Problem Set #3: Heaps, Sorting, and MST

Handed out on Tues, Mar 4, due on Tues, Mar 11 at the beginning of class. Remember: write your own answers and use English or pseudocode when algorithms are requested. Late homeworks will not be accepted (turn in whatever you have).

**Problem 0.** Consider the two heaps shown below.

![](image)

1. Label each of the above heaps with their $npl$ values.
2. Show the result of merging these two heaps assuming they are leftist heaps.
3. Show the result of merging them assuming they are skew heaps.

**Problem 1.** Describe a data structure that will allow you to delete any item in $O(\log n)$ time given only its key, while still supporting $\text{extract\_min}$ and $\text{insert}$ in $O(\log n)$ time.

**Problem 2.** Explain why it is not possible to design a data structure that is guaranteed to be able to perform a series of $n$ $\text{makeheap}$, $\text{insert}$, $\text{extract\_min}$ operations in amortized constant time. In other words, the total time for the $n$ operations cannot be bounded by $O(n)$.

**Problem 3.** If $e$ is some maximum-weight edge on a cycle of $G$, argue why it is always possible to construct a minimum spanning tree that does not contain edge $e$.

**Extra Credit Problem.** Explain how to modify AVL trees to implement a List ADT that supports the following operations. Each operation should run in $O(\log n)$ time, where $n$ is the number of elements currently in the list. There are no “gaps” allowed in the list: if it contains an item at index $i$ then it contains items at indices $0\ldots i-1$.

a. $\text{access}(L, i)$: return the $i$th item in the list.

b. $\text{insert}(L, i, k)$: insert value $k$ at the $i$th position in the list, shifting items at indices $i+1$ and above up by one.

c. $\text{delete}(L, i)$: remove the $i$th item in the list and shift all items at indices $i+1$ and above down by one.

You solution should consist of an explanation (English or pseudocode) for how to implement each of the above operations.