TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Fall Semester, 2005/2006

Homework 5, January 26, 2005

Due on Thursday February 9. Don't forget to keep a copy of the homework.

1. You are given arbitrary intervals on the real line each has a priority assigned to it. Build a data structure (as simple as you can) that supports insertions of new intervals and queries of the form: "given a point find an interval of maximum priority containing it". (Try to find a data structure in which each operation takes $O(\log n)$ time, and the space is linear.)

2. Assume that the input to Dijkstra's algorithm is a graph such that all the weights of the arcs are positive integers each not larger than C. In this case you can implement the heap in an array such that a vertex with key $d(v)$ is stored in a list hanging off entry $d(v)$ of the array. (The list hanging off entry i is empty if there are no vertices with key i in the heap.)
1) Complete the description of this implementation and prove that its running time and space consumption is $O(m + nC)$.
2) Change the implementation such that it requires only $O(m + n + C)$ space.
3) Find an alternative implementation whose running time is $O(m + n\sqrt{C})$.

3. Consider an implementation of Fibonacci heaps without cascading cuts (all other details are as shown in class, the only difference is that delete and decrease-key just cut the subtree and do not continue with cascading cuts). For any large enough m show a sequence of m operations on heaps of size at most n such that the average cost of an operation is as high as possible. (By m large enough we mean larger even than some function of n.)

4. Use persistent search trees to solve the following problem. Given a set S of n points in the plane, preprocess them to build a data structures which given a triple x_1, x_2, y_1 allows to report efficiently all the points $(x, y) \in S$ such that $x_1 \leq x \leq x_2$ and $y \geq y_1$. Analyze the space, preprocessing time, and query time of your data structure.

5. Augment the redundant binary counters shown in class to support both increment and decrement in $O(1)$ worst case time. Describe the algorithm that decrements the counter and establish its correctness.