DSA Graph (slides)

→ flight scheduling

Graph = $G(V, E)$

array & linklist must for pass

array → tree in final lab

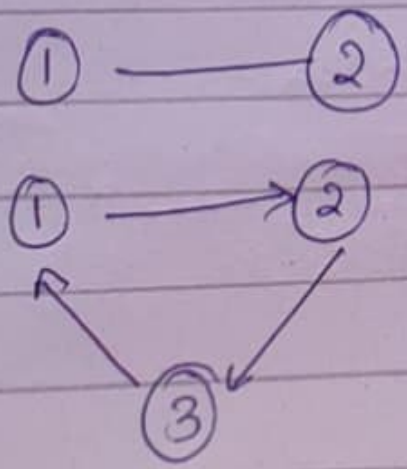
eg

$V = \{0, 1, 2, 3, 4\}$ edges = show relationship

$E = \{(0,1), (1,2), (0,3), (3,0), (2,2), (4,3)\}$

(0,1): 0 = source node, 1 = destination node

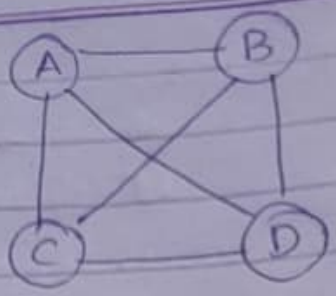
Path - a sequence of consective edges from source to destination



src = 1

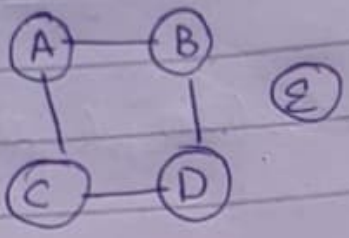
dst = 3

Path [1, 3] = [(1, 2), (2, 3)]



Complete graph
undirected

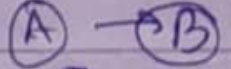
if complete, then connected



Directed - strongly

Agr har ek node say har dusry node pr jany ka rasta hai

Weekly - ek direction ka rasta ho



node لیکن ہر ایک
link okay ہے I

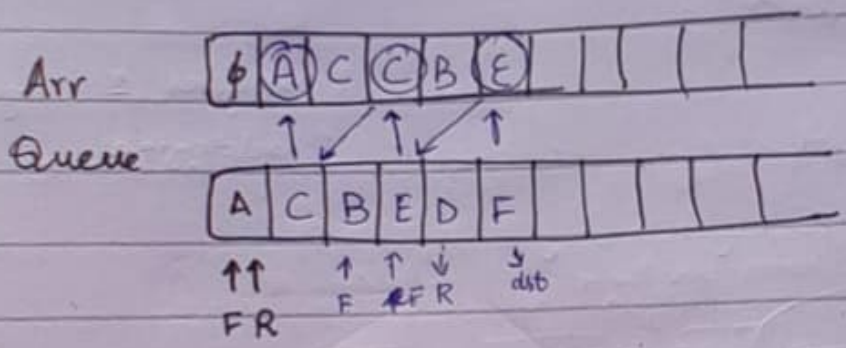
→ Indegree = 0 ⇒ Source node
→ Outdegree = 0 ⇒ Sink node

Array based : Adjacency Matrix
Link list based = Adjacency List

1- Breadth First Search → Shortest path / Path with min steps
2- Depth First Search → Reachability

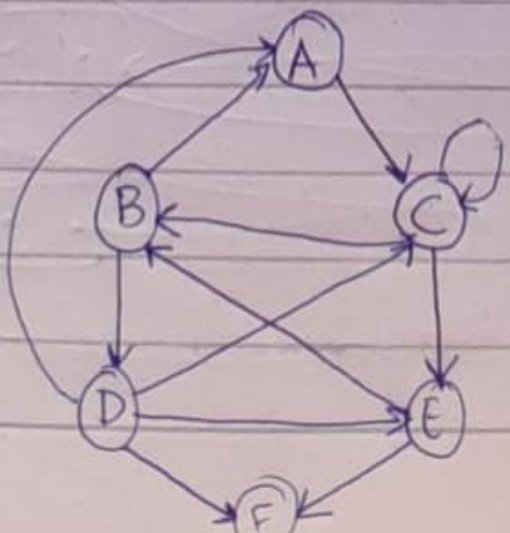
BFS

1- Array Define (Origin Array)
2- Queue



→ Insert adjacents (neighbors) of parent in queue & parent in arr until dst is found

