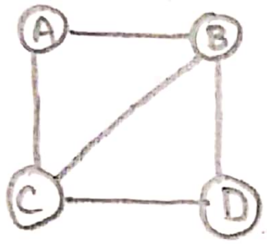
CSE221 Assignment 2

Mohammod Ariful Islam

ID: 20101192

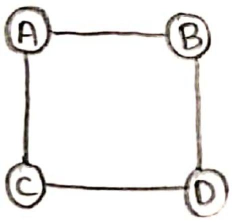
Sec: 02

Solution to 1:



For BFS: A — B — C — D

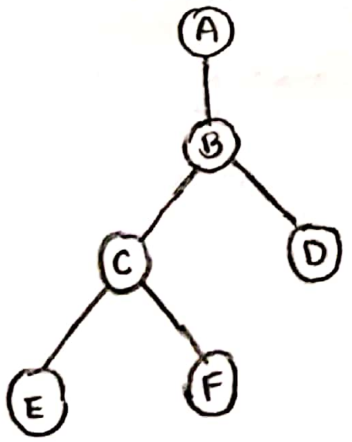
For DFS: A — B — C — D



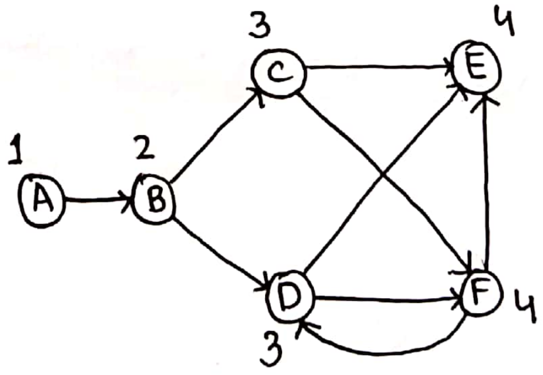
For BFS: A — B — C — D

For DFS: A — B — D — C

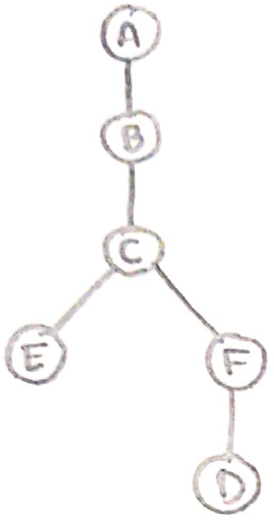
Solution to 2:



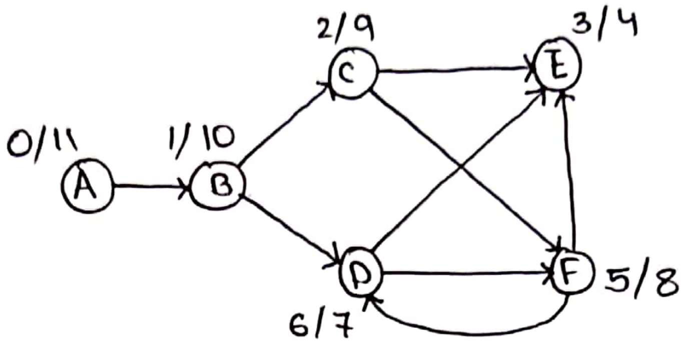
BFS



A – B – C – D – E – F



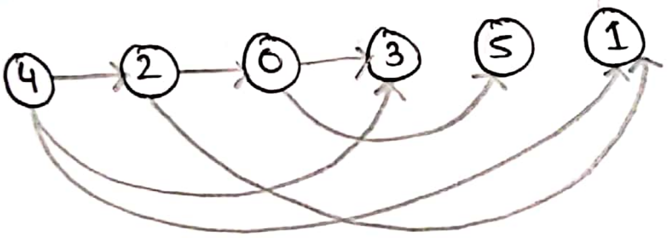
DFS



A – B – C – E – F – D

Question 5

(a)

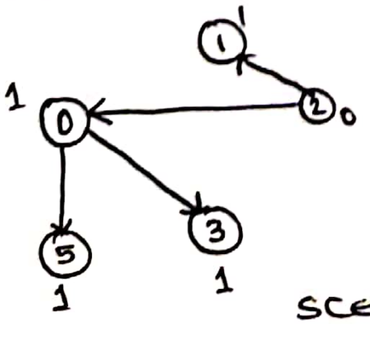


We are going to use a clever way to find all possible topological orderings of G. For each node, we keep a count of ~~the~~ it's indegree and after selecting a node, we update our order and in-degree of remaining nodes.

We begin with ④ in every order because it has zero indegree to begin with.

Order 1 : ④

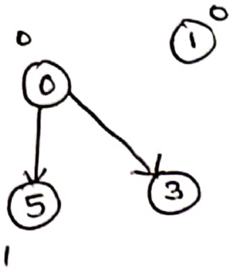
Then, delete it and update indegree of other nodes.



