Problem Set 2, Written Part
Balanced Binary Search Trees and MSTs

Due: 11:55pm Thursday, October 25, 2018

---

1 Number of nodes and balance (10 marks)

For each sequence of insertions below, start with an empty 2-3 tree.

Using distinct letters of the alphabet, devise a sequence of 7 insertions that maximizes the number of nodes in a 2-3 tree. Draw the final corresponding left-leaning red-black tree (you need not draw any intermediate trees).

Using distinct letters of the alphabet, devise a sequence of 8 insertions that minimizes the number of nodes in a 2-3 tree. Draw the final corresponding left-leaning red-black tree (again, you need not draw any intermediate trees).

Informally discuss the relationship between the number of nodes in a 2-3 tree (for fixed n) and the level of balance in the corresponding left-leaning red-black tree. You can interpret "level of balance" as the difference in the length of the longest root-to-leaf path and the length of the shortest root-to-leaf path in a left-leaning red-black tree.

2 A hybrid MST algorithm (10 marks)

Consider the following hybrid "Kruskal-Prim" MST algorithm that constructs a set of edges A by alternating between *Kruskal steps* and *Prim steps*:

Algorithm 1: Hybrid Algorithm

```
A = ∅
while |A| < |V| − 1 do
    if |A| is even then
        // Kruskal step
        Add to A the minimum weight edge that does not induce a cycle in the
         subgraph induced by A.
    else
        // Prim step
        Randomly select a connected component from the subgraph induced by A.
         Then add to A the minimum weight edge among all edges that would connect
         this connected component to another connected component.
    end
end
return A
```

Does this algorithm construct a minimum spanning tree? If you say no, show a counterexample for why the algorithm does not work. If you say yes, prove that it does. You may make use of the Cut Property Theorem, restated below.

Cut Property Theorem. *Let A be a subset of the edges of the minimum spanning tree of G. If $(S, V \setminus S)$ is a cut for which no edge of A crosses the cut, then the minimum weight edge crossing the cut is contained in the minimum spanning tree.*

3 Uniqueness of MSTs (10 marks)

Let G be a connected, weighted graph. Prove that if all the edge weights are distinct, then the minimum spanning tree of G is unique.

*** Optional extra credit question: Non-distinct edge weights ***

In class, we proved that both Kruskal's algorithm and Prim's algorithm correctly find the unique minimum spanning tree when the edge weights are distinct. Pick one of these algorithms and prove that it still correctly finds a minimum spanning tree even when the edge weights are not all distinct.

4 Quick-Union with Union-by-Rank (10 marks)

In class, we saw the Weighted Quick-Union algorithm, which performs UNION in the following way: make the root of the tree containing less elements the child of the root of the tree containing more elements (with ties resolved arbitrarily). Let's call this version of UNION "Union-by-Weight".

We will now look at a similar algorithm, which instead uses a version of UNION we will call "Union-by-Rank". Union-by-Rank proceeds via two cases:

1. Suppose we are taking the union of two trees T_1 and T_2 with heights h_1 and h_2 respectively, with $h_1 > h_2$. Then the root of T_2 becomes a child of the root of T_1. In this case, the resulting tree has height equal to the height of T_1.
2. If the two trees have the same height, the root of one of the trees becomes a child of the root of the other tree. In this case, the resulting tree has height one greater than each of the original trees.

Let n be the number of vertices. Prove that when using Quick-Union with Union-by-Rank, a call to CONNECTED takes time $O(\log n)$ in the worst case.