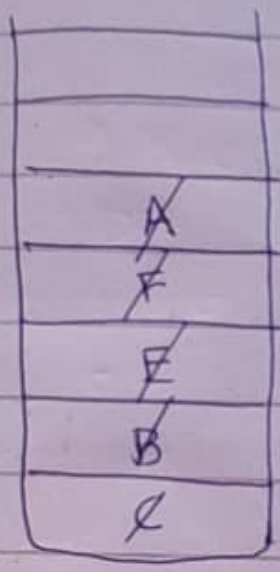
A → C → E → F



Depth First Search

DFS — If all nodes are reachable from specific point or not.

airline scenario



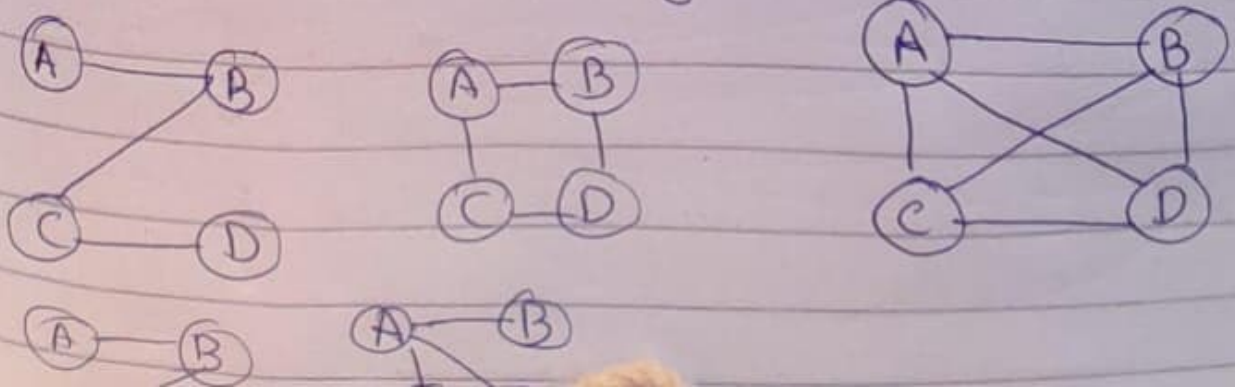
First push src &
Then pop it &
push its neighbors

Popped
C, E, F, B, A,

Minimum spanning tree

Spanning Tree

→ concept of undirected graph
→ weighted graphs
→ to remove redundancy



15/12/23

HEAP Tree

⇒ Type of Complete Tree

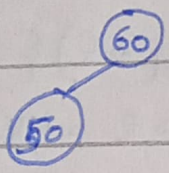
Min Heap → Ascending Order

Max Heap → Descending Order.

Parents node ≤ Child Nodes parents > child nodes.

Min Heap :-

60, 50, 45, 40, 50, 65, 70, 80, 30, 35, 25

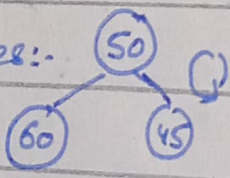


$2^0 = 1$ → level 1

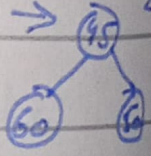
Now Compare ~~If parent~~ node with parent node :-

