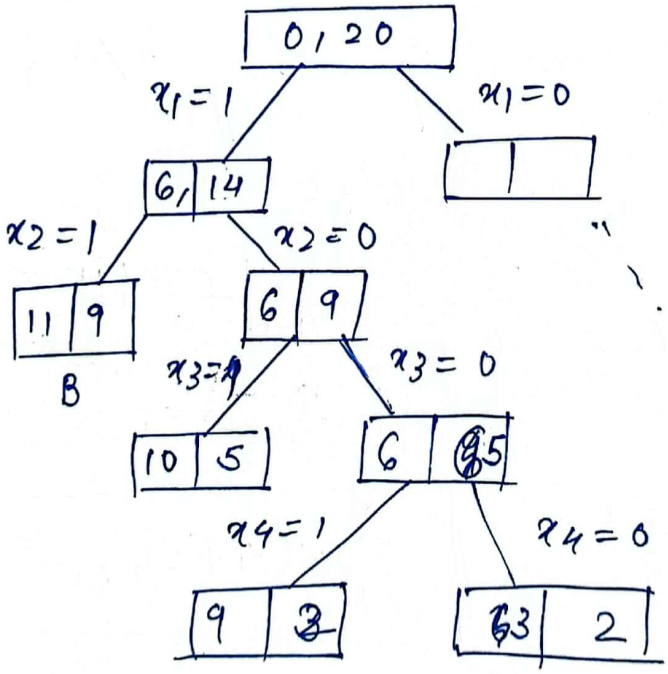
$$\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1} w_i > m$$

$A = \{6, 5, 4, 3, 2\}$ and $d = 10$

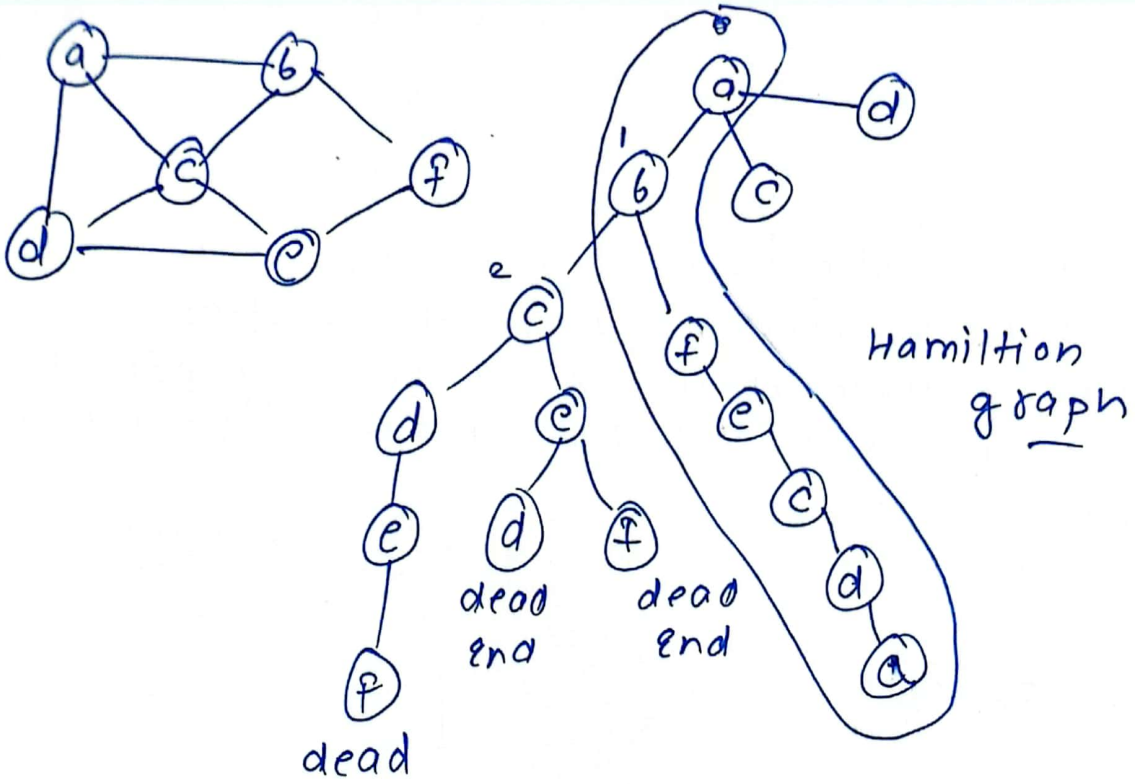
6	5	4	3	2
x_1	x_2	x_3	x_4	x_5

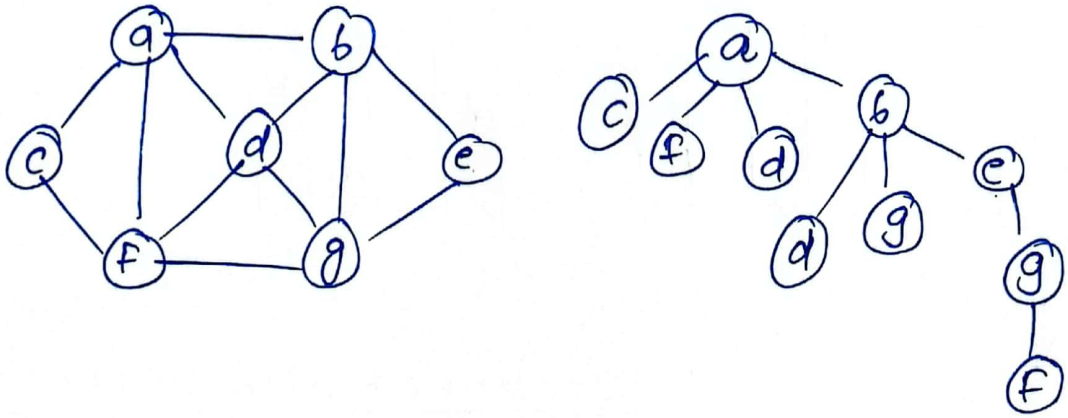


← soln



continue Building to find all solutions !!

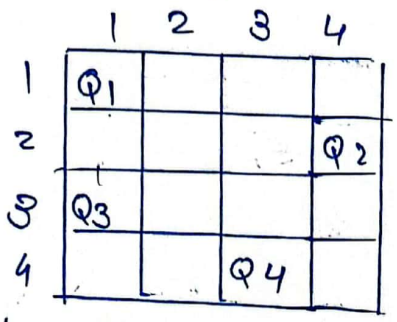

a
b
c
d
e
f
dead
dead end
dead end
Hamiltion graph


a
b
c
d
e
f
g

Chapter 12 : Coping with Limitation of Algorithm

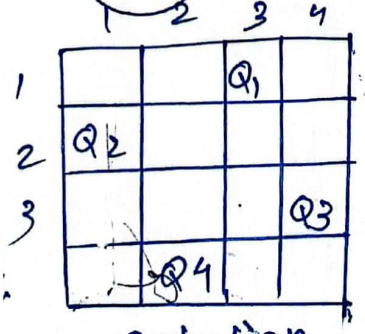
A. Backtracking :-

n-Queen's problem :- N queen problem is the classical example of backtracking N-Queen problem is defined as "given N x N chess board. N queens in such a way that NO two queens attack each other



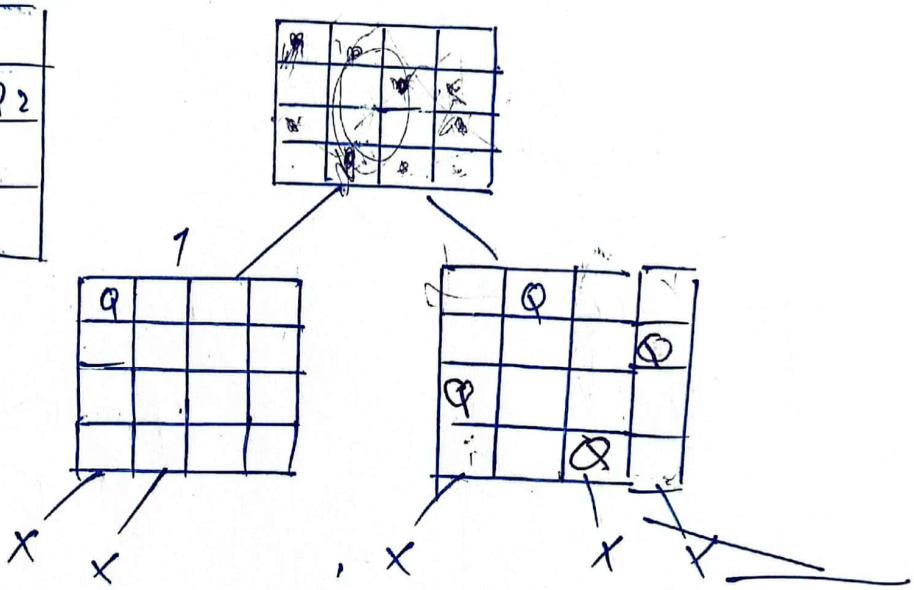
solution

OR



solution

Build a **state space tree**



continue with rest graph

Hamiltonian circuit

Hamiltonian Graph

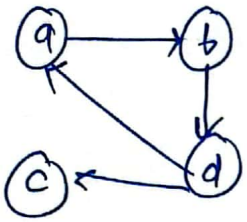
problem is about or concerned finding Hamiltonian circuit in a given graph

Hamiltonian ckt :- defined as a cycle that passes through all vertices of a graph exactly once except starting and ending vertices

chp II Dynammic programming

Warshall's and Floyd's Algorithm :-

Warshall's Algorithm is used to find the transitive closure of a directed graph

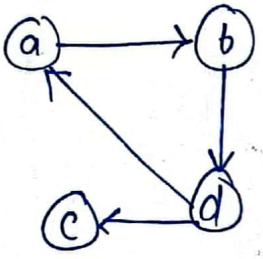


Transitive closure

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	0	0	0	0
d	1	1	1	1

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	0	0	0	0
d	1	1	1	1

Dynammic-programming Approach



	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

$\Rightarrow R^0$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	0	0

$R^2 =$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

$R^2 =$

	0	1	0	1
	0	0	0	1
	0	0	0	0
	1	1	1	1

$R^3 =$

	0	1	0	1
	0	0	0	1
	0	0	0	0
	1	1	1	1

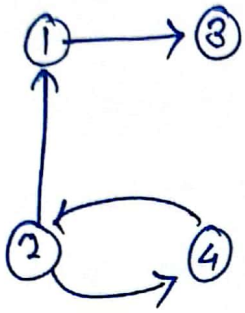
$\Rightarrow$

0	1	0	1
1	1	0	1
0	0	0	0
1	1	0	1

$R^4 =$

1	1	1	1
1	1	1	1
0	0	0	0
1	1	1	1

//



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

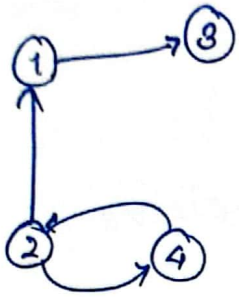
Warshall's Algorithm

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$R(p) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$R(1) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad R(3) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

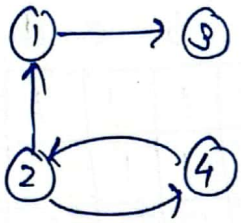
$$R(4) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & & \\ & & & \\ & & & \end{bmatrix}$$



