TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in algorithms
Spring Semester, 2012/2013

Homework 1, March 10, 2013

Due on Sunday March 24 (so we could check it over passover vacation). Please put a hardcopy in my mailbox and keep a copy of the homework!

1. Let $G = (V, E)$ be an undirected graph in which each edge is either blue or black. Assume$|V| = n$ and $|E| = m$. Let ℓ be the minimum number of blue edges in a spanning tree of G and let h be the maximum number of blue edges in a spanning tree of G. Recall the algorithm described in class for finding a minimum spanning tree containing q blue edges for every $\ell \leq q \leq h$. Describe how to modify this algorithm so that it runs in $O(T(MST) + n \log n)$ time where $T(MST)$ is the running time of the fastest algorithm for finding a minimum spanning tree. (The version described in class ran in $O(T(MST) + m \log n)$ time.) Prove that your algorithm is correct.

2. A semi-splaying is a variation of splaying where in the zig-zig step we only do the top rotation (on the edge (y, z) in the slides) and continue from y (rather than from x). Prove that the access lemma holds for semi-splaying.

3. Show how to extend dynamic trees to support the operation $evert(v)$. This operation changes the actual tree containing v such that following the operation v is the root of this tree. The directions of all the edges from v to the previous root of the tree are reversed. An efficient implementation should run in $O(\log n)$ time. Do not hurt the logarithmic running time of any of the other operations.

4. Assume that each edge e of a dynamic tree has a fixed *weight*, $w(e)$, associated with it, which is given with e when e is inserted by a link operation and never changes (do not confuse $w(e)$ with the cost of e, $c(e)$, which is a different thing). Show how to extend dynamic trees to support the operation $sum_w(v)$ that returns $\sum_{e \in P} w(e) \cdot c(e)$ where P is the path from v to the root. Note that for the special case where $w(e) = 1$ for every e, $sum_w(v)$ is just the sum of the costs of the edges on the path from v to the root. An efficient implementation should run in $O(\log n)$ time. Do not hurt the logarithmic running time of any of the other operations.

5. On a set of n nodes we perform a sequence of operations, each of which is one of the followings.
(a) $insert(u, v)$: Adds an edge between u and v.
(b) $delete - oldest$: Removes the edge that was inserted first among the edges currently in the graph.
(c) $connected(u, v)$: Answers true if there is a path from u to v in the current graph. Otherwise answers false.

Describe a data structure that supports these operations and analyze its performance. (A structure supporting each operation in $O(\log n)$ amortized time would receive maximum score.)