If node is less than parent node the swap

Tree becomes :-

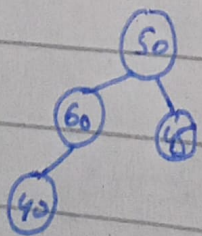


Againswap →

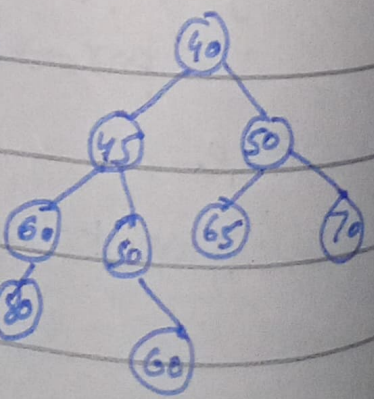


~~Again Sw~~

Inserting (40) to new level :-



Again Swapping

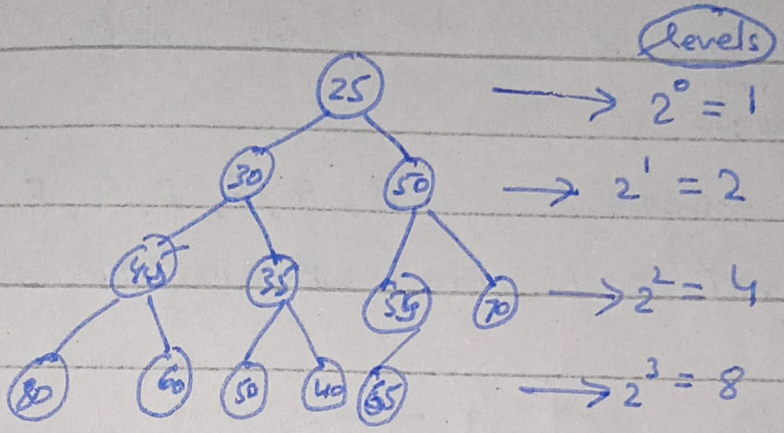


inserting (50) (65) (70) (80)

Again swap (45) and (60):-

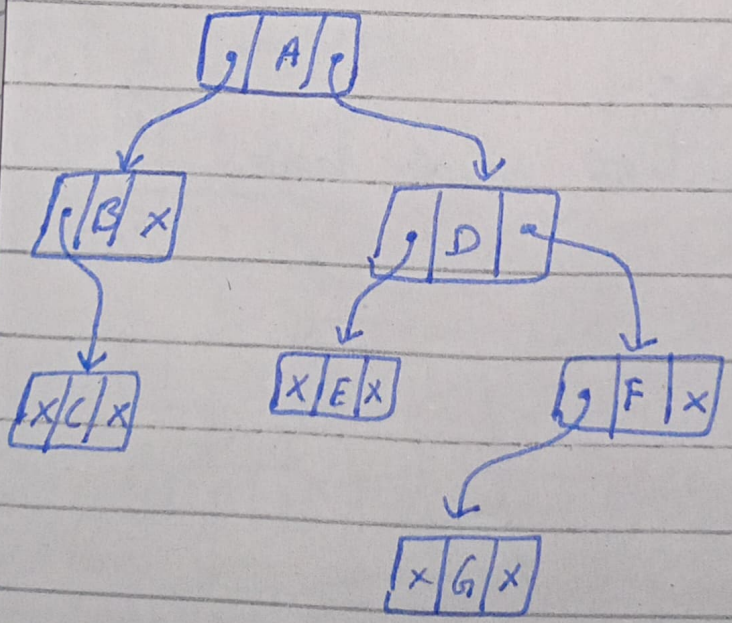
The tree becomes true under Min heap Condition:-

parent node ≤ child node :-



Print root (25)

Threads



Nodes = 7

pointers = 14

used = 6

unused = 8

How to save this unused memory?

(By using threading)