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

← Required Transitive matrix

Apply Warshalls Algorithm



construct Adjacency matrix

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

$R^{(0)} =$

0	0	1	0
1	0	0	1
0	0	0	0
0	1	0	0

$\Rightarrow$

0	0	1	0
1	0	1	1
0	1	0	1
0	1	0	1

$R^{(1)} =$

0	0	1	0
1	0	1	1
0	1	0	1
0	1	0	1

$\Rightarrow$

0	0	1	0
1	0	1	1
1	1	1	1
1	1	1	1

$=$

0	0	1	0
1	0	1	1
1	1	1	1
1	1	1	1

$=$

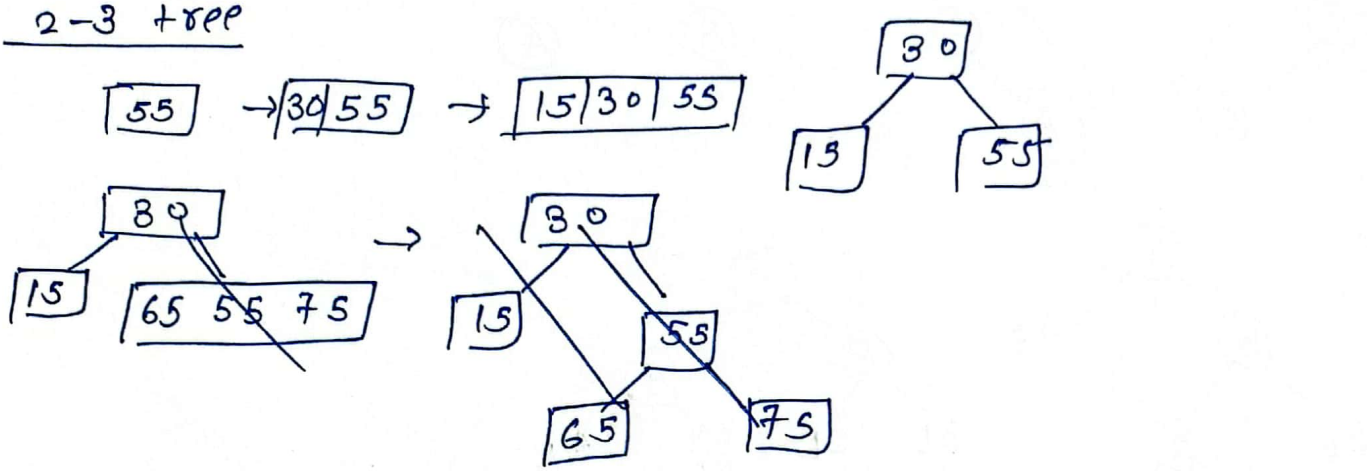
0	0	1	0
1	0	1	1
1	1	1	1
1	1	1	1

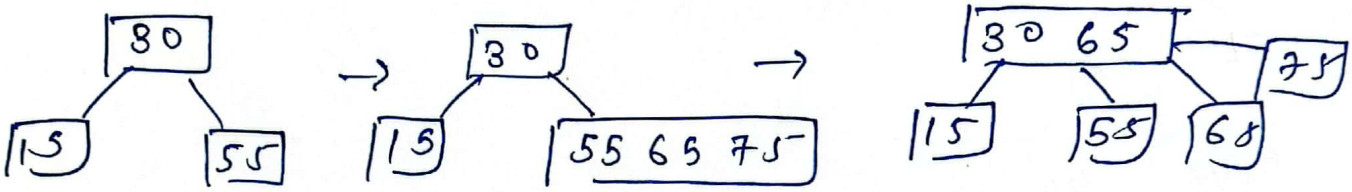
$=$

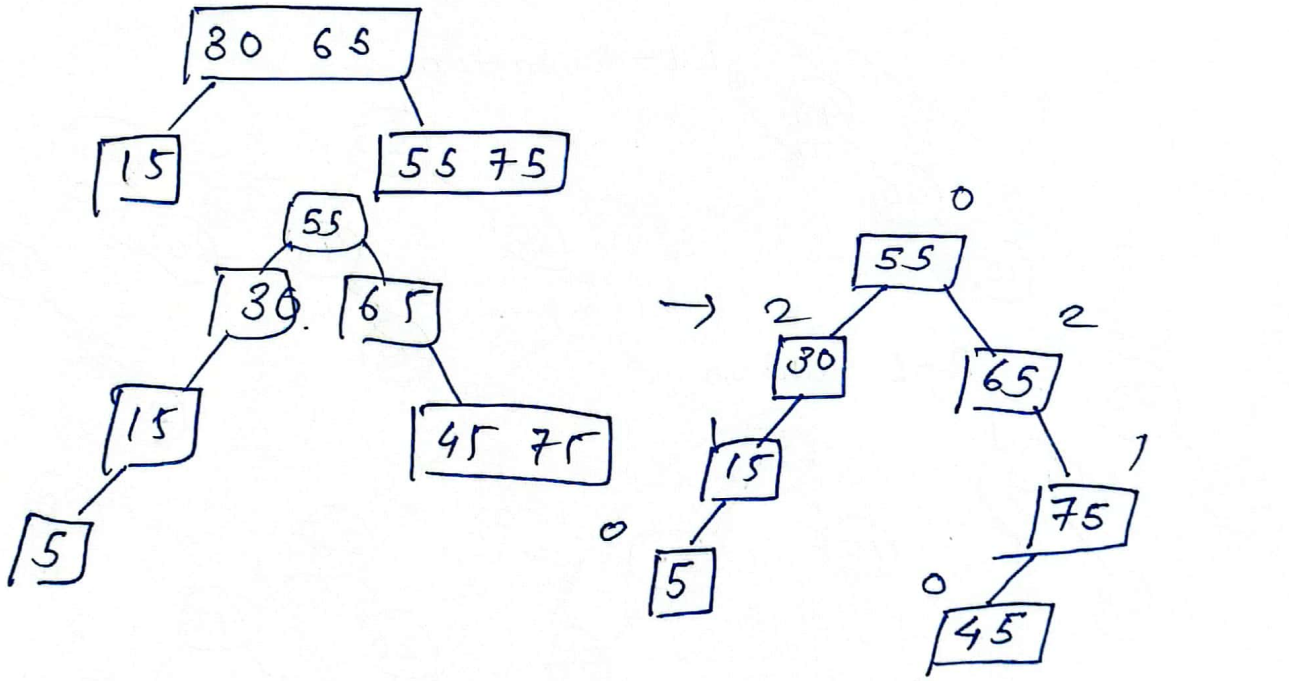
construction of 2-3 trees

55, 30, 15, 75, 65, 45, 5

2-3 tree





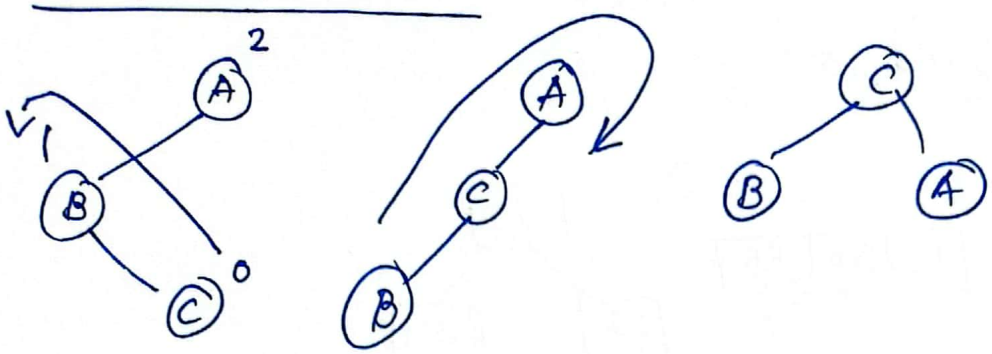


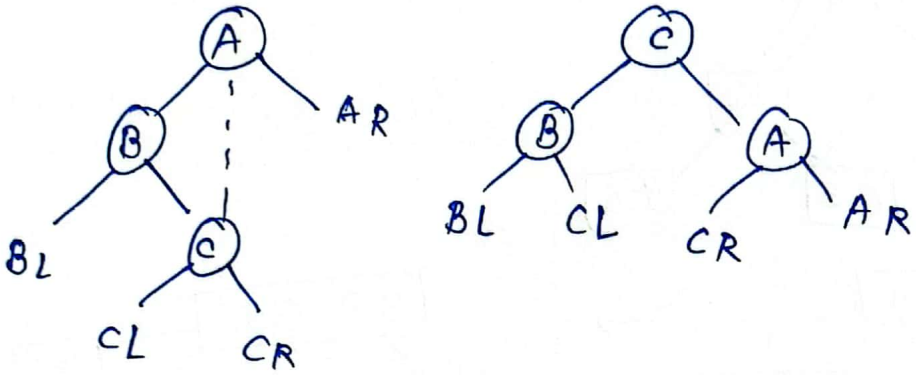
~~AVL~~ Tree

2,3 tree

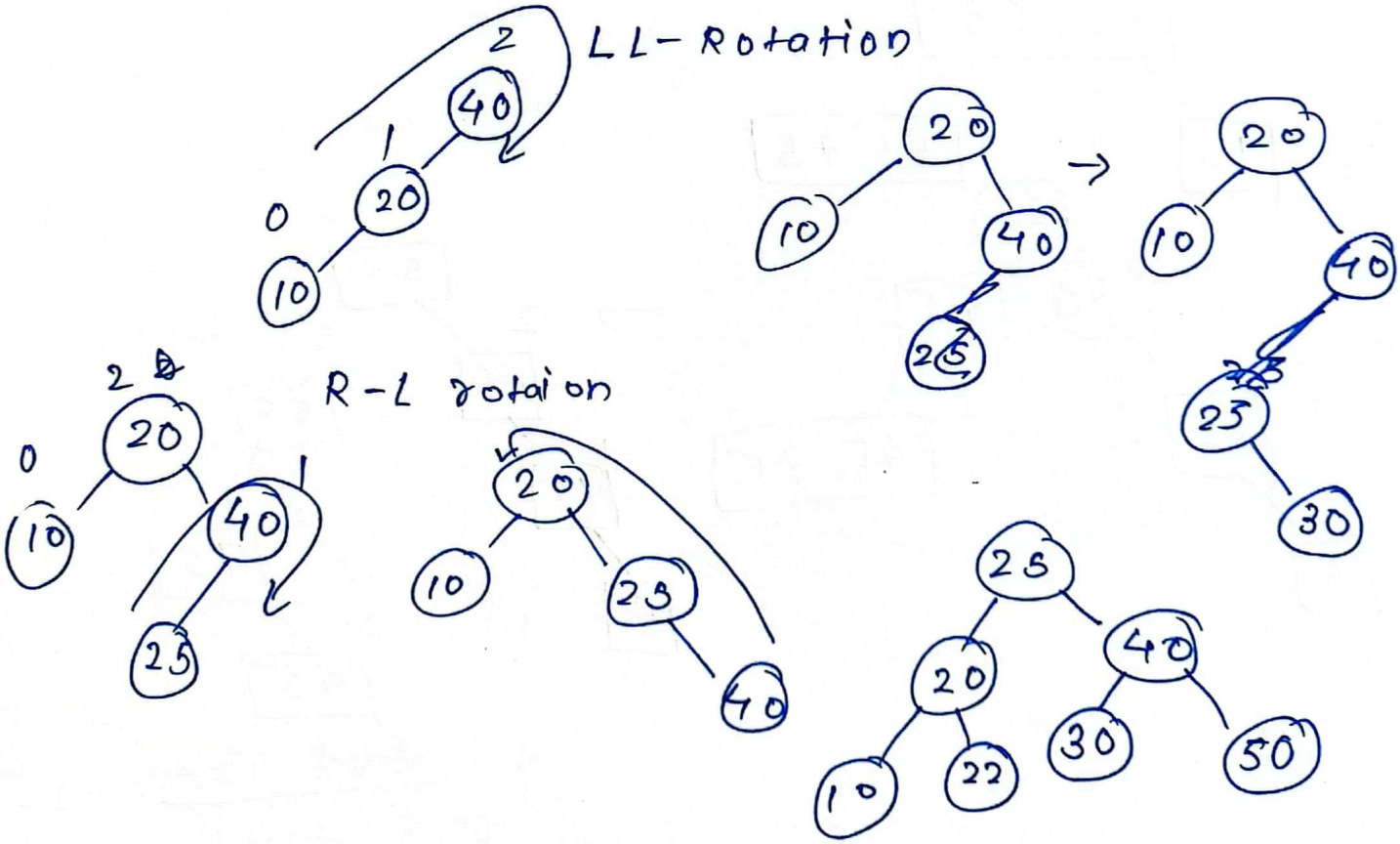
Balanced tree

L-R Rotations

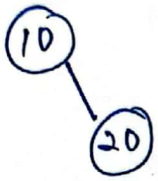




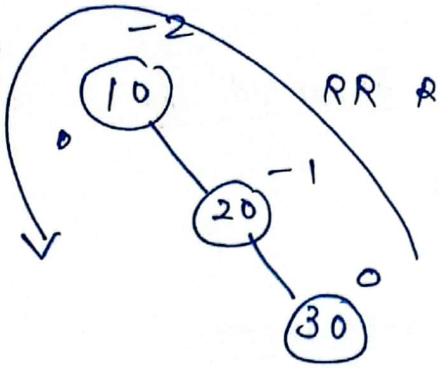
Keys 40, 20, 10, 25, 30, 22, 50



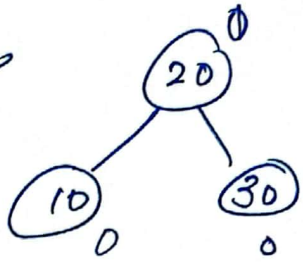
Initially



Insert 30



RR Rotate

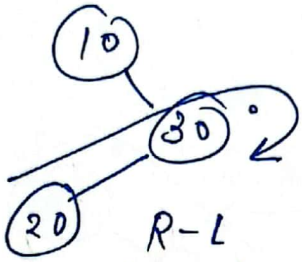


RR Rotation

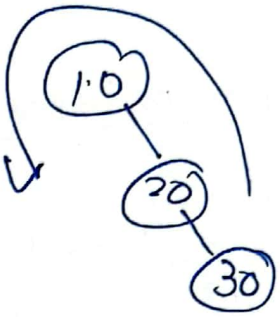
RR-Imbalance

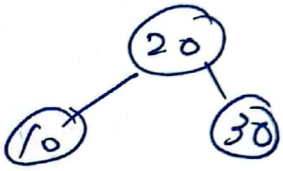
Initially

Insert 20



R-L imbalance

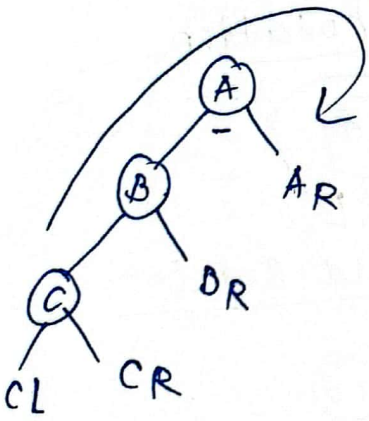




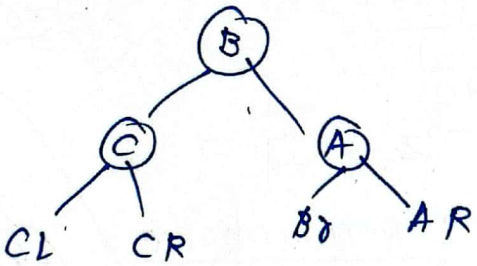
R R
L L] single Rotations

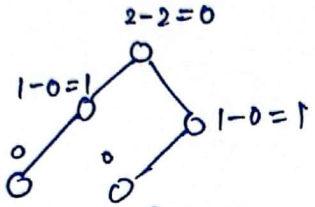
LR
RL] Double Rotations

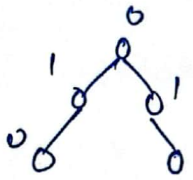
AVL- TREES



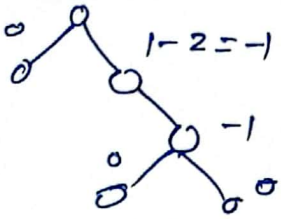
LL - Rotation Formula



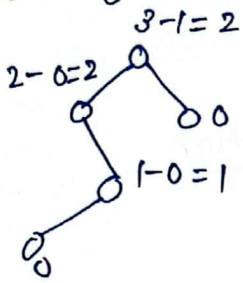


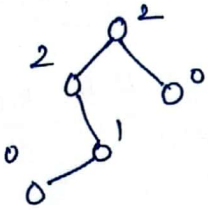


Balanced tree



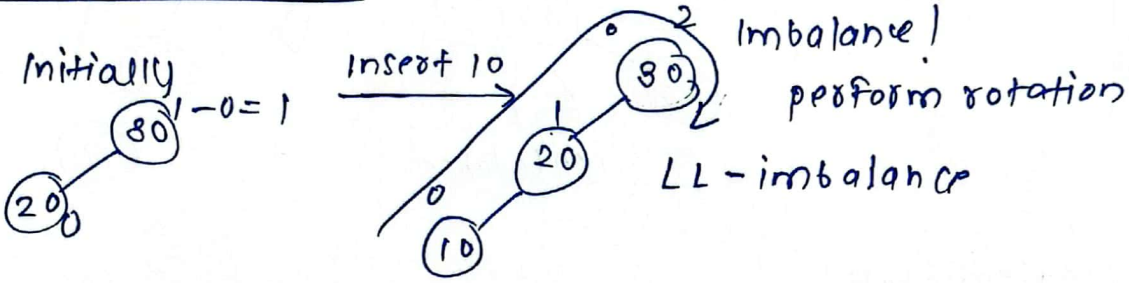
Imbalanced tree





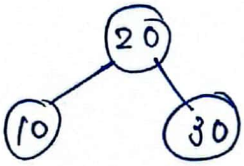
Imbalanced tree

Rotation of trees



(1)

