CS140 - Group Assignment 8

Due: Friday, Nov. 4 at 11:59pm

1. Kruskal's algorithm for finding a minimum spanning tree is typically implemented using a disjoint-set data structure. This is a data structure that supports the operations:

   makeSet(x): creates a new set containing x as its only element

   findSet(x): returns the set representative (identifier) for the set containing x

   union(x,y): merges the set containing x and the set containing y so that, after calling union(x,y), findSet returns the same set representative for any element that was either in the same set as x or in the same set as y. (Note that typically it is assumed that findSet(x) and findSet(y) are called first and union(x,y) is invoked only if x and y are in different sets.)

   One implementation for disjoint-sets uses a singly-linked list for each set. Every element maintains both the standard "next" pointer and, additionally, a set-representative pointer back to the head of the list. The head of the list is then the set representative for that list. Note that when you call union, you not only need to connect the two linked lists, but you also need to update all of the set representative pointers in one of the lists. With this implementation for the disjoint-set data structure, what is the running time for each operation? Can you do better with an amortized analysis (assume a combination of m operations where n are make-sets (where $m > n$)? **Hint: if we knew the size of the two sets we were merging, could we do something more intelligent?**

2. We're almost 2/3rds of the way through the course.

   (a) What has been the most challenging thing about this course?

   (b) Are there any topics that you hope we'll cover in the remaining part of the course?

3. Was everyone in the group at the meeting and, if not, who was missing?