Compensation Assignment 2
Final solutions due: Sat, Nov 5, 23:59
In-lab solutions due: Fri, Nov 4, 17:15

In this compensation assignment, we will first read in a directed acyclic graph (DAG) with numNodes nodes. The nodes are assumed to be identified by the natural numbers 1, 2, ... numNodes. The graph itself will be provided to you as a set of edges in a file in the following format:

```
numNodes
DirectedEdge1_source  DirectedEdge1_target
DirectedEdge2_source  DirectedEdge2_targer
...
```

For example, if our file has the following content

```
5
1 4
2 3
```

then there are 5 nodes in the graph numbered 1, 2, 3, 4, 5. There is a directed edge from node 1 to node 4, and from node 2 to node 3. There are no other edges in the graph. Note than node 5 has no incoming or outgoing edges in this graph.

A finite directed acyclic graph always has a non-empty set of source nodes, i.e. nodes with no incoming edges, and a non-empty set of sink nodes, i.e. nodes with no outgoing edges. Your first task will be to print a space separated list of all source nodes (i.e. no nodes with no incoming edges) in the DAG.

We now define a "similarity" relation on the nodes of the DAG as follows. Two nodes n1 and n2 are said to be similar iff one of the following conditions is satisfied:

- Both n1 and n2 are source nodes, and the user specifies n1 and n2 as similar.
- Both n1 and n2 are non-source nodes, and for every node m1 that has an edge to n1, there is a node m2 that has an edge to n2, where m1 and m2 are similar, and vice versa.
- If n1 has a single incoming edge from n2, and there are no other incoming edges of n1, then n1 and n2 are said to be similar.
- No other nodes n1 and n2 are similar.

You are now required to write a program that takes a set of pairs of similar source node identifiers from the user as input, computes all pairs of similar nodes in the graph and prints out the graph obtained by collapsing the symmetric, transitive closure of all pairs of similar nodes in a file named outgraph.txt. When printing the graph use the same file format as indicated earlier, but with identifiers of nodes clearly printed space-separated after a colon after the number of nodes. Moreover, you must always use the maximum identifier of all collapsed nodes to denote the collapsed node. You need not draw self-edges between collapsed nodes.

You are free to write your own .h and .cpp files to implement the above functionality. **There must be a file named main.cpp in the set of files you submit. We will compile your code simply by running**
g++ -g main.cpp

and run the executable as ./a.out < graphFile.txt
where graphFile.txt is an ASCII text file representing the graph and pairs of similar source nodes.

Input format of file:

```
numNodes
DirectedEdge1_source  DirectedEdge1_target
DirectedEdge2_source  DirectedEdge2_targer
...
-1 -1
Pair1 of similar source nodes
Pair2 of similar source nodes
...
-1 -1
```

Output format of file

```
Source nodes: sourceNodeId1 sourceNodeId2 ...

Similar node pairs: (similarNodeId1, similarNodeId2) (similarNodeId2,
similarNodeId3), ...

numNodesInCollapsedGraph :  nodeId1 nodeId2 ...
DirectedEdge1_source  DirectedEdge1_target
DirectedEdge2_source  DirectedEdge2_targer
...
```

For example, the input file

```
5
1 4
2 3
-1 -1
```

```
1 2
-1 -1
```

specifies the same graph that we had earlier, but now also says the two source nodes 1 and 2 are similar.

In this case, your program should print out the following (please maintain this format):

```
Source nodes: 1 2 5

Similar node pairs: (1, 2) (4, 1) (3, 2)

Collapsed graph:
2: 4 5
```

i.e. there are two nodes numbered 4 and 5 and there are no edges in the graph.

You must upload a single .tgz file named <roll_number>_C2.tgz containining all files (no folders) required for your implementation.

We will simply run tar -zxvf on <roll_number>_C2.tgz and then compile using g++ -g main.cpp