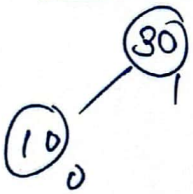
After rotation



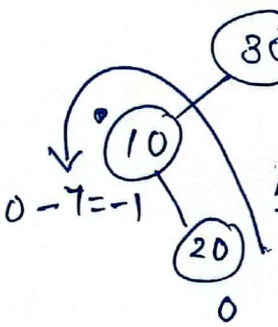
LL-Rotation

two-step rotations - LR Rotation

Initially

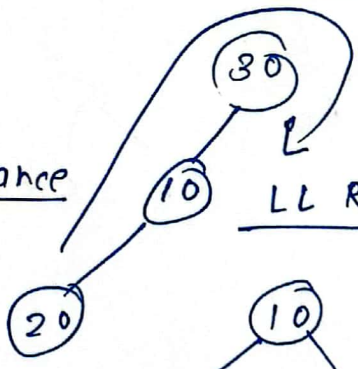


Insert 20

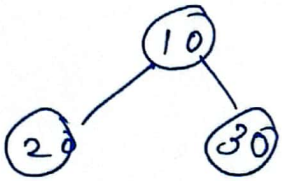


L-R imbalance

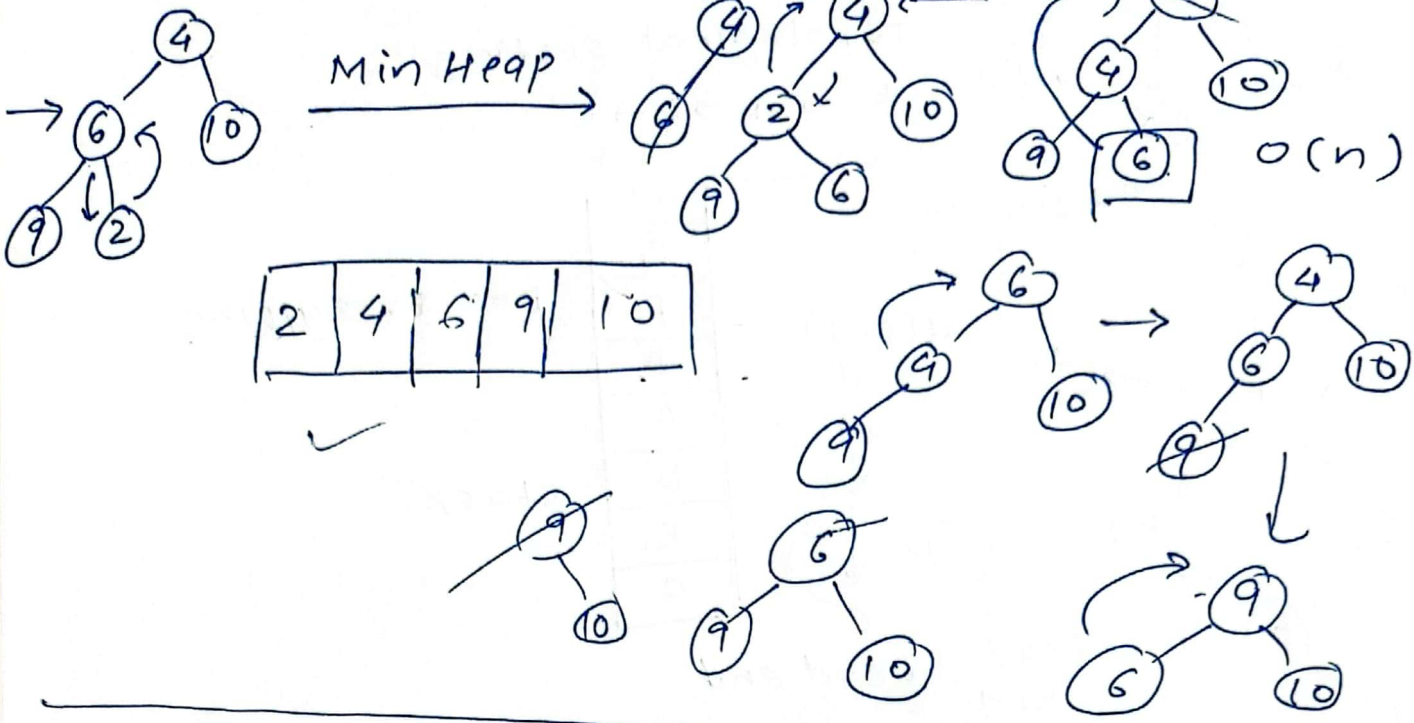
After rotation



LL Rotation

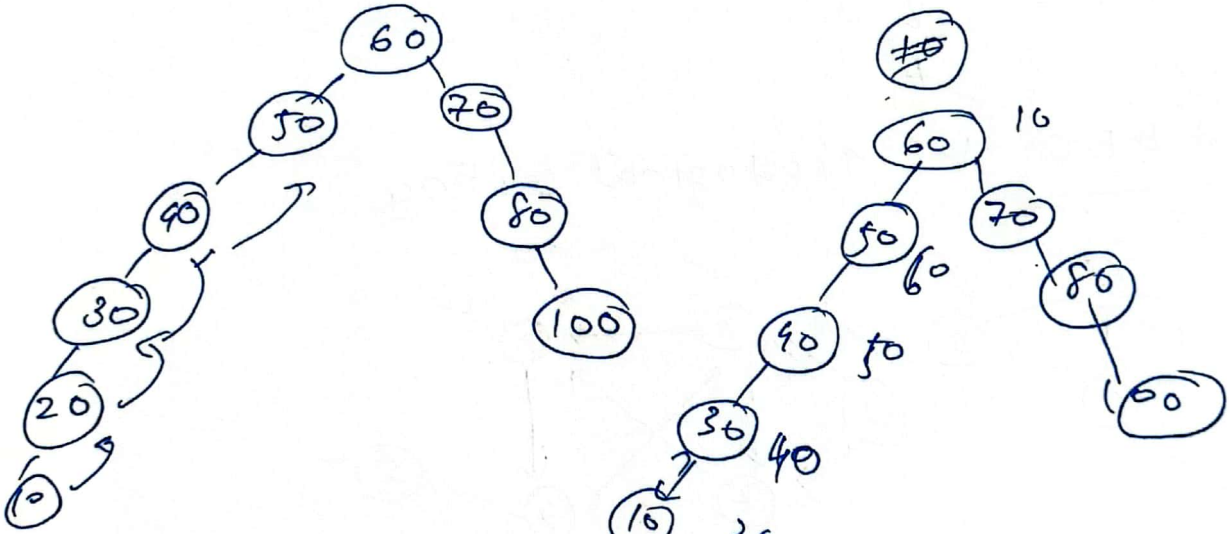


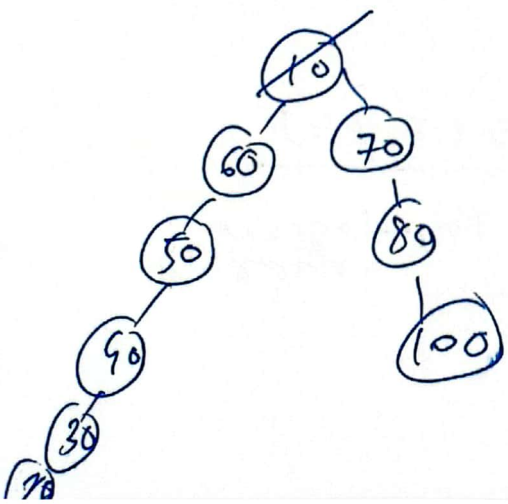
Heap sort

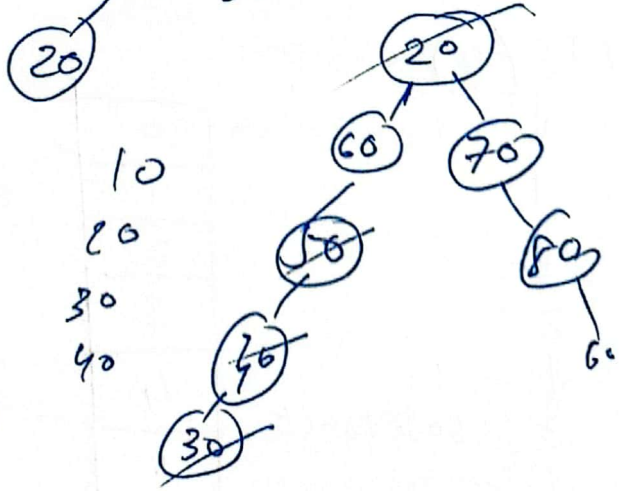


2	4	6	9	10

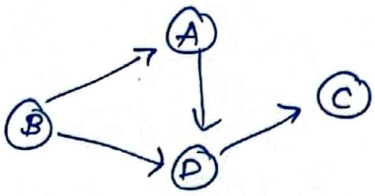
50, 60, 70, 40, 30, 20, 10, 80, 100







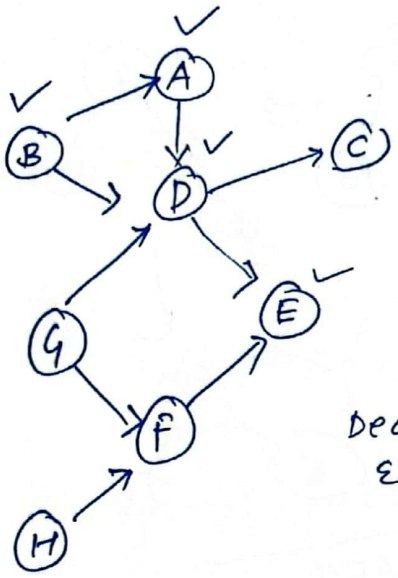
Topological Sorting



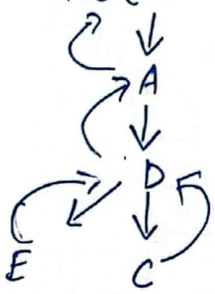
Topological sorting

