Problem Set 3

Due: 11:59pm, Monday, February 26th

See homework submission instructions at http://sites.fas.harvard.edu/~cs124/cs124/problem_sets.html

Problem 4 is worth 40% of this problem set, and problems 1-3 constitute the remaining 60%.

For each problem where you are asked to give an algorithm, more points are given for asymptotically faster algorithms. In judging the number of points to award a correct solution, we only consider the running time in asymptotic notation, i.e. a writeup of an algorithm taking 1000\(n^2\) steps versus one taking .01\(n^2\) steps would receive the same number of points — both would be simply treated as \(\Theta(n^2)\)-time solutions.

Problem 1

You are given \(n\) boxes labeled 1, 2, …, \(n\), each initially in its own stack. You would like a data structure supporting the following two operations:

- **move**(\(x,y\)): lift the stack containing box \(x\) and place it directly on top of the stack containing box \(y\).

- **under**(\(x\)): return the number of boxes under \(x\) in its stack.

Describe a data structure which supports any sequence of \(m\) move and under operations efficiently and justify its correctness. Full credit will be given to solutions requiring time \(O((m + n) \log^* n)\).

Problem 2

In class we described the Bellman-Ford algorithm, which on input \((G, s)\), for \(G\) a weighted directed graph possibly with negative edge weights and \(s\) a vertex in \(G\), returns either

1. an array dist where \(\text{dist}[u]\) is the length of the shortest path from \(s\) to \(u\) (if no negative-weight cycle is reachable from \(s\)), or

2. “negative cycle detected”, if a negative-weight cycle is reachable from \(s\)

Such a guarantee, though, is not always satisfying. See for example Figure 1. That is, it may be possible that although \(s\) can reach some negative-weight cycle, the shortest path distance from \(s\) to some other vertex \(t\) may still be well-defined.
Figure 1: The negative-weight cycle \( A \rightarrow B \rightarrow C \rightarrow A \) is reachable from \( s \), but the shortest path distance from \( s \) to \( D \) is still well-defined.

Show how to solve the following problem: given a weighted directed graph \( G \) possibly with negative edge weights and two vertices \( s, t \) in \( G \), output the length of the shortest path from \( s \) to \( t \) if it is well-defined, else output \(-\infty\) if it can be made arbitrarily small by taking advantage of a negative cycle before reaching \( t \). You should demonstrate that your algorithm is correct, as well as bound its running time and memory consumption.

**Problem 3**

You are given two undirected trees \( T_1, T_2 \) on the same vertex set \( \{1, \ldots, n\} \). You may assume that the trees are given in adjacency list representation, where each vertex’s list of neighbors is given in a linked list in increasing sorted order by vertex ID. You would like to convert \( T_1 \) into \( T_2 \) by a sequence of the minimum number of operations possible. Each operation choose one edge to delete in \( T_1 \), followed by adding a new edge to \( T_1 \); it is required that the resulting graph after each operation must remain a tree. Devise an algorithm which computes the minimum number of operations to turn \( T_1 \) into \( T_2 \), and also prints those operations in order (each operation should be described as “delete edge \((a, b)\) then add edge \((c, d)\)”), and justify its correctness. Full credit is given for running time \( O(n \log^* n) \). **Hint:** first determine the connected components of \( T_1 \cap T_2 \), and then work your way up from the leaves.

**Programming Problem**