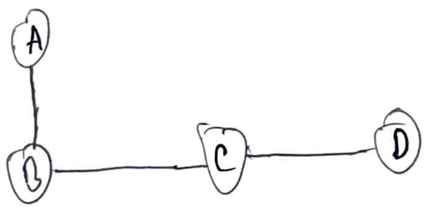
Name: Md Raihanul Islam Bhuiyan
Id: 20101239   Section: 3   faculty: MII

CSE 221

Assignment - 2

1. Same order for BFS and DFS:

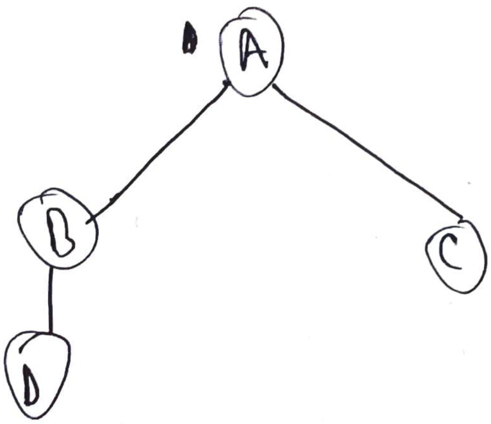


Start = A

BFS = A → B → C → D

DFS = A → B → C → D

Different order for BFS and DFS:



BFS:

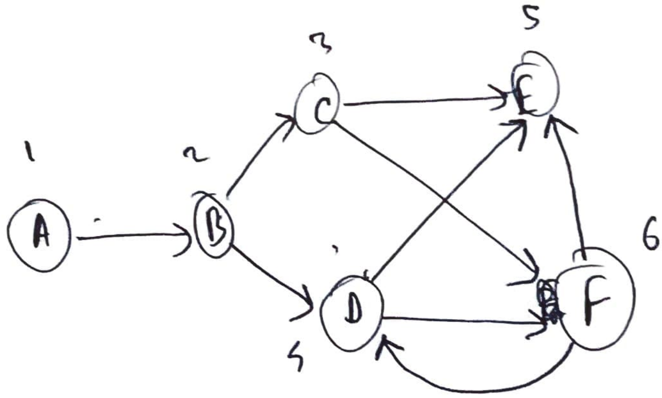
A	B	C	D

DFS:

A	B	D	C

Ans to the Q: N: 2

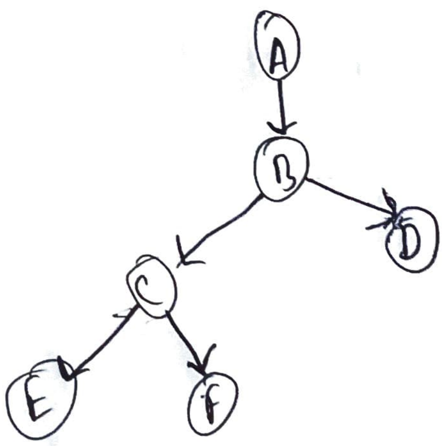
BFS:



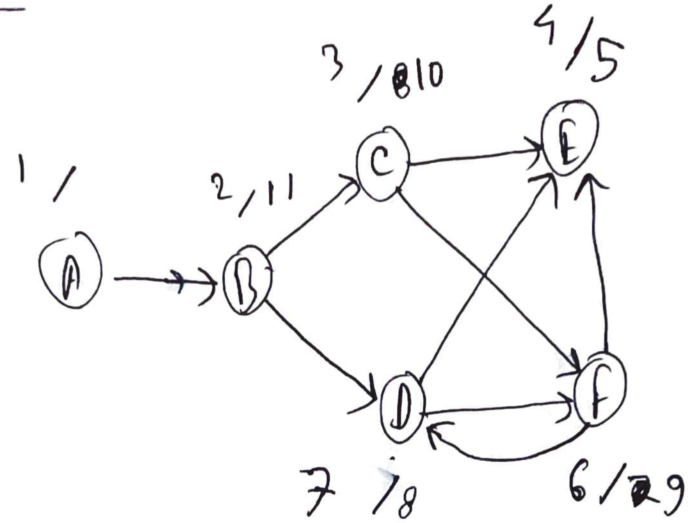
list =

A	B	C	D	E	F

Tree:



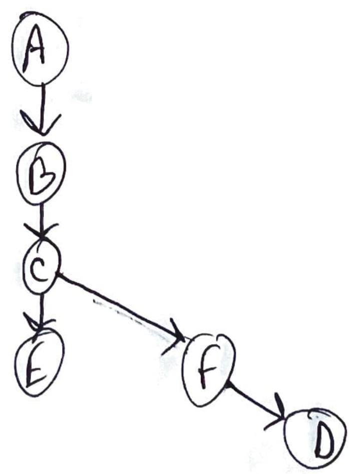
DFS:



List:

A	B	C	E	F	D

Tree:



Ans to the Q: N: 3

We are going to use BFS, so that we can find if each node is connected to another. ~~directly or~~

Algorithm:

BFS (G, s):

visit[s] = True

Q ← s

out = [ ]

out ← s

While Q not empty:

m ← DEQUEUE (Q)

```
for each child n of m:
    if visit[n] = False:
        visit[n] = True
        out ← n
        Q ← ENQUEUE (n)

if len(out) == G.keys():
    print ('All nodes are connected)
else:
    print ('All nodes are not connected)
```

Here, we have applied ~~f~~ BFS to find if ~~ea~~ we can transport to any node from any other node.

We applied BFS and ~~af~~ put the path of BFS inside an array named 'out'. So, if all nodes are connected, then 'out' will have all nodes from the graph. That means, ~~we~~ 'out' will be equal to graph.keys(). Otherwise, ~~the~~ all the nodes are not connected.

Time complexity: ~~As we have~~

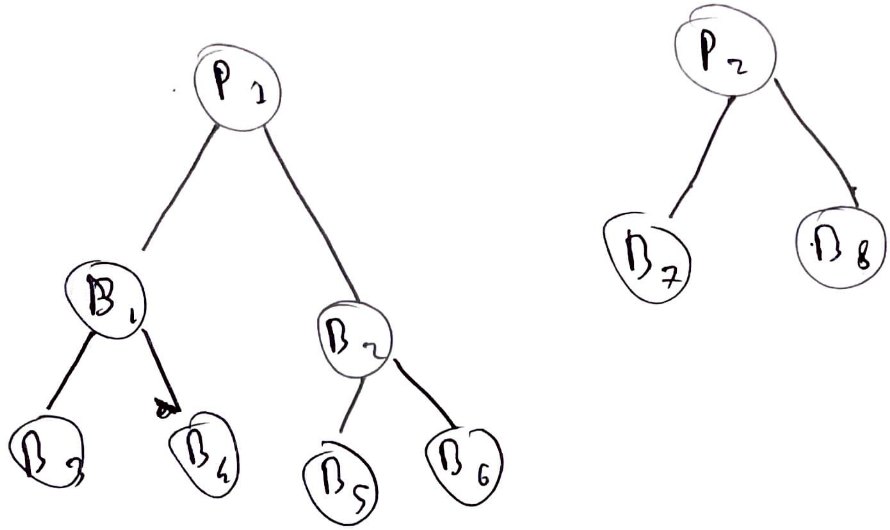
Here, time complexity of while loop is $O(m)$ and time complexity of the for loop is $O(n)$.

$\therefore$ Total time complexity: $= O(n+m)$

Ans to the Q: N: 5

We need to check which powerplant has the maximum number of buildings connected to it. We can use DFS algorithm to solve this problem.

In DFS algorithm, we ~~vis~~ go in Depth of a node and visit ~~it~~ all the children of that node. For this problem, let $n = 2$ and $n^3 = 8$



Now, if we apply DFS on ~~one~~ every powerplant as source, the powerplant which has maximum number of buidings visited will have the generator.

Here ~~vis~~ len(visit[P1]) will be 6 and len(visit[P2]) will be 2. So, P1 will have the generator.

Just like that we need to apply DFS to n number of powerplants.