Inorder Traversal:- C, B, A, E, D, G, F

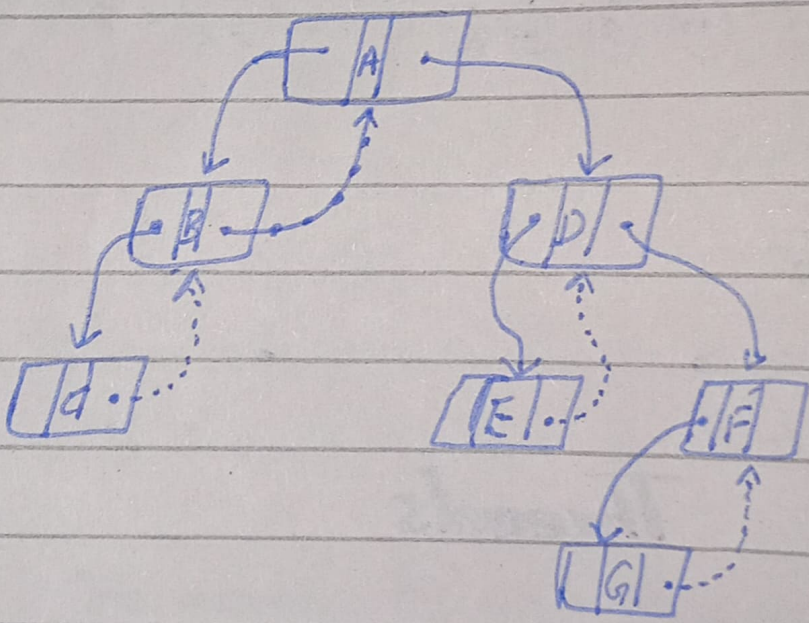
Threading :-

1- One-way inorder threading
2- Two-way " " with header.
3- Two-way " " with node.

One-way :-

Inorder traversal :- C, B, A, E, D, G, F

Connect each node with its successor



In Two-way with Header :-

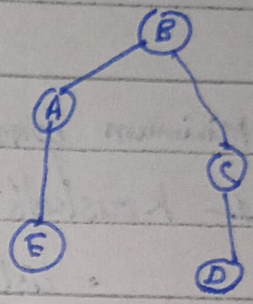
Connect free nodes with header node.

18/12/23
LAB.

Graphs.

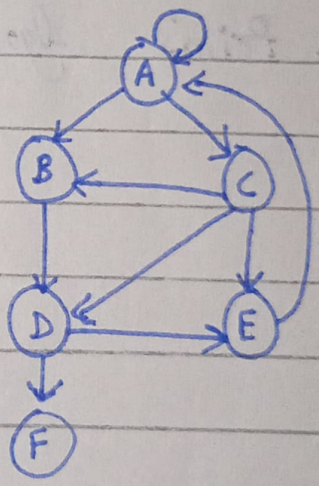
$M[5,5]$ =

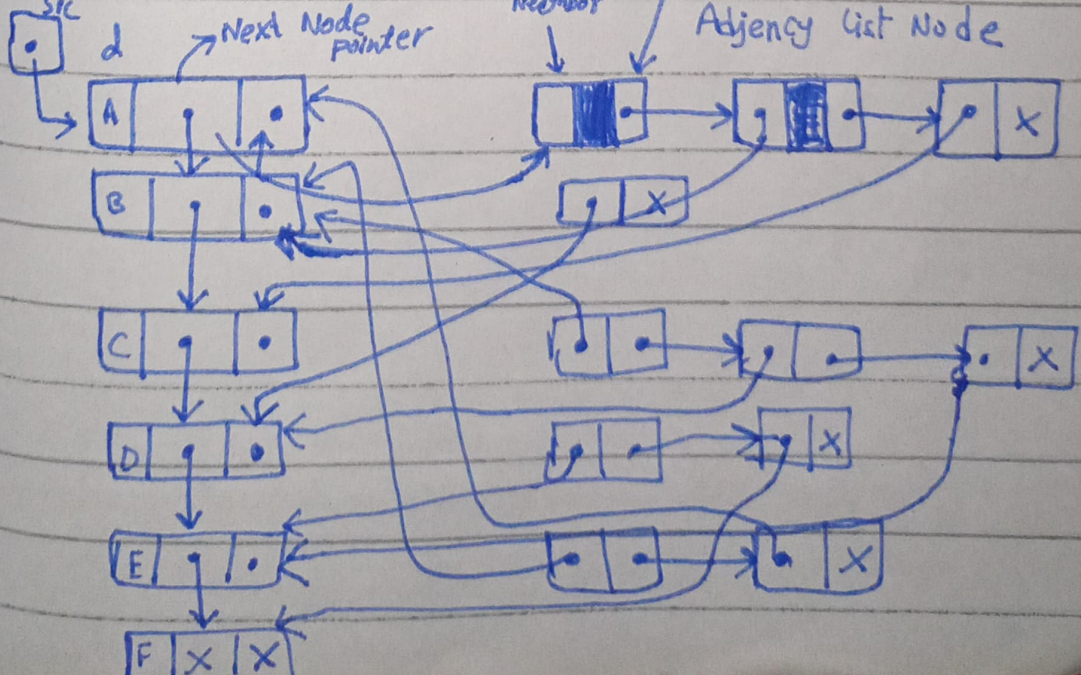
	A	B	C	D	E
A	0,0	0,1	0,2	0,3	0,4
B	1,0	1,1	1,2	1,3	1,4
C	2,0	2,1	2,2	2,3	2,4
D	3,0	3,1	3,2	3,3	3,4
E	4,0	4,1	4,2	4,3	4,4

