Algorithms PS-4

Deepanshu Parihar

Answer-1

Here I use BFS to detect a cycle in a graph. There are two cases here, first in case of edge within a layer and second of a vertex v has multiple parents.

**Algorithm-**

1. Run BFS on a graph G, taking any vertex say v (source).

2. If there exists an edge e within a layer.

Take the two vertices of that edge and backtrack their paths say path1 and path2 till a common parent is met or source s is reached(root)

3. Path = path1 +path2 +e

4.return Path

5.Else do,

For every vertex v except source s in BFS.

If there exists a vertex d with multiple parents

Backtrack the paths until a common parent is found or we reach source s.

Path= Path backtracked above

return Path

7. Else

Return “No cycle exists”

BFS tree takes (n+m) time to create. Then we do a lookup in BFS tree for n vertices. This time will be dominated by the time taken to form a BFS tree. So a total time of (n+m) would be needed for this algorithm.

Answer-2

Using the Hint provided on piazza-

Creating a cartesian product of nodes of graph g as a new graph g’ .Following all the rules made for Bob and Alice to follow, first we should remove all diagonal elements of the new graph g’ as at no time can Alice and Bob be on the same vertex. Also at no time they can be crossing the same edge. So there is no edge where the elements are swapped i.e. (s,t) to (t,s) is not possible. So these two rules are to be followed. Then using BFS we can find the shortest path possible.

**Algorithm**-

1. Create a new graph g’ containing all the possible vertices formed by the combination of every u,v ∈ V.

2. As per the constraints given , first discard all the possible edges to diagonal elements.

3. As per the constraints don’t make any edges in g’ where swapping of places is being done i.e. for all (u,v) belonging to graph g’ remove edge to (v,u).

4. Now according to the adjacency list of G create edges between (u,v) and (u’,v’) in g’ following the constraints given in steps 2 and 3 and in such a way that (u’,v’) can be reached from (u,v) by-

(a) Moving Alice represented by u and not moving Bob represented by v by one step in G.

(b) Moving Bob represented by v and not moving Alice represented by u by one step in G.

(c) Moving both Alice and Bob one step in G.

5. Run BFS on the newly formed graph g’ and find the shortest path from starting points (s.t) to exchanged points (t,s).

6. if value is equal to zero return “Not possible to trade places”

7. else return shortest path.

Running time- The formation of Cartesian product will take ( n2) time where n is the number of vertices in graph g. Formation of edges will take another time of ( n2). Then the BFS tree formation will take ( n2+m) time where m is the total no of edges in graph g’. Hence the total running time of the algorithm is ( n2+m).

Answer-3

1. The graph that will be formed for this problem here is definitely a DAG. This is because if there is a cycle of prerequisite courses then a student will not be able to take a course. So here I use topological sort.

Using Topological sort (as taught in class without any modification)

**Algorithm-**

Course\_Selection(G)

1. While there is a vertex v with no incoming edges
2. Add v to a List l
3. Remove vertex v and its edges

The list l is the list of courses to be taken over semesters and the time complexity here is same as topological sort Theta(n+m).

1. Again using topological sort with a slight modification. The graph here is a DAG hence using Topological sort-

**Algorithm-**

SemTotal(G)

1.Identify all source vertices with no incoming edge.

2.count=0

3.while graph is not empty

remove all source vertices and its edges and identify new source vertices

1. count++

First step will take n time at max to identify all source vertices with no incoming edge. Rest of the steps will take Theta(n+m) time at max. So the total is Theta(n+m).