In the worst ~~can~~ case, all the ~~plan~~ buildings will ~~so~~ be connected to one powerplant.

The powerplant which has the highest number of visited building will have the generator.

<u>Time complexity:</u>

Here ~~so~~ vertices $= n^3$

edges $= \dfrac{n^3(n^3+1)}{2}$

$= \dfrac{n^6+n^3}{2}$

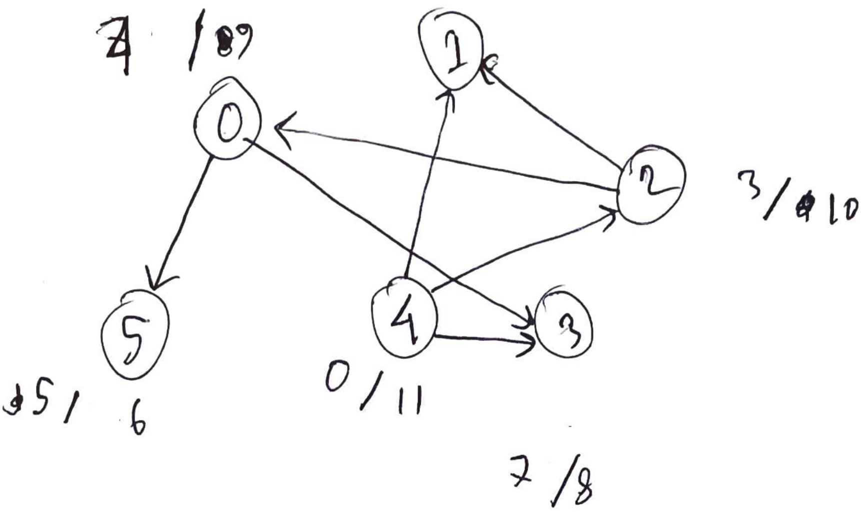
$\therefore$ Time complexity of DFS $= O\left(n^3 + \frac{n^6 + n^3}{2}\right)$

$$= O(n^6)$$

So, in the worst case, time complexity will be $O(n^6)$

Ans to the Q. N: 5

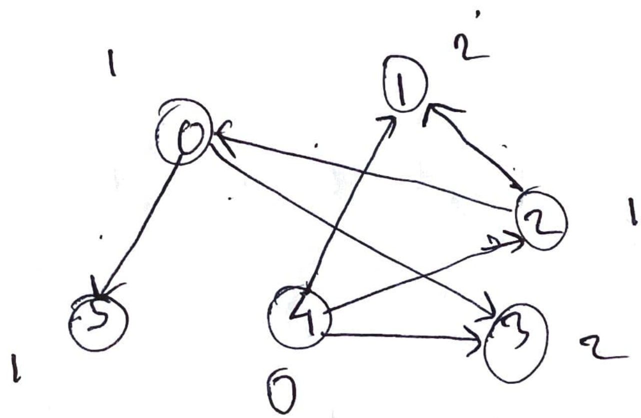
(a)



Topological sort: 4, 2, 0, 3, 5, 1

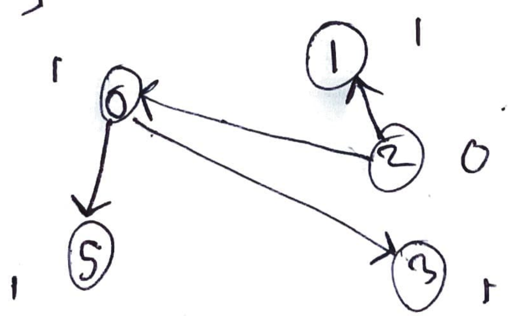
Here, we have found only 1 topological orders. But there can be more distinct topological sort. For that, we are going to apply different method. We will mark the number of incoming edges of each vertices and extract those vertices whose incoming edges are 0. We will keep extracting them untill all of the vertices are extracted.

Graph



Extract 4 and all its outgoing edges.