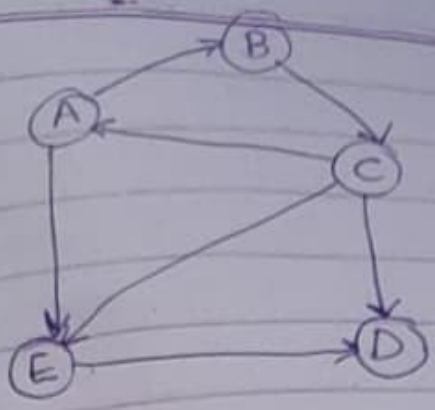


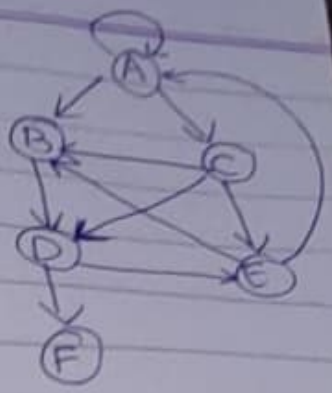
Adjency lists representation:-

Adjency list	Directly reachable nodes
A	A, B, C
B	D
C	B, D, E
D	E
E	B, A
F	-









Directly connected (outdegrees)

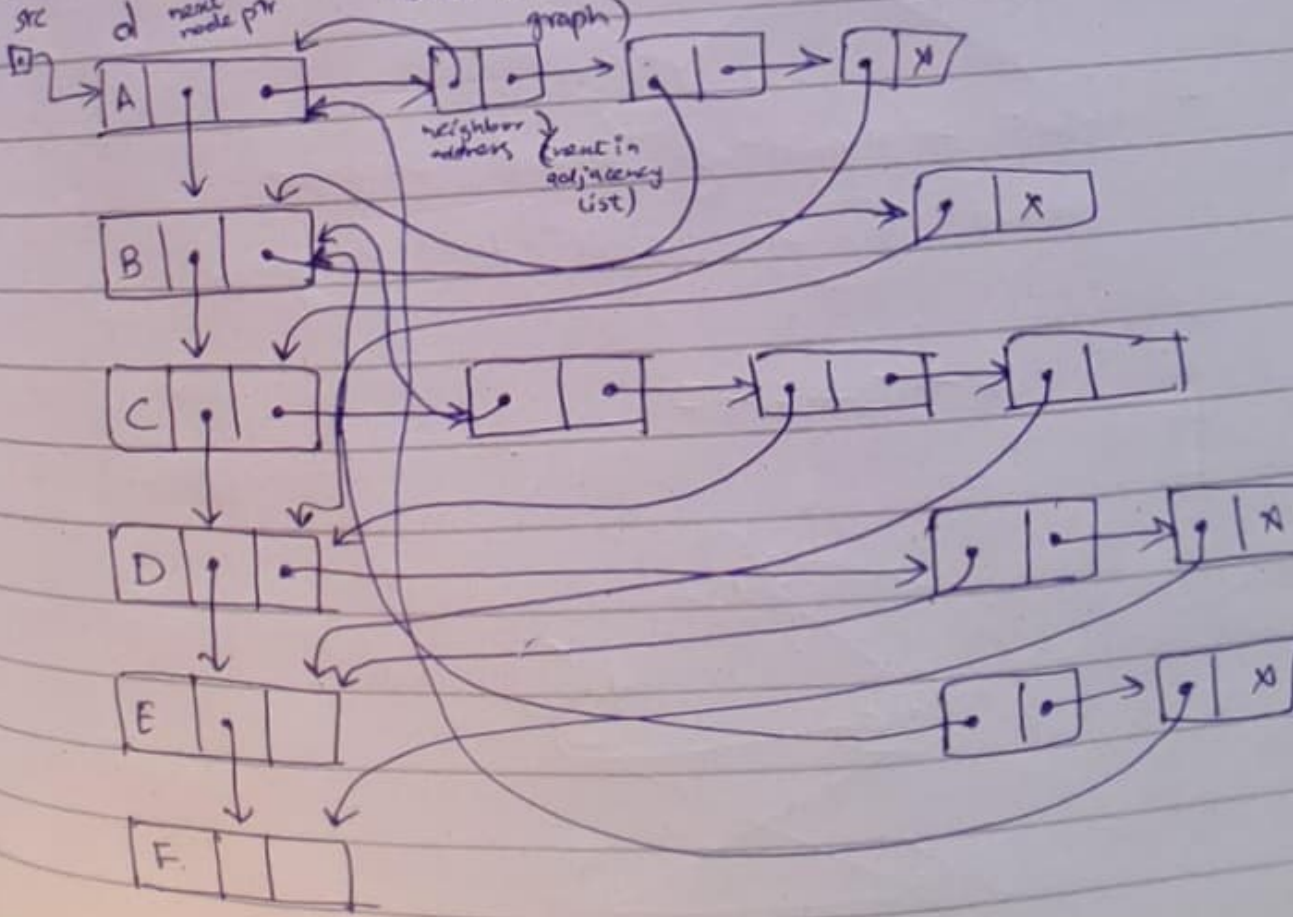
A	A, B, C
B	D
C	B, D, E
D	E
E	B, A
F	

Digraph — loops
Direct edge dekha

Loop gives 1 indegree
1 outdegree

→ (ensuring how many nodes are there in graph)

Adjacency list



Spanning Tree :-

