Massachusetts Institute of Technology | Handout 11
6.046J/18.410J: Introduction to Algorithms | March 6, 2001
Professors Piotr Indyk and Madhu Sudan

Problem Set 4

This problem set is due in class on *Tuesday, March 13th* at the end of class.

There are four problems. **Each problem is to be done on a separate sheet (or sheets) of paper. Separate problems should *not* be stapled together.** Mark the top of **each and every sheet** with your name, 6.046J/18.410J, the **problem number**, your **recitation section, your TA's name**, and the date. It is very important to hand in problems separately and to indicate your recitation section **and** TA name on every sheet. Failure to do so will result in delays in returning your work and possible confusions with entering your grade. All problem sets throughout the semester should be handed in this way.

Reading: Chapters 10, 12 of CLR.

Problem 4-1. Improving Lower Bounds by Using Randomization

Suppose you are given an array A of size n that contains either all zeros or $n/2$ zeros and $n/2$ ones, and your problem is to determine whether A contains any ones.

(a) Give a lower bound on the worst-case running time of any deterministic algorithm that solves this problem.

(b) Give a randomized algorithm that runs in $O(1)$ time and gives the right answer with probability at least $3/4$, where the probability is taken over the random choices of the algorithm. (That is, your algorithm is allowed to be wrong, but it cannot be wrong with probability greater that $1/4$ on *any* input.) Prove your answer.

(c) Design a randomized algorithm which never makes any errors. Can you do better than the deterministic case? If so, under what circumstances?

Problem 4-2. Selection in linear time

In the algorithm SELECT, the input elements are divided into groups of 5. Will the algorithm work in linear time if they are divided into groups of 7? How about groups of 3?

Problem 4-3. Data Structures for Order Statistics

Design a data structure that can perform the following operations in $O(1)$ time per operation. You can assume free initialization of an array $A[1 \dots n]$ to 0.

- insert(a): insert a from $1 \dots n$ into the structure and return a pointer to a.
- decrement(ptr): decrement the number pointed to by ptr (unless it is equal to 0 in which case return ERROR).

- min: return the minimum element.
- max: return the maximum element.

Problem 4-4. Uniform Families of Hash Functions

Let $\mathcal{H}$ be a family of hash functions $h : X \mapsto \{0, \cdots, m-1\}$ such that for any $x \in X$ and $i \in \{0, \cdots, m-1\}$, the probability that $h(x) = i$ when h is taken randomly from $\mathcal{H}$ is exactly $1/m$. Is $\mathcal{H}$ a universal family? Prove that it is or give a counterexample.