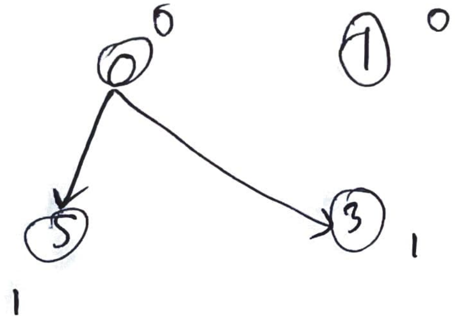
list=[4]



extract 2.

list = [4, 2]

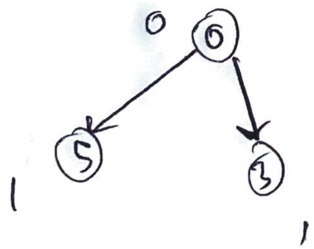
Graph:



We can either extract 0 or 1. So, there can be two combinations.

~~list~~ Let, list1 = [4, 2, 1] (if we extract 1)

Graph after list 1:



extract 0.

list -- [4, 2, 1, 0]

Graph :

5   3

0   0

if we extract 5,

list 1 1 = [4, 2, 1, 0, 5]

graph !   3 0

extract 3

list 1 1 1 = [4, 2, 1, 0, 5, 3] ......(i) if we extract 3 first, instead of

list I I,

list II2 = [4, 2, 1, 0, 3, 5] ii

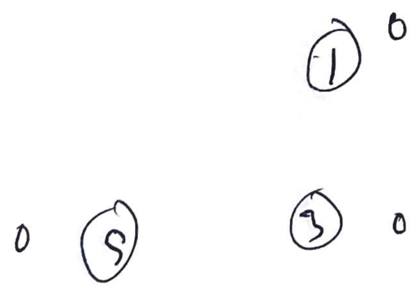
Now,

before we created list I, if we extract

0 first,

list 2 = [4, 2, 0]

graph:



Now. we can extract in 6 orders.

list 21 = [4, 2, 0, 1, 3, 5] .... (iii)

list 22 = [4, 2, 0, 1, 5, 3] .... (iv)

list 23 = [4, 2, 0, 3, 1, 5] .... (v)

list 24 = [4, 2, 0, 3, 5, 1] .... (vi)

list 25 = [4, 2, 0, 5, 1, 3] .... (vii)

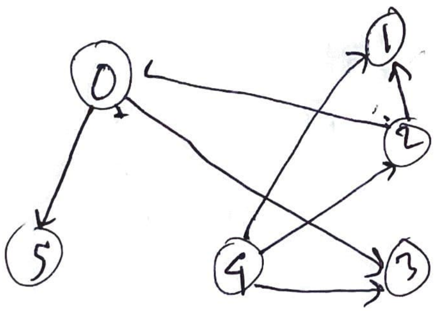
list 26 = [4, 2, 0, 5, 3, 1] .... (viii)

So, we find that there can be 8 distinct topological orders.

(b)

There won't be any topological sort in G if there is a cycle in the graph G.

Given graph:

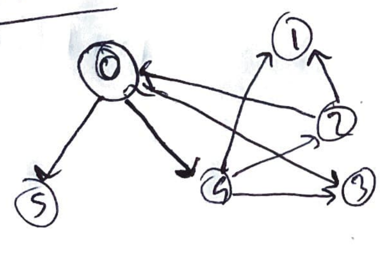


Here, we see that 4 has no incoming edges. Now if ~~we connect~~ 4 gets any incoming edges from a vertice which is not directly connected with 4 at present, then it will not have any topological sort.

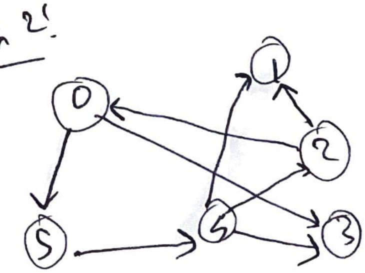
Here, we find that only 0 and 5 has no direct connection with 4.

So, there can be two ~~[illegible]~~ distinct edges that could be added to G and construct a graph with no topological ordering. Those graphs are given below.

Graph 1:



Graph 2:



The edges are:

$0 \to 4$ and $5 \to 4$