We now select ② in our order and delete the node from this simulation.

scenario 1

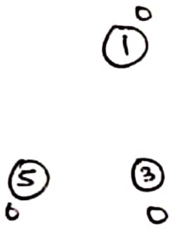
order 1 : ④ ②



We have two choices now which is either ① or ⓪. Let's choose ⓪.

scenario 2

order 1 : ④ ② ⓪



Here we can either choose ⑤ or ③ or ① in any sequence

order 3 : ④ ② ⓪ ① ⑤ ③

order 4 : ④ ② ⓪ ① ③ ⑤

order 1 : ④ ② ⓪ ③ ⑤ ①

order 2 : ④ ② ⓪ ⑤ ③ ①

order 5 : ④ ② ⓪ ③ ① ⑤

order 6 : ④ ② ⓪ ⑤ ① ③

Going back to scenario 2, we now choose ① first and then ⓪. and so on.

order 7~~3~~ : ④ ② ① ⓪ ③ ⑤

order 8~~4~~ : ④ ② ① ⓪ ⑤ ③

Therefore graph G has 8 topological orderings.

(5)(b) A directed edge from ⑤ to ④.

To construct a simple graph with no topological ordering, we need to add edges that would make G cyclic but also keeping in mind that there ~~are~~ is only one edge between two nodes i.e. there does not exist parallel edges.

Considering ④, we can ~~have~~ add only two edges

⓪ → ④ and ⑤ → ④

considering ②, we can add two edges

⑤ → ② and ③ → ②

considering ①,

~~① → ③~~

~~⓪ → ①~~ and ⑤ → ① and ~~③ → ①~~ ③ → ①

① → ⓪

considering ③,

~~⑤ → ② ③~~ ⑤ → ③

Hence, we can add 8 distinct edges to convert G to a simple graph with no topological ordering.

Question 3

The edges are directed and each node can have an indegree or out degree edge. To transport fuel from ANY node to ANY other node, the system or the graph must be strongly connected. We apply Kosaraju's algorithm in this problem.

For the network represented as graph X, we apply the DFS algorithm on X and put nodes in a stack whenever a node ~~is finished~~ has no more path to go towards. ~~This~~ The overall time the DFS would take is O(n+m).

Then, we transpose the graph X^T of graph X which takes O(n+m) time.

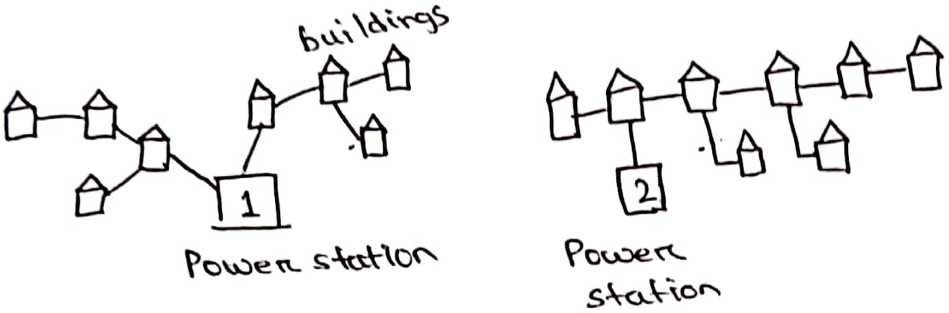
We apply the DFS algorithm on X^T by popping values of node from the stack. which takes O(n+m).

The whole graph should be one ~~co~~ strongly connected components so that there exists a path from any node to any node. If we get multiple connected components, all nodes may not have a path towards the rest of the nodes.

The overall time complexity of the process is 3(n+m) which is basically O(n+m)

Question 4

If we assume a scenario of the networks of the power plants and buildings ,



They are disconnected graphs. We can use a DFS algorithm with an adjacency matrix to traverse the entire network.

During the DFS traversal, when we reach a point when no new nodes are left to be discovered, we switch to another network by choosing another power station as our source node. For the DFS traversal of each power station, we keep a count of how many buildings it provides electricity and store it in a ~~varia~~ tuple (power station 1, x) and name it max.
↑ number of buildings.

After the DFS ~~trav~~ is run through power station 2, we ~~com~~ get y buildings that are connected to it.

If $x >$ ~~=~~ y, we keep max as it is.

If $x <$ ~~=~~ y, we update max = (power station 2, y)

If $x = y$, we cannot choose both so keep max as it is.

In this way, we traverse through n power stations and n^3 buildings.

no. of nodes = $n + n^3$

The adjacency ~~list~~ matrix gives us a time complexity of

$$O((n+n^3)(n+n^3))$$

$$= O(n^2+n^4+n^4+n^6) = O(n^2+2n^4+n^6)$$

which is basically $O(n^6)$.

We add the generator to the power station which is max[0].