① AdjlistConvert(int [][] a)

	Time Req.
l = a[0].length	1
adjlist = new ArrayList(l)	1
for i = 0 to i < l do	n
adjlist.add(new ArrayList())	n-1
for i = 0 to a[0].length	n
for j = 0 to a.length do	n^2
if a[i][j] == 1	n^2-1
adjlist.get[i].add[j]	n^2-1
return adjlist	1

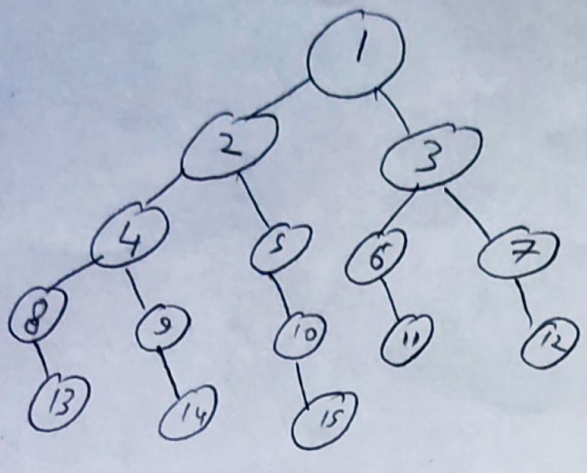
The time complexity of the algorithm is $O(N^2)$

② AdjMatrix convert(adj[], V)

	Time Req.
matrix[][] = new [V][V]	1
for i = 1 to V	n
for j = 1 to adj[i] do	m
matrix[i][j] = 1	m-1
return matrix	1

The time complexity is $O(N*M)$

(3)



i) BFS Traversal

1 → 2 → 3 → 4 → 5 → 6 → 7 →
8 → 9 → 10 → 11 → 12 → 13 → 14 → 15

DFS Traversal

1 → 2 → 4 → 8 → 13 → 8 → 4 → 9 → 14 →
9 → 4 → 2 → 5 → 10 → 15 → 10 → 5 → 2 → 1 → 3 →
6 → 11 → 6 → 3 → 7 → 12

ii) Maximum Size of Queue in BFS is 5

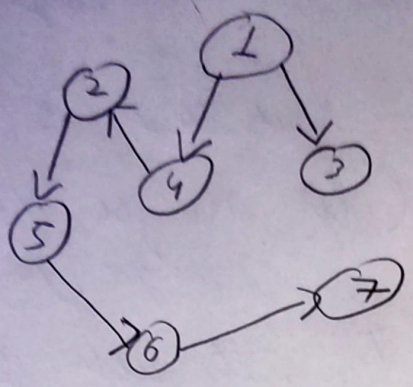
Maximum size of Stack in DFS is 4

iii) for searching node 6, we preffare BFS as it will take less time to reach node 6 than DFS

iv) for searching node 14, we preffer DFS as it takes less time to reach node 14 than BFS

(4)

i)



ii) Tree Edges : 1-3 , 1-4 , 4-2 , 2-5 , 5-6 , 6-7

Back-Edge : 2 - 1

forward Edge : 4 - 6

cross Edge : 7-3 , 7-4

iii)

iv) Topologically sorted order of nodes of graph is

1 → 4 → 2 → 5 → 6 → 7 → 3

(5)

let G be a graph with V vertices & E edges.

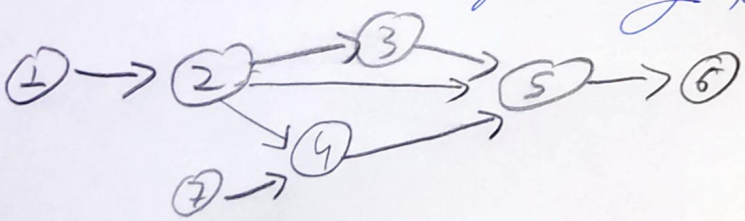
let T be a MST such that E_1 is not included in T (let E_1 be the least weighted Edge)

if we add E_1 in T, it will form 1 cycle.

we have to remove an edge from T to get back a MST. So we choose to remove the most weighted vertex (which is not E_1)

∴ E_1 must be included in T for it to be a minimum spanning tree.

(6)



in Graph G

let we reverse the forward Edge 2 → 5 to back edge 2 ← 5 and use DFS traversal

we traverse 1 → 2 → 3 → 5

in 5 we have an option to traverse back to 2. Hence, there exist a cycle in G.

So, G is not DAG

now if we use the given 2 → 5 edge in a and traverse

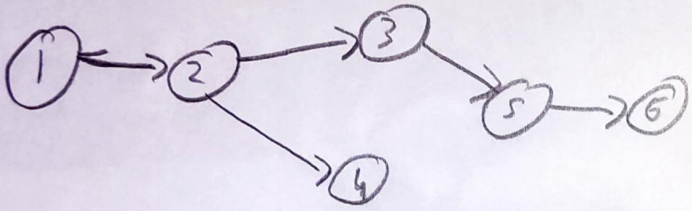
after reaching 5, there won't be any cycles to track back to a visited vertex.

Hence G is acyclic.

∴ G is a DAG when there is no back edge in DFS traversal of G

Hence proved.

DFS traversal for tree in G



Tree Edge : 1-2, 2-3, 3-5, 5-6, 2-4

forward Edge : 2-5

~~Back Edge : 7-4~~

Cross Edge : 7-4, 4-5