Problem 1

List of in-vertices is {(A, 0), (B, 1), (C, 1), (D, 2), (E, 3), (F, 1), (G, 0)}

A new stack is created (S). Scanning through the list of in-vertices to locate the vertices having no in vertices generates the stack S = {(A, 0), (G, 0)}

While S is not empty

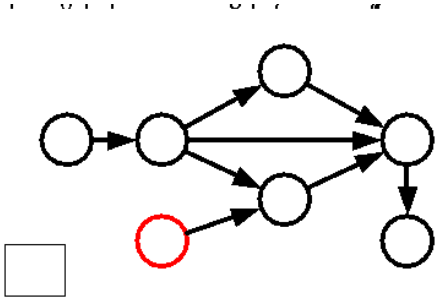
My n will be 7 since we have seven nodes on which to perform topological ordering.

n□ 7

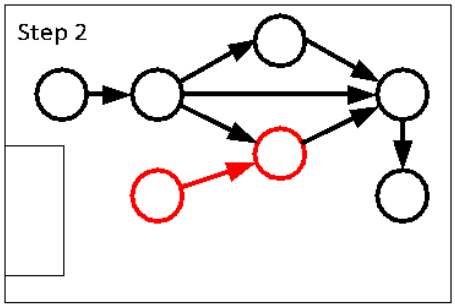
s□ S.pop()

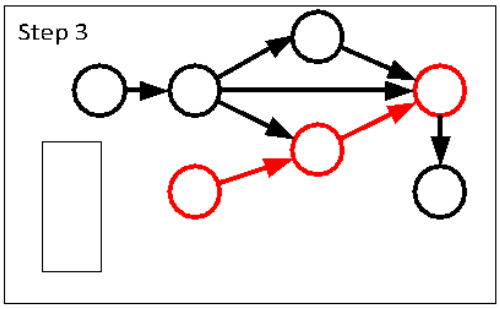
then running topological sorting

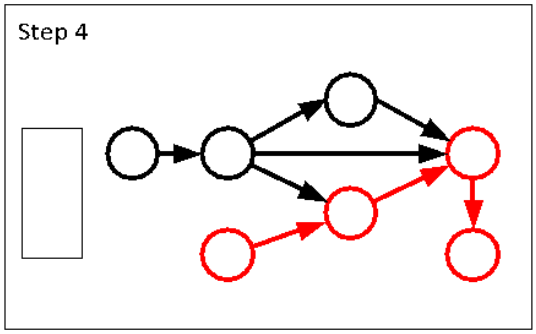
1. Running DFS.

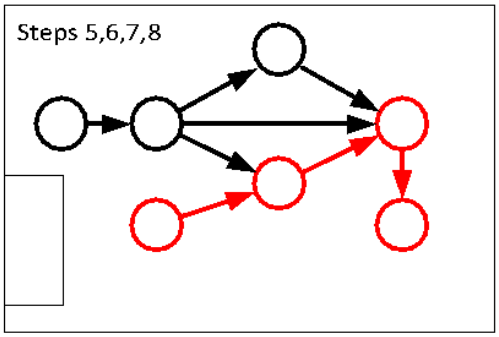


Steps



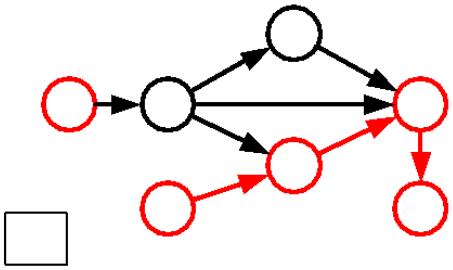


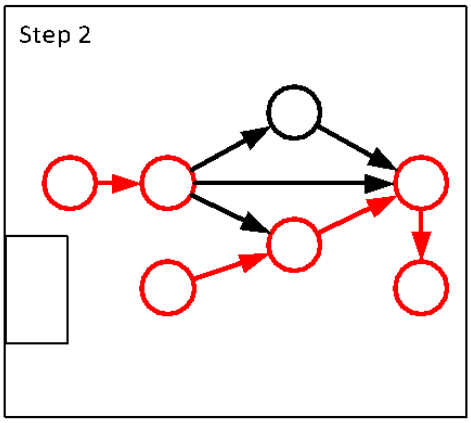


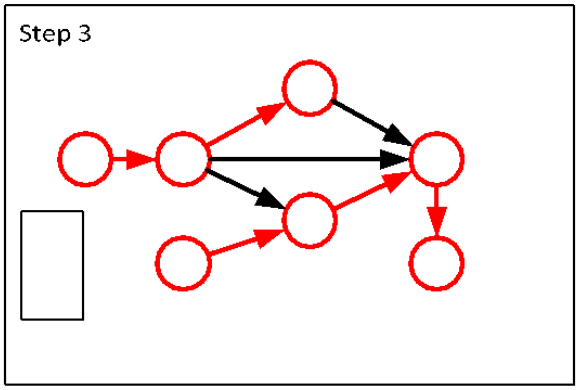


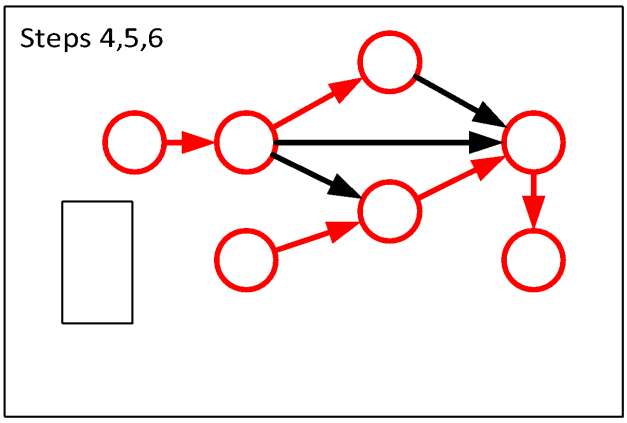
			G	D	E	F
1	2	3	4	5	6	7

2. After completing G. I pop A from the stack that has vertices with zero in vertices. And perform DFS again









Since all the vertices have now been visited, the topological ordering of the diagraph is,

A	B	C	G	D	E	F
1	2	3	4	5	6	7

Problem 2

Algorithm: IsReachableFrom(G, u, v)

```
Input: A directed graph G, vertices u, v in G

Output: TRUE if there is a directed path from u to v in G, false otherwise

visitedVertices □ new Set<Vertex // Set that stores all the vertices visited

S □ initialisation of a Stack

S.push(u)

visitedVertices.add(u)

mark u as visited

While! S.isEmpty() do

        v□ S.peek()

        if some out-vertex of u has not yet been visited then

                w□ next un-visited out-vertex of u

                mark w as visited

                visitedVertices.add(w)

        else

                S.pop()

return visitedVertices.contains(v) ? true : false
```

Running time

1. Each vertex is marked, pushed to the stack and eventually popped out of the stack. This takes O (1) time for each vertex.
2. Each vertex is added to the visited vertices map. This takes O (1) time for each vertex.
3. In the worst case, each vertex has a peek operation at the point of checking whether the vertex has any unvisited out-vertices which is the outdeg(v) for each vertex.
4. The time to check if the set of visited vertices contains the other end of the vertex that we are checking is O (1).

Thus, total runtime will be the summation from one to n of (1 + outdeg(v)) = O (n + m). Thus, the run time of the algorithm is O (m + n);