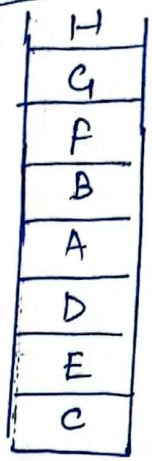
B → A → D → C

BADC ✓



dFS(B)





pop everything

stack

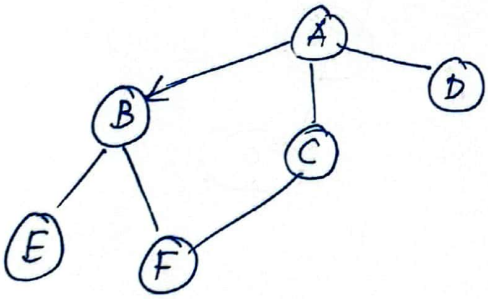
Dead End

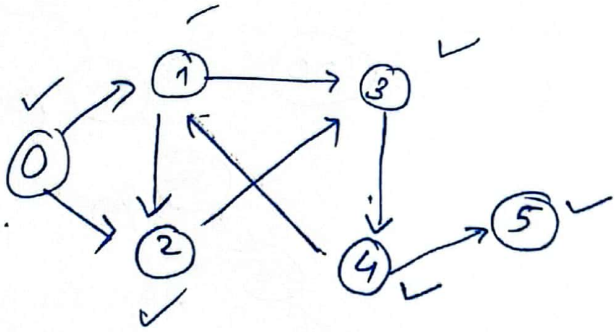
Dead end

dfs(G) dFS(H)

F

HGFBADEC ✓ Topological sorting

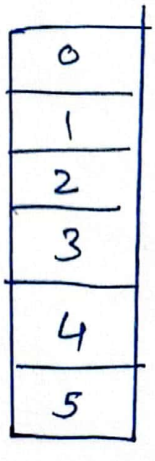




DFS(0)

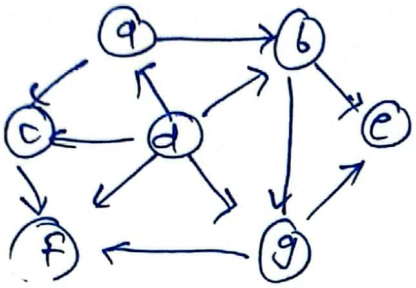
0 → 1 → 2 → 3 → 4 → 5

Backtrack

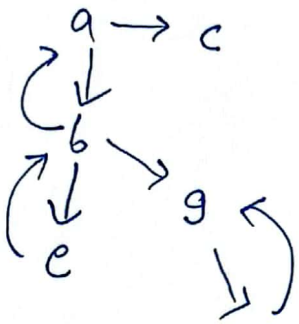


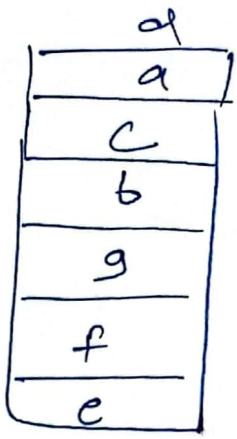
0 1 2 3 4 5

Topological order

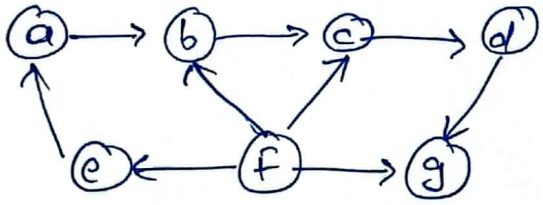


dfs (d)



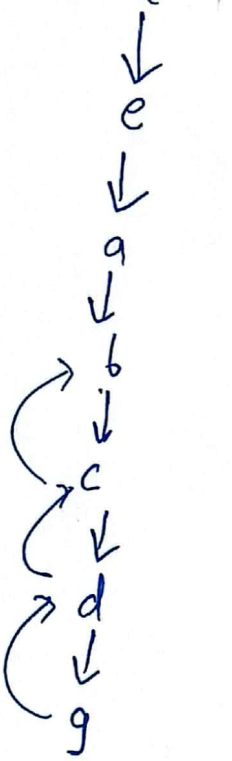


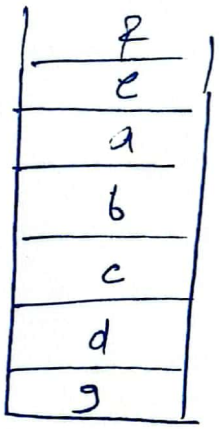
d a c b g f e



f e a b c d g Topological sequence

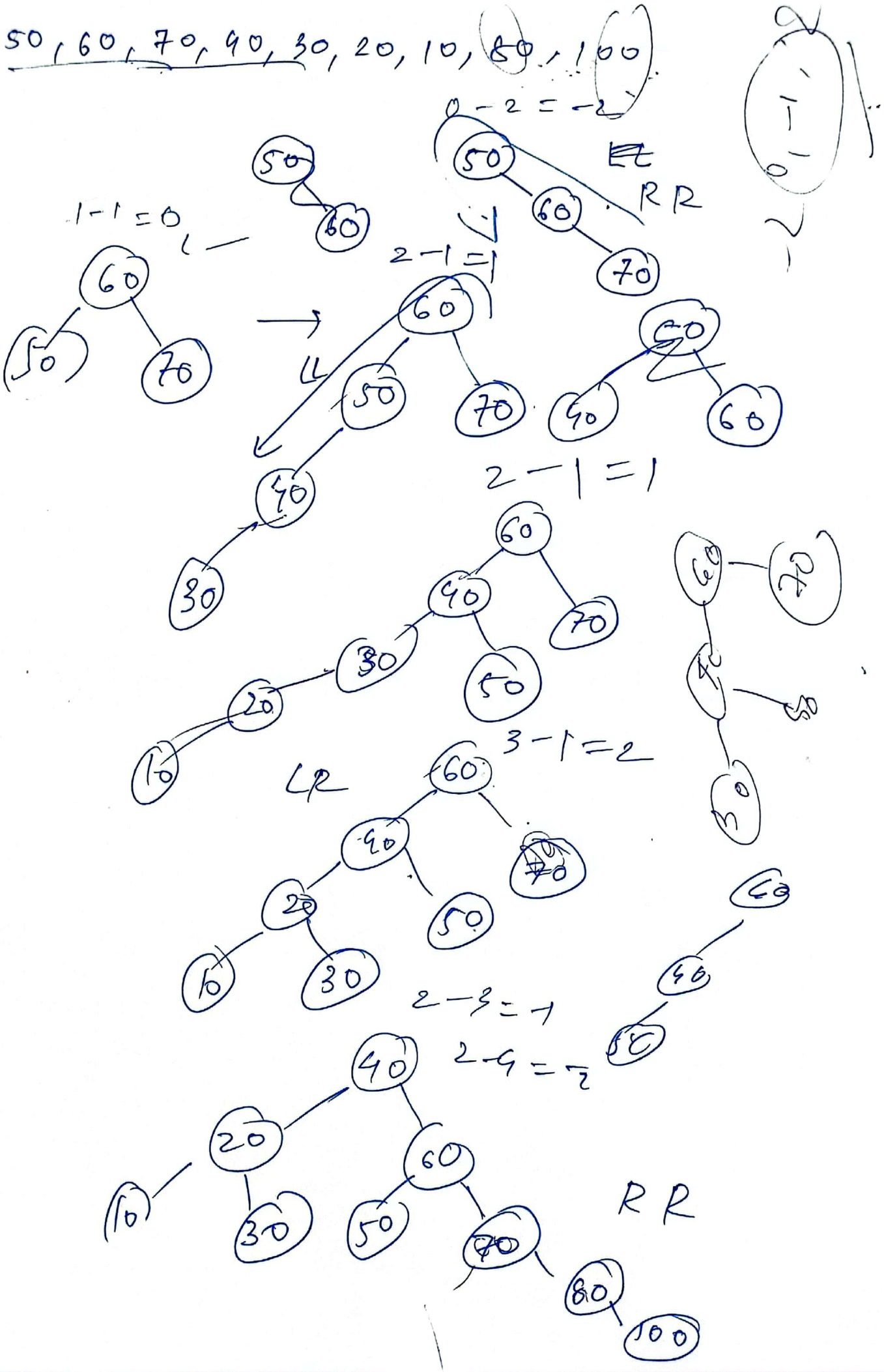
dfs (f)

