CS 260P HW3

#2

1 If graph G is connected:

2 Run BFS starting from an arbitrary node s, we can then have a BFS tree T.

3 If every edge of G is in the tree then there are no cycles.

4 else there is some edge e= {u, v} that in G but not in T, which means there is a cycle from edge e and edge {w, v} and edge {w, u}, w is the closest common parents of u and v in the tree T.

5 Else if graph G is not connected (assume there are n parts):

6 do 2-4 separately in n parts of graph

#3

First, run the same algorithm in the Section 3.6.

If there is at least one node has no incoming edges then we can get the topological ordering.

Else if in the ith iteration, we find that every node has at least one incoming edge, then the graph must has a cycle. Then we randomly choose a node s and it’s incoming edge e<u, s>, and we choose node u and one of it’s incoming edge e’ and repeat the process until we find edge e’’<s, w> and the set of nodes that we get in the iteration is a cycle.

#4

To begin with, assign each specimen with a node, and there is an edge between two node is there is a judgment that they are “same”. Then we get some component Gi.

Arbitrarily select a component let’s say Gk, label Gk “A”.

Build another graph with each node for each component Gi.

For each judgement

{

if pairs(i, j) involving in the judgement occur in same component

then return there is an inconsistency

else if pairs(i,j) involving in the judgement occur in different component let’s say Gu and Gv

{

add edge e{Gu, Gv}

}

}

from Gk, put Gk in stack

while stack has node

{

pop n from stack

label n’s adjacent node with n’s opposite value

if any adjacent node already has label but is the same with n’s then return inconsistency

delete n from graph

put n’s adjacent nodes in stack

}

#6

Assuming that G has an edge e={v, w} that doesn’t in T.

Cause T is depth-first tree, let’s say v is w’s ancestor node, and since it’s also breadth-first tree, so the distance between v, w and u can differ by at most one. With these two conclusion that v must be direct parent of w in T which contradicts to the assumption.

#7

True, assuming the G is not connected, and the smallest part of G is S which have the least number of nodes. So in this case |S|<=n/2. Consider a node n is in S, the degree of n is at most n/2 – 1 even if it connected to all the rest nodes in S which contradicts to the assumption.

#9

When applying BFS to the G from node s, the node t is on the ith layer of the tree T and i>n/2. If every layer of T has at least two nodes then the total number of nodes is bigger than n, so there are at least one layer only has 1 node. And if we delete this node, there would be no path from s to t.

#10

fun(Graph G, node s, end t, number of path NOP)

{

Find adjacent node in G of s and put them in stack S.

Delete s from G so G’ = G-{s}

while( S is not null)

{

Pop n from S

If n == t then NOP++ then return

fun(G’, n, t)

}

}

int nop=0

execute fun(G, s, t, &nop);

then the nop is the number of shortest path from node s to node t in graph G.

#12

First, build a graph with the information given by the interviews in the following way:

Giving two nodes Bi and Di for each Pi representing the birth and death of the person. Then adding edges(Bi, Di) for each person. For the first form of fact we add edge (Di, Bj) and for the second form of fact we add edge (Bi, Dj) and edge (Bj, Di).

After the construction of the graph G, we test if there is cycle in the G. Suppose G has cycle, which is impossible if given information is consistent in time, shows there are time conflict in G.

Or if G has no cycle, it shows the information given by interviews are consistent.