→ Undirected Graph

→ Weighted Graph

Minimum Spanning Tree :-

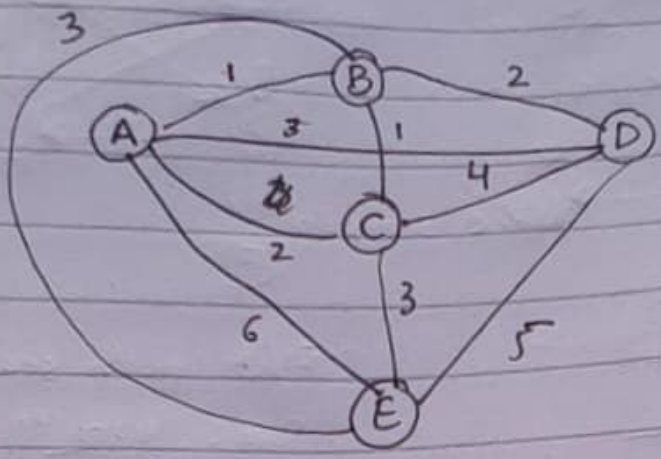
1- Kruskal's Algorithm.

- List all edges in ascending

2- Prim's Algorithm

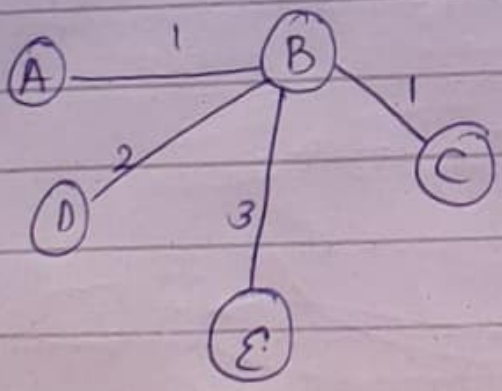
Best possible solution

1 Kruskal's Algorithm



1 List all edges in ascending

safe edge → jisey koi cycle na bney



A, B 1
B, C 1
× A, C 2
B, D 2
A, D 3
B, E 3
× C, E 3
× C, D 4
× D, E 5
A, E 6

2- Prim's Algo

time - efficient

Start from B