Homework #3: Minimum Spanning Trees and Priority Queues
Due Date: Tuesday, February 24, 1998 (9:00am)

Reading

Read Ch. 7, Ch. 24 of CLR.

Practice

Recall, these exercises are purely for your own practice. You need only turn in the official problems.

- Ex. 24.1-1 of CLR
- Ex. 24.1-3 of CLR
- Ex. 24.1-5 of CLR
- Ex. 24.2-3 of CLR

Collaboration Policy

You may collaborate with others on this entire homework. However, please re-read the policy guidelines from Handout #1.

Problems

1. CLR 24.1-4
   Give a simple example of a graph such that the set of all edges that are light edges crossing some cut in the graph does not form a minimum spanning tree.

2. CLR 24.1-6
   a. Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut.
   b. Show that the converse is not true by giving a counterexample.

3. CLR 20-2
   Chapter 24 presents two algorithms to solve the problem of finding a minimum spanning tree of an undirected graph. Here, we shall see how mergable heaps can be used to devise a different minimum-spanning-tree algorithm.

The following pseudocode, which can be proven correct using techniques from Section 24.1, constructs a minimum spanning tree T (*you do not have to prove the correctness here*). It maintains a partition $\{V_i\}$ of the vertices of V and, with each set V_i, a set

$E_i \subseteq \{(u, v) : u \in V_i or v \in V_i\}$

of edges incident on vertices in V_i.

```
MST-Mergeable-Heap(G)
 1  T ← ∅
 2  for each vertex v_i ∈ V[G]
 3      do V_i ← {v_i}
 4         E_i ← {(v_i, v) ∈ E[G]}
 5  while there is more than one set V_i
 6      do choose any set V_i
 7         extract the minimum-weight edge (u, v) from E_i
 8         assume without loss of generality that u ∈ V_i and v ∈ V_j
 9         if i ≠ j
10         then T ← T ∪ {(u, v)}
11              V_i ← V_i ∪ V_j, destroying V_j
12              E_i ← E_i ∪ E_j
```

Describe how to implement this algorithm using the mergeable-heap operations given in Figure 20.1. Give the running time of your implementation, assuming that the mergeable heaps are implemented by binomial heaps (*Please be very specific about which lines in the code above correspond to which specific mergeable heap operations.*)

4. Suppose edge weights in a graph are integers in the range from 1 to W. Show that you can make Prim's algorithm run in $O(WE)$ time. For what values of W does this result improve on the bound using Fibonacci Heaps? (*Hint: We do not want you to re-design Prim's algorithm. You should only re-design the priority queue implementation to achieve the new bound. To get going, think about what you could do if $W = 1$. What if $W = 2$?*)

Extra Credit

CLR 24.1-8. Let T be a minimum spanning tree of a graph G, and let L be the sorted list of the edge weights of T. Show that for any other minimum spanning tree T' of G, the list L is also the sorted list of edge weights of T'.