Introduction to Algorithms
Massachusetts Institute of Technology
Professors Erik Demaine and Madhu Sudan

Problem Set 2

This problem set is due in lecture on Wednesday, September 27, 2006. The homework lab will be held 7–9 P.M. on Monday, September 25, 2006, in room 32-124.

Reading: Sections 5.1–5.3 and Chapters 6, 7, and 8.1–8.3.

Both exercises and problems should be solved, but only the problems should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered in the exercises.

Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date and the names of any students with whom you collaborated.

You will often be called upon to “give an algorithm” to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of the essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. At least one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the running time of the algorithm.

Remember, your goal is to communicate. Full credit will be given only to correct solutions which are described clearly. Convoluted and obtuse descriptions will receive low marks.

Exercise 2-1. Do Exercise 5.2-4 on page 98 in CLRS.

Exercise 2-2. Do Exercise 6.5-7 on page 142 in CLRS.

Exercise 2-3. Do Exercise 7.2-1 on page 153 in CLRS.

Exercise 2-4. Do Exercise 8.1-3 on page 168 of CLRS.

Problem 2-1. Sorting can be boring!

A comparison sort is boring if, for infinitely many \( n \), when sorting some sequence of \( n \) items, at least one item (or any copy of it) gets compared \( \Omega(n) \) times. In other words, a comparison sort is boring if there exists a positive constant \( c > 0 \) such that, for infinitely many values of \( n \), there is a sequence of \( n \) items on which the sorting algorithm compares one item at least \( c n \) times. Assume throughout this question that all \( n \) items must be distinct.
(a) Is Mergesort boring? Prove your answer.

(b) Is Quicksort boring? Prove your answer.

(c) Is Heapsort boring? Prove your answer.

(d) Bonus Part: This part is optional. However, solving this part does not exempt you from solving the other parts of this question. Modify any one of the three sorting algorithms above that you find boring, in such a way that it is no longer boring (but keep the time to sort $n$ items $O(n \log n)$—in expectation if randomized).

Problem 2-2. Sorting in Other Models

The decision-tree model is a good model to capture comparison sorts, but it really captures other algorithms as well. In the first two parts of this problem, we consider Radix Sort with a radix of 2. In this case, the internal nodes of the decision tree can be annotated by a pair $i : j$ indicating that the bit being examined is the $j$th bit of the $i$th input item, and we take the left branch if this bit is 0 and the right branch if this bit is 1. If all possible permutations are possible as input, then the decision tree must have a leaf for each such permutation, so the depth of the decision tree must be at least $\log(n!) = \Omega(n \log n)$. Because the depth of the decision tree is a lower bound on the worst-case sorting time, the worst-case running time of Radix Sort with a radix of 2 must be $\Omega(n \log n)$.

On the other hand, we know that Radix Sort with a radix of 2 sorts $b$-bit integers in $O(nb)$ worst-case time. This time bound is $o(n \log n)$ if $b = o(\log n)$. So the decision-tree lower bound cannot give an $\Omega(n \log n)$ worst-case lower bound in this case.

(a) Explain why the $\Omega(n \log n)$ decision-tree lower bound fails when $b = o(\log n)$.

(b) What lower bound can you derive on sorting $n$ $b$-bit integers, using the decision-tree model described above? State your lower bound using $\Omega$ notation in terms of both $n$ and $b$.

On a very fast planar chip, the time to access the $i$th memory location is bounded by the speed of light, and is thus proportional to the distance between the CPU and that memory location. If a conceptually linear memory is arranged in a spiral around the CPU, the time to access the $i$th location is then $\Theta(\sqrt{i})$. In the next three problem parts, you will consider the HMM$_\sqrt{x}$ model of computation, where it costs $\sqrt{x}$ “time” to access memory location $x$, for any integer $x \geq 1$