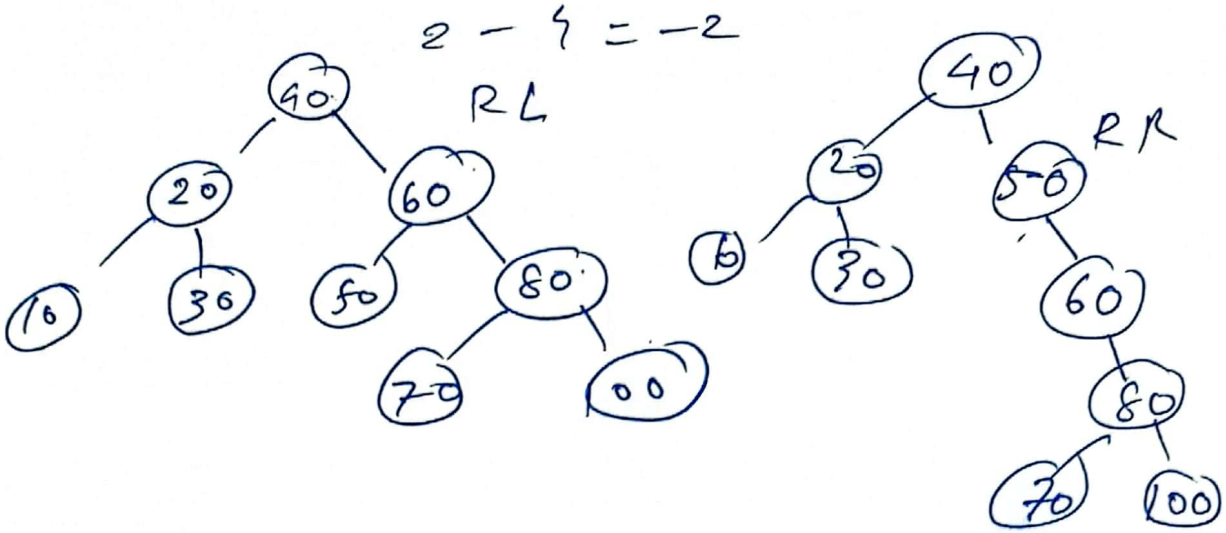


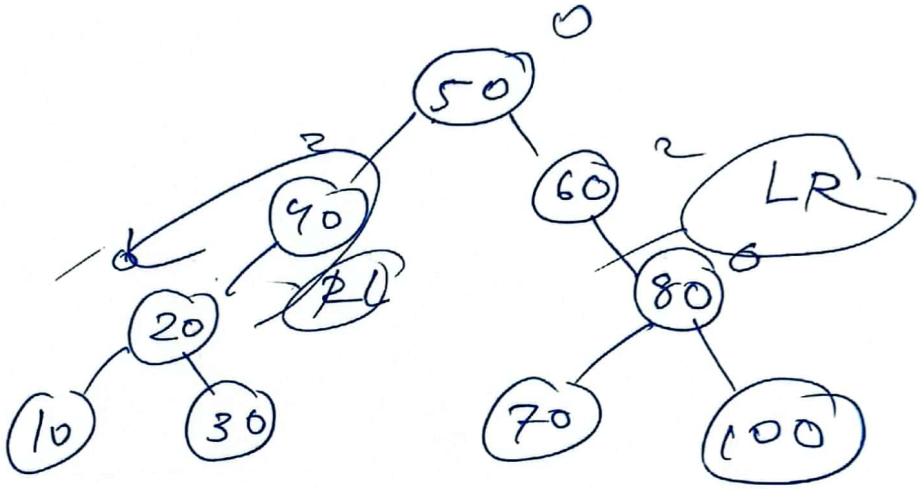


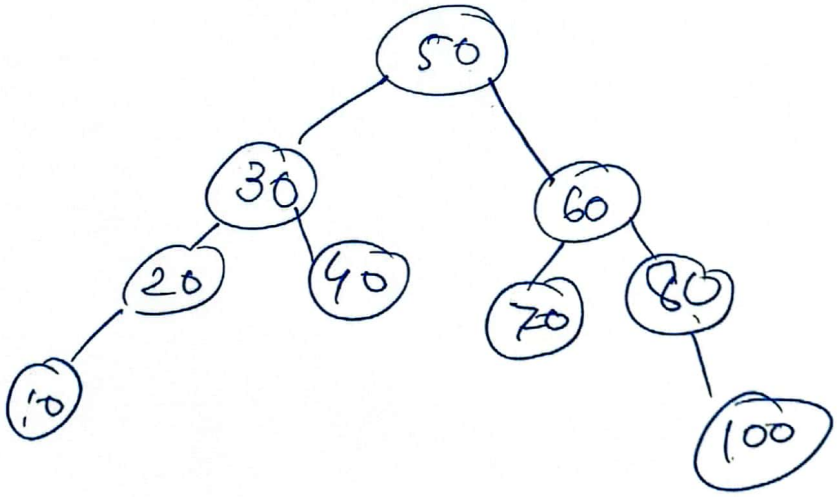
stack

50, 60, 70, 40, 30, 20, 10, 80, 100

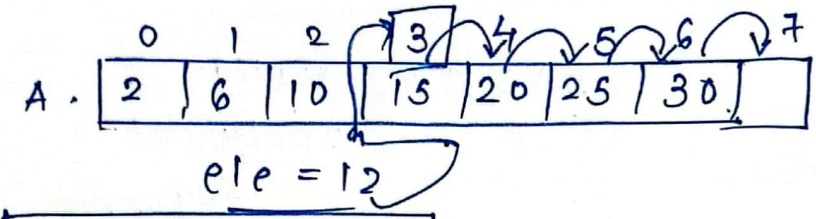



2 - 4 = -2
RL
RR
40
20
60
10
30
50
80
70
100
40
20
50
10
30
60
80
70
100


50
2
60
2
40
LR
RL
20
80
10
30
70
100


50
30
60
20
40
70
80
10
100

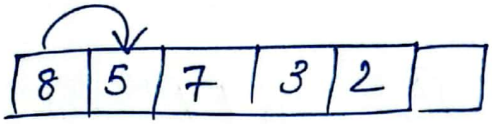
Chp 3 :- Decease and conquer

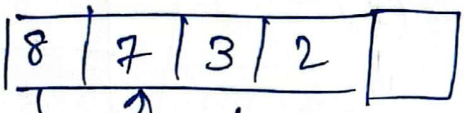


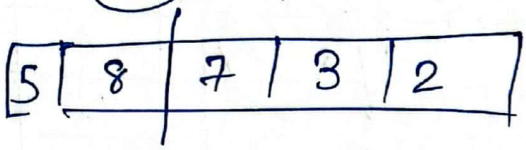
Algorithms

Insertion sort

i) Insert 5 in

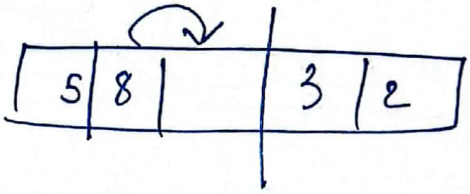






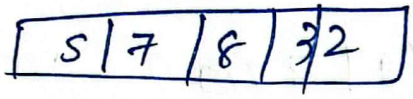
1st pass

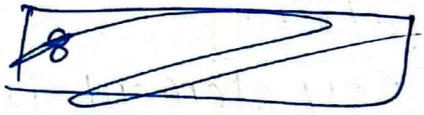
Insert 7

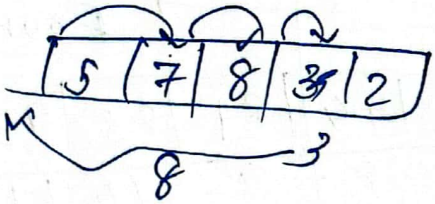


7

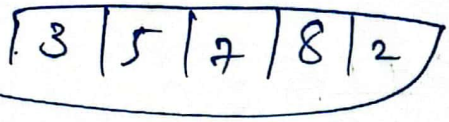
II nd pass

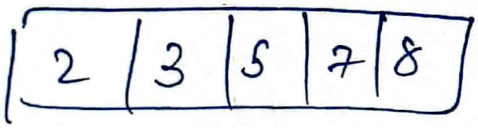




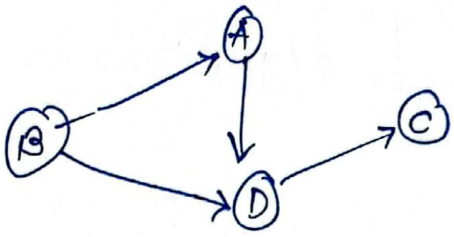


III rd pass

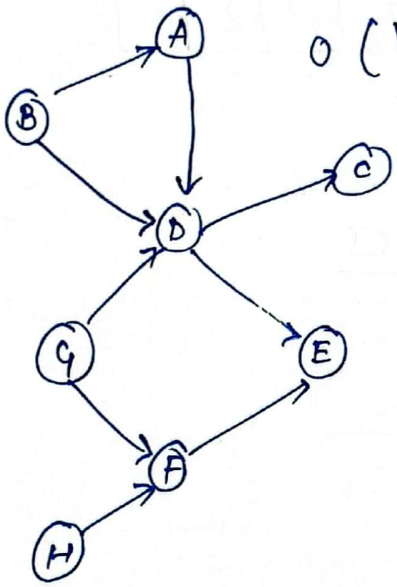




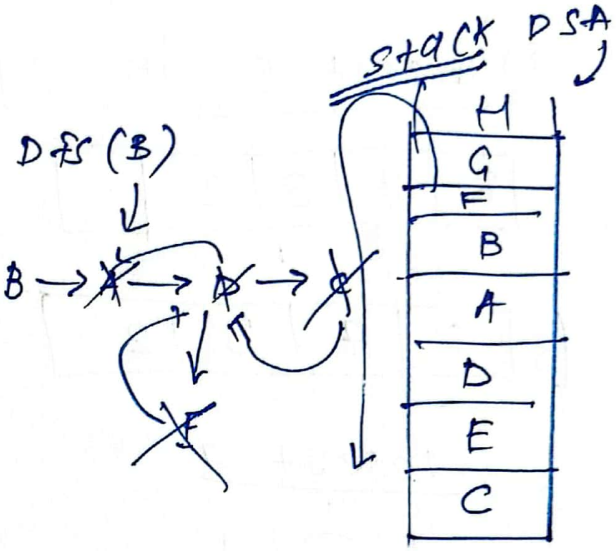
Topological order



BADC ✓ valid Topological sequence



O(V+E)



BADEC ✓

DFS(G) → G → F →

DFS(A)

81 > 92 yes

A[9] ≠ 70 ? ∴ mid-1

A[7] ≠ 70

$\frac{6+9}{2} = \frac{15}{2} = 7$

Key element is found

Binary searching :-

3	14	27	31	39	42	55	70	74	81	85	93	98
0	1	2	3	4	5	6	7	8	9	10	11	12
						m						

$mid = \left(\frac{start + end}{2}\right) = \frac{0+12}{2} = 6$

mid > 70 ?

