CSE 202: Algorithm Design and Analysis Spring 2015

Problem Set 1

Instructor: Fan Chung Graham **Due on:** Wed April 15

Instructions

- For your proofs, you may use any lower bound, algorithm or data structure from the text or in class, and their correctness and analysis, but please cite the result that you use.
- If you do not prove that your algorithm is correct, we will assume that it is incorrect. If you do not provide an analysis of the running time, we will assume you do not know what the running time is.

Problem 1

The following statements may or may not be correct. For each statement, if it is correct, provide a short proof of correctness. If it is incorrect, provide a counter-example to show that it is incorrect. In the following, assume that the graph $G = (V, E)$ is undirected and connected. Do not assume that all the edge weights are unique unless stated.

1. If G has more than $|V| - 1$ edges, and there is a unique heaviest edge e, then e cannot be part of any minimum spanning tree of G.
2. The worst case time for a find operation in the Union-Find data structure can be $\Theta(\log n)$ even with path compression.
3. A lightest edge in a cut is present in all minimum spanning trees of G.
4. If e is a heaviest edge in a cycle, then e cannot occur in any minimum spanning tree of G.
5. Prim's algorithm works correctly when some of the edge weights are negative.

Grading: Each subproblem is worth 4 points: 2 points for correctly answering True or False, 2 points for a proof or counterexample.

Problem 2

Given a list of n positive integers $d_1, d_2, \ldots, d_n$, we want to efficiently determine whether there exists an undirected graph $G = (V, E)$ whose nodes have degrees precisely $d_1, \ldots, d_n$. That is, if $V = \{v_1, \ldots, v_n\}$, then the degree of v_i should be exactly d_i. We call $(d_1, \ldots, d_n)$ the degree sequence of G. This graph G should not contain self-loops (edges with both endpoints equal to the same node) or multiple edges between the same pair of nodes.

1. Give an example of d_1, d_2, d_3, d_4 where all the $d_i \leq 3$ and $d_1 + d_2 + d_3 + d_4$ is even, but for which no graph with degree sequence (d_1, d_2, d_3, d_4) exists.
   Grading: 5 points
2. Suppose that $d_1 \geq d_2 \geq \ldots \geq d_n$ and that there exists a graph $G = (V, E)$ with degree sequence $(d_1, \ldots, d_n)$. We want to show that there must exist a graph that has this degree sequence and where in addition the neighbors of v_1 are $v_2, v_3, \ldots, v_{d_1+1}$. The idea is to gradually transform G into a graph with the desired additional property.
   (a) Suppose the neighbors of v_1 in G are not $v_2, v_3, \ldots, v_{d1+1}$. Show that there exists $i < j \leq n$ and $u \in V$ such that $(v_1, v_i), (u, v_j) \notin E$ and $(v_1, v_j), (u, v_i) \in E$.

(b) Specify the changes you would make to G to obtain a new graph $G_0 = (V, E_0)$ same degree sequence as G and where $(v_1, v_i) \in E$.

(c) Now show that there must be a graph with the given degree sequence but in which v_1 has neighbors $v_2, v_3, \ldots, v_{d_1+1}$.

Grading: 10 points

3. Using the result from part (2), describe an algorithm that on input $d_1, \ldots, d_n$ (not necessarily sorted) decides whether there exists a graph with this degree sequence. Your algorithm should run in time polynomial in n and in $m = \sum_{i=1}^{n} d_i$.
Grading: 5 points – make sure to include runtime analysis!

Problem 3

Consider the traffic camera problem: the police department would like to set up traffic cameras to catch traffic offenders on every road. However, the department budget is being cut, and there is not enough money to place a camera at every intersection. Fortunately there is no need to place cameras at every intersection because a camera placed at an intersection can monitor all roads adjacent to this intersection. Given a network of roads, the traffic camera problem is to find the minimum number of traffic cameras that need to be placed at intersections (as well as the intersections where we need to place these cameras) such that all roads can be monitored. (Note that each road is adjacent to two intersections, one at each end.)

1. First, write down a graph-theoretic formulation of the problem. Do not forget to completely specify the inputs, any specific constraints, and the required outputs.
Grading: 6 points for all required components

2. Below is a greedy algorithm for this problem. Does this algorithm work correctly? If so, write down a proof of correctness. If not, write down a counterexample.

```
1: Initialize: Intersection set $S = \emptyset$. Road set $R$ is the set of all roads.
2: while $R$ is non-empty do
3:     Pick an arbitrary road $r \in R$, and pick an intersection $v$ adjacent to $r$. Add $v$ to $S$.
4:     Delete from $R$ all roads $r'$ that are adjacent to $v$.
5: end while
6: Output $S$.
```

Grading: 7 points: 3 points for correctly answering whether the algorithm works, 4 points for a proof of correctness or counterexample

3. Suppose we modify the greedy algorithm to the algorithm below. Does this algorithm work correctly? If yes, write down a proof of correctness. If not, write down a counterexample.

```
1: Initialize: Intersection set $S = \emptyset$. Road set $R$ is the set of all roads.
2: while $R$ is non-empty do
3:     Pick an intersection $v$ from the set of intersections that is adjacent to the maximum number of roads in $R$. Add $v$ to $S$.
4:     Delete from $R$ all roads $r'$ that are adjacent to $v$.
5: end while
6: Output $S$.
```

Grading: 7 points: 3 points for correctly answering whether the algorithm works, 4 points for a proof of correctness or counterexample