University of Waterloo
CS 341 Algorithms Fall 2013
Assignment 4

Due: Wednesday, November 13, 2013, at 3:00pm.

1. Create an efficient algorithm that determines if a directed graph is rooted tree. A rooted tree is an acyclic, connected, directed graph which contains a vertex r such there is a unique path from r to every other vertex in the graph [10 points].

   Analyze and state the asymptotic worse-case running time of your algorithm, $O(|E|)$ where $|E|$ is the number of directed edges. You should pick a suitable representation for your graph [10 points].

2. Create an efficient algorithm that determines how many colours it takes to colour an undirected graph G where the degree of each vertex in G is at most two [10 points].

   Analyze and state the asymptotic worse-case running time of your algorithm, $O(|V|)$ where $|V|$ is the number of vertices [10 points].

3. Create an efficient algorithm that does two tasks to an undirected graph G: (1) eliminates multiple copies of edges by replacing them with a single edge and (2) replaces edges (u, v) and (v,w) by an edge (u,w) where v is an edge of degree two. Be aware that removing a vertex with degree two can create multiple copies of edges and that removing multiple copies of edges can create a vertex with degree two. [10 points].

   Analyze and state the asymptotic worse-case running time of your algorithm, $O(|V|, |E|)$ where $|V|$ and $|E|$ have their usual meanings [10 points].

4. Create an efficient algorithm that determines if for each vertex v in a graph there is a path from v to at most twenty other vertices. [10 points].

   Analyze and state the the asymptotic worse-case running time of your algorithm $O(|V|)$ [10 points].

5. One method of performing a topological sort on a directed acyclic graph is the following: repeat until the graph is empty: find a vertex of in-degree zero, output it, and delete it from the graph. Design an efficient algorithm for this idea [10 points].

   Analyze the $O(|V|, |E|)$ asymptotic worse-case running time of this algorithm and compare its running time to the method describe in Section 22.3 of the course text [10 points].

6. Create an efficient algorithm that given a graph $G(V, E)$, a spanning tree T for $G(V, E)$, an additional vertex v, and additional edges E_a adjacent to v, it creates a spanning tree for $G(V \cup \{v\}, E \cup E_a)$ [10 points].

   Analyze and state the asymptotic worse-case running time of your algorithm, $O(|V|, |E|, |E_a|)$. [10 points].

7. For each of a) Kruskal's algorithm, b) Prim's algorithm and c) Dijkstra's algorithm what happens if edges with negative weights are allowed. If the algorithm does not work properly give a case where it fails. [5 points for each algorithm]

8. Programming Assignment. Create an efficient program that given a graph as an adjacency list and a list of edge queries, labels each of the edges as one of {*tree-edge, forward-edge, back-edge, cross-edge, no-such-edge*}. For example, given the following input in stdin

```
1 : 4 2
2 : 5
3 : 5
4 : 2
5 : 4
```
$\leftarrow$ *blank line*
```
1 2
1 4
4 2
3 5
2 1
```
$\leftarrow$ *blank line*

the following output is produced.

```
1 2 tree-edge
1 4 forward-edge
4 2 back-edge
3 5 cross-edge
2 1 no-such-edge
```

Note: there are two blank lines delimiting the input. There is a single blank space between any two vertices and between a vertex and the colon. There are at most 2^{15} vertices and they are all labelled by unsigned ints.

In order to keep the labels consistent across all submissions do a DFS of the vertices in ascending order of their labels. Your code must provide meaningful error messages. [35 points for correctness/test cases, 5 for coding style]

In the written part of this question briefly a) state how you represented the edges and labels, b) how you processed the queries and c) analyze your algorithm in terms of $O(|V|, |E|, |Q|)$ where $|V|$ and $|E|$ have their usual meanings and $|Q|$ is the number of edges you made queries about [10 points].