A[6] > 70 ; NO

$\frac{6+12}{2} = \frac{18}{2} = 9$

sort using insertion sort

A = | 34 | 8 | 64 | 51 | 32 | 21 |

| 34 | 64 | 51 | 32 | 21 | |

8

| 8 | 34 | 64 | 51 | 32 | 21 | |

1st pass

| 34 | 64 | 51 | 32 | 21 |

insert 8

| 8 | 34 | 64 | 51 | 32 | 2 | |

end pass

| 8 | 34 | 51 | 64 | 32 | 2 |

3rd pass

| 8 | 34 | 51 | 32 | 2 | 64 |

4th pass

| 8 | 34 | 32 | 2 | 51 | 64 |

5th pass

| 8 | 32 | 34 | 2 | 51 | 64 |

| 8 | 32 | 2 | 34 | 51 | 64 |

A[3] ≥ 64 NO

A[2] ≥ 51 NO

| 2 | 8 | 32 | 34 | 51 | 64 |

0 1 2 3 4 5

key = 64

result

$\frac{3+5}{2} = \frac{8}{2} = 4$

$mid = \frac{2+5}{2} = \frac{7}{2}$

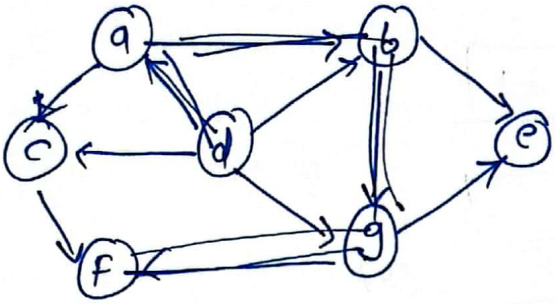
$\frac{4+5}{2} = \frac{9}{2} = 4.5 = 4$

key element found

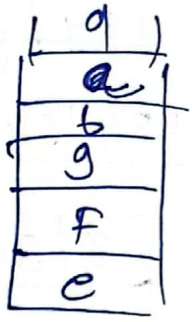
$\frac{0+1}{2} = 2.5 = 2$

$mid = \frac{start + end}{2}$

A[2] > 64 NO

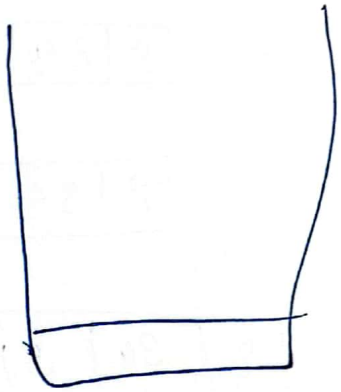


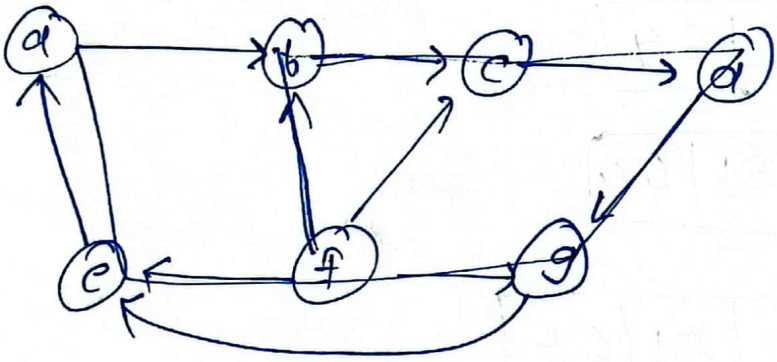
c - f

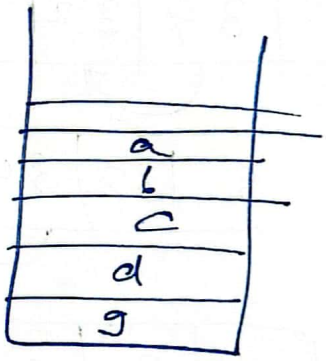


a → b → e
c
c ~~g~~
f ~~f~~

d a b g f e



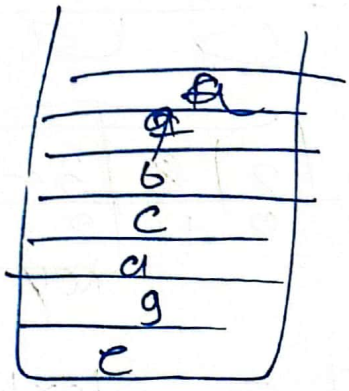




a → b → c → d → g → e

~~e — d~~.

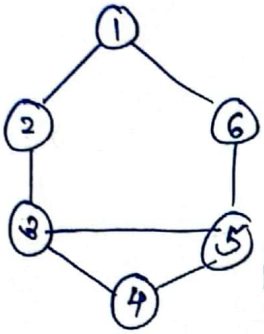
e - a - b - c - d - g



f → a → b → c → d → g → e

Greedy technique

Prims Algorithm



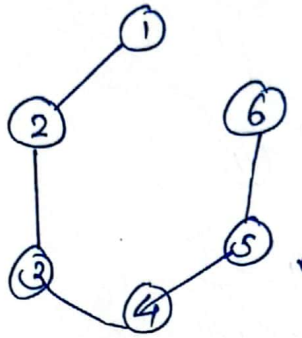
$G = (V, E)$

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

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

if ?

Minimum cost spanning tree (MST)



$|V| = n = 6$

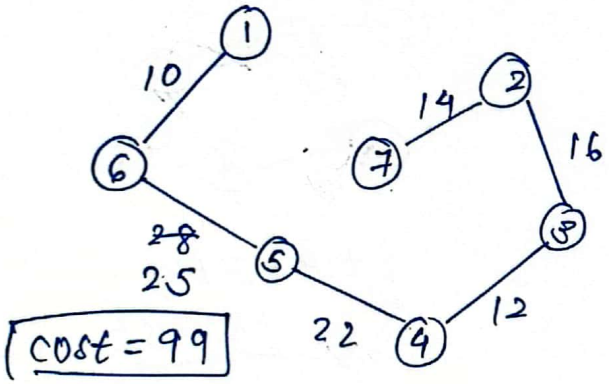
$|V| - 1 = 5$ edges

$|E| = |V| - 1$

$^6C_5$ = 6 different spanning trees

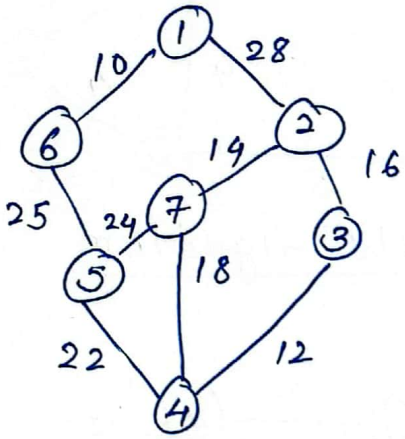
$^7C_5$

Minimum cost spanning tree using prim's Algorithm



cost = 99

most

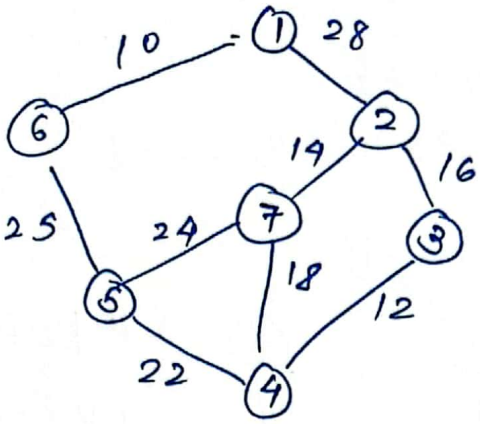


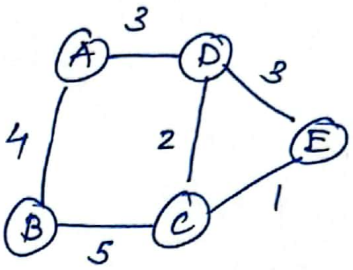
step 1 :- connect, minimum edge possible in graph

step 2 :- keep comparing and connecting smallest possible edge for a tree and remember $|E| = |V| - 1$ to satisfy cond tree.

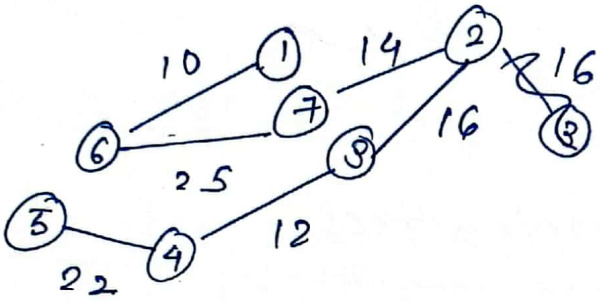
Krushkal's Algorithm

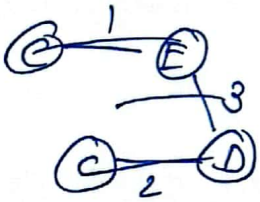
step 1 :-

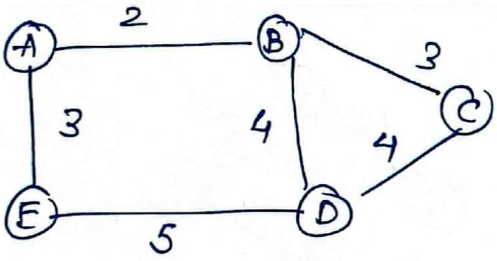




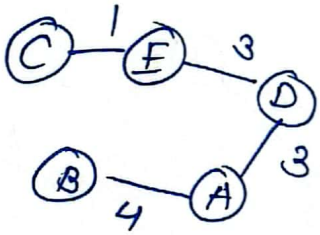
MST using prims





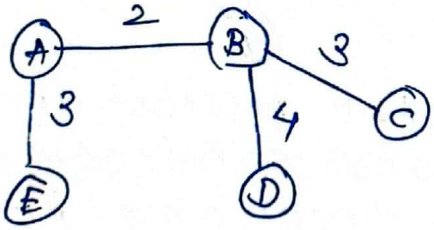


Krushkal's Algorithm



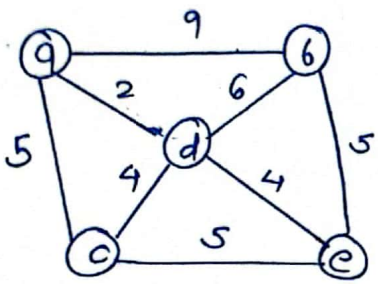
1 + 3 + 3 + 4

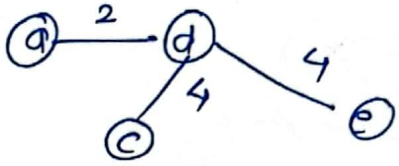
11 cost

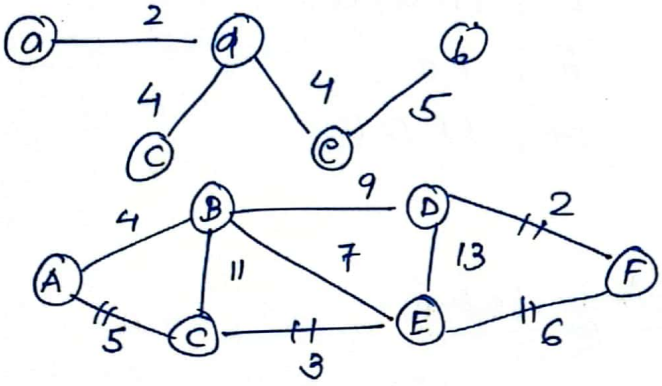


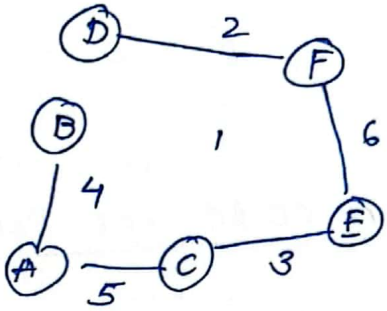
2 + 3 + 4 + 3 = 12 cost

minimum spanning tree using prims algorithm





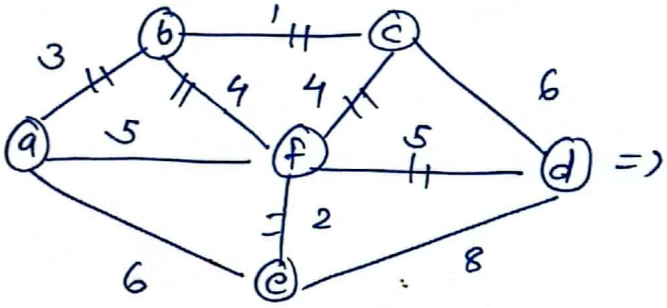




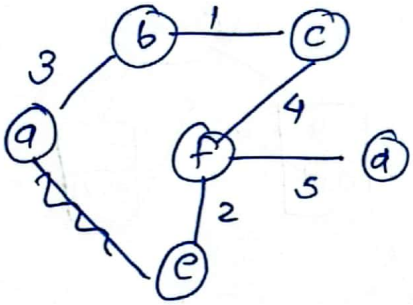
2+6+3+5+4

= 20

Kruskals Algorithm



=>



very easy!

1 + 3 + 4 + 5 + 2 = 15

Cost

Dijstraks Algorithm

Huffman coding Trees

Message :- BCCABBDDAECCABAEDDCC

length of message = 20

ASCII - 8-bits

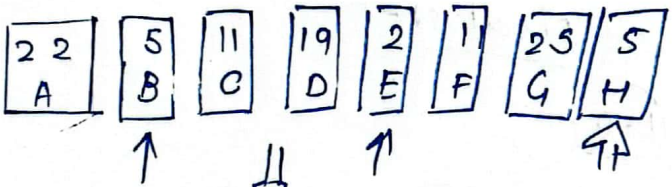
$8 \times 20 = 160$ BITS

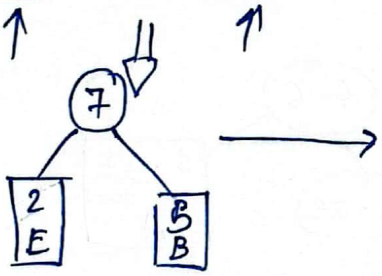
A	65	01000001
B	66	01000010
C	67	:
D	68	:
E	69	:

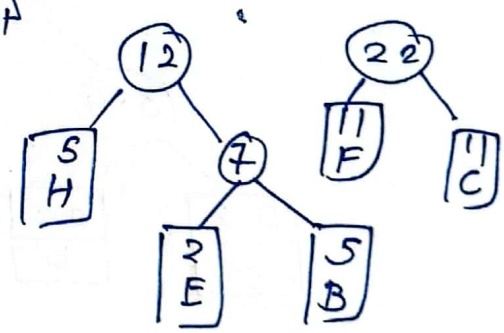
character	count / freq	code.

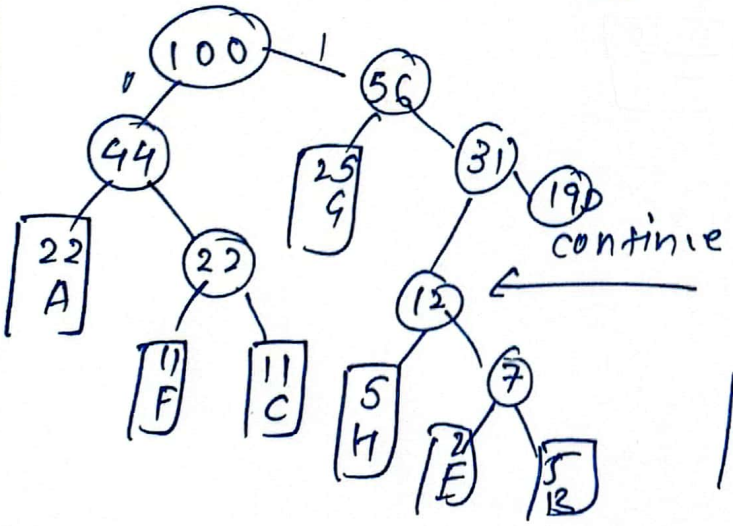
Suppose A, B, C, D, E, F, G and H are 8 data items and they are assigned weights as

Data items	A	B	C	D	E	F	G	H
weights	22	5	11	19	2	11	25	5

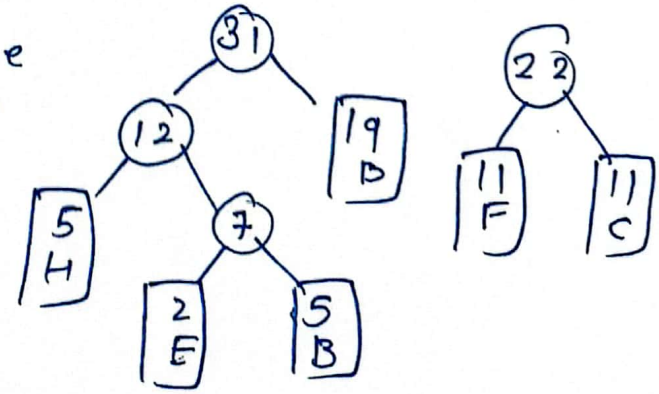




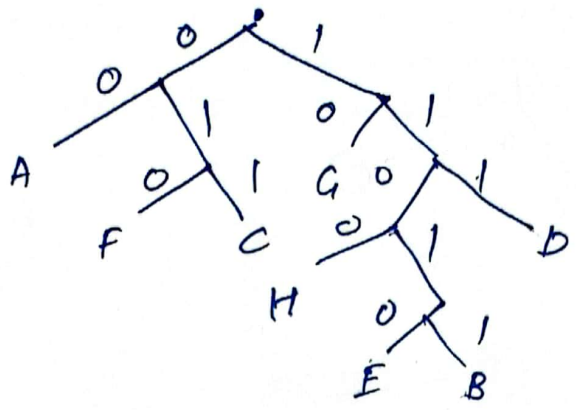




continue



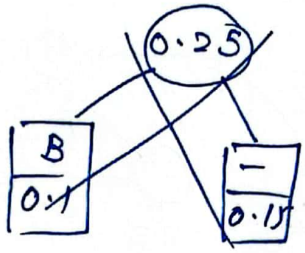
Huffman coding



A : 00
B : 1101
C : 011
D : 111
E : 11010
F : 10
H : 1100

Huffman code for following data

symbol	A	B	C	D	—
frequency	0.4	0.1	0.2	0.15	0.15



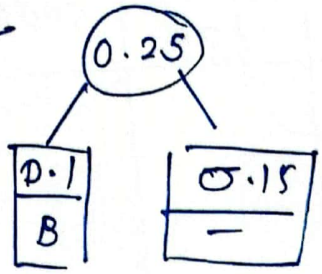
Step 1

Constructing of Huffman coding tree

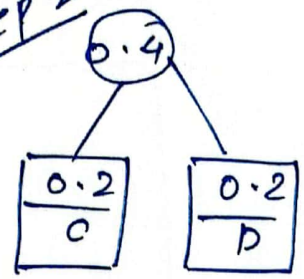
symbol	A	B	C	D	-
frequency	0.35	0.1	0.2	0.2	0.15

} Always consider 2 minimum frequencies to compare

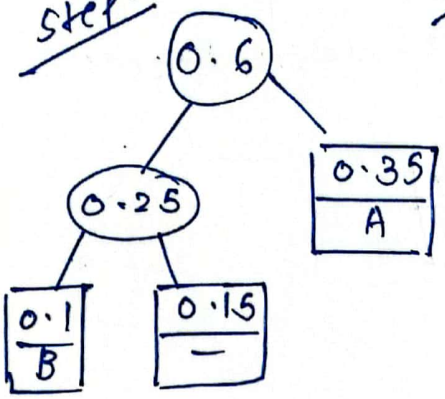
step 1



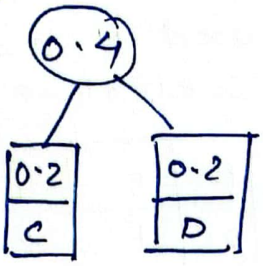
step 2



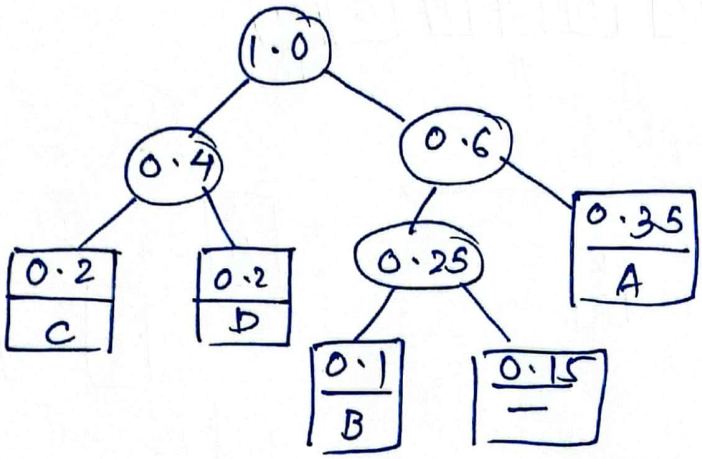
step 3



step 4

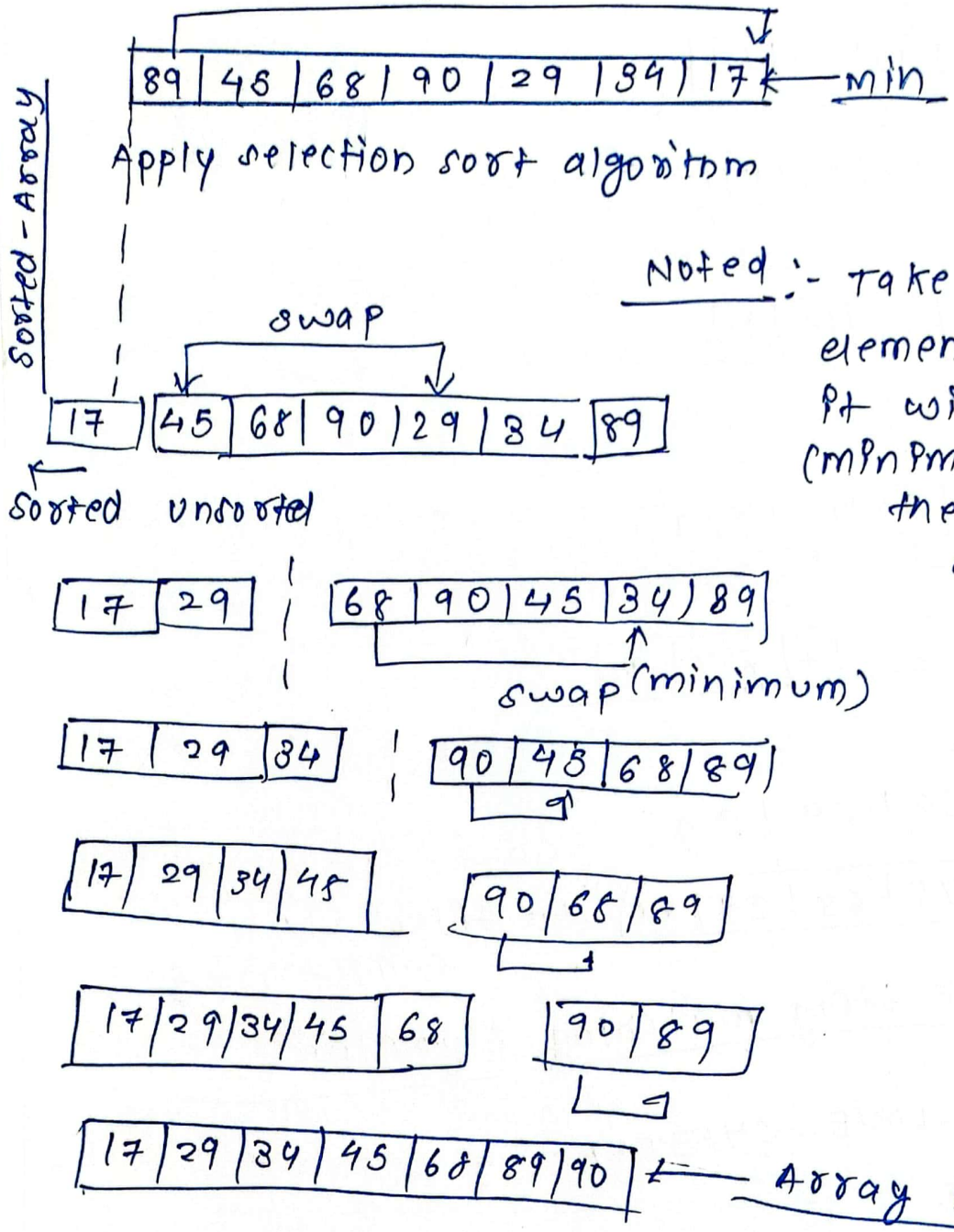


final step



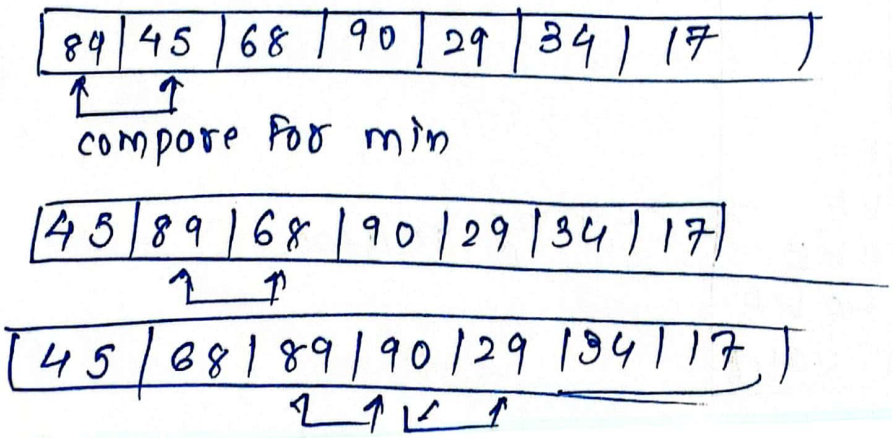
Chp 2: Brute Force Algorithms

Selection sort Algorithm



Noted :- Take the first element and compare it with the smallest (minimum element of the unsorted array.

Bubble-sort Algorithm



| 45 | 68 | 89 | 90 | 29 | 34 | 17 |

| 45 | 68 | 89 | 29 | 90 | 34 | 17 |

| 45 | 68 | 89 | 29 | 34 | 90 | 17 |

| 45 | 68 | 89 | 29 | 34 | 17 | 90 |

| 45 | 68 | 29 | 89 | 34 | 17 | 90 |

∴ | 45 | 68 | 29 | 34 | 17 | 89 | 90 |

∴ ~~17 | 34 | 4~~

45

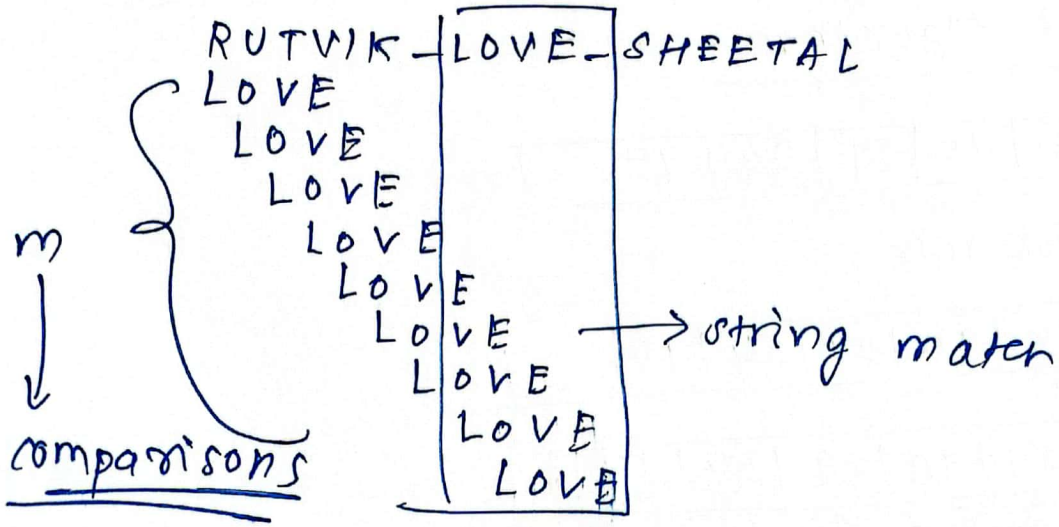
17 | 29 | 34 | 68 | 89 | 90

| 17 | 29 | 34 | 45 | 68 | 89 | 90 | ← After all comparisons

Brute-Force String matching Algorithm

& RUTVIK_LOVE_SHEETAL = T

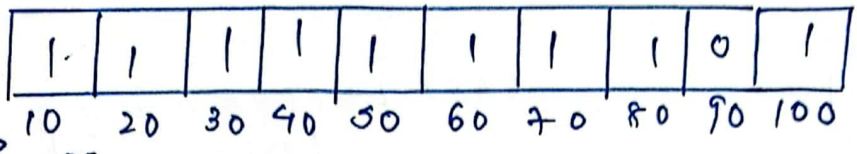
P = LOVE



Space and Time-Trade-off

Sort - given element using sorting By counting

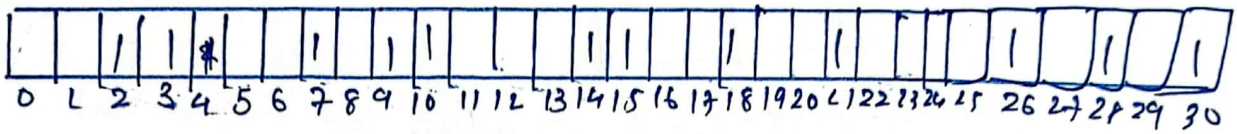
50, 60, 70, 40, 80, 20, 10, 80, 100



input size
A[10 . . . 100]

Traverse → 10, 20, 30, 40, 50, 60, 70, 80, 100

input size A[0 . . . 30] 21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3



Traverse from L to R Array

~~2, 3, 4~~

2, 3, 4, 7, 9, 10, 14, 15, 18, 21, 26, 28, 30

Horspool Algorithm

BANGALORE

STRING MATCH LORE

string match table

A	B	C . . . E . . L . . O	R	. . Y . . Z
4	4	4 . . . 4	3	2	1	. . 4 4

B A N G A L O R E
L O R E
. . . L O R E
. . . . L O R E

string match succesful

Boyer Moore Horspool Algorithm for pattern Matching.

T = A I N A I S E S T I Z A I N A I N E N

P = A I N A I N E N (m = 8)

LAST LETTER = * = 8

m = 8 = *

Bad match table

A	I	E	N	*
4	3	1	8	8

value = length - index - 1

E = 8 - 6 - 1 = 2 - 1 = 1

value I = 8 - 4 - 1 = 3

8 - 3 - 1 = 5 - 1 = 5

11th pos

T = A I N A I S E S T I Z A I N A I N E N

P = A I N A I S E N

mismatch

A I N A I N E S

1 2 3

A I N A I N E N

match found

Apply Horspool and boyer-Moore algorithm.

P - NOT

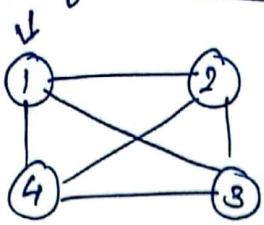
T = NOBODY_NOTICED_HIM

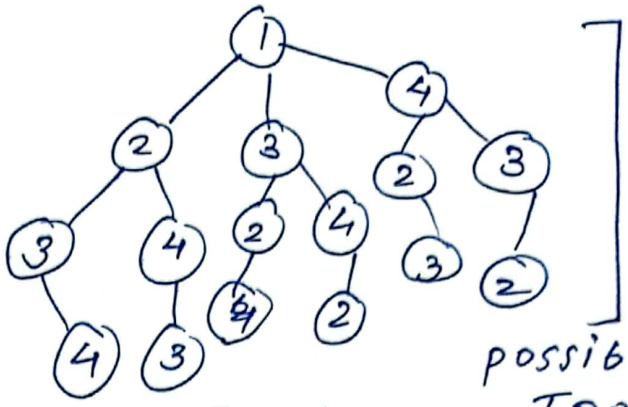
Horspool method

STRING MATCH TABLE

A	B	C	D	..	N	O	T
3	3	3	3		2	1	3

Traveling-salesperson Branch-& Bound





possible

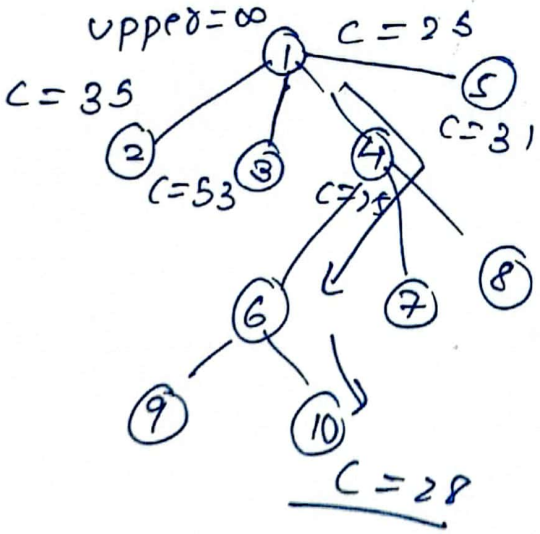
	1	2	3	4	5	
1	∞	20	30	10	11	10
2	15	∞	16	4	2	2
3	3	5	∞	2	4	2
4	19	6	18	∞	3	3
5	16	4	7	16	∞	4
						21

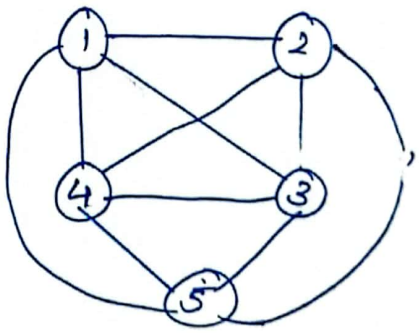
=>

	∞	10	20	0	1	10
	13	∞	14	2	0	2
	1	3	∞	0	2	2
	16	3	15	∞	0	3
	12	0	3	12	∞	4
	1	0	3	0	0	+ 21 = 25

	1	2	3	4	5
1	∞	10	17	0	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞

reduced cost = 25





$c(i,j) + c(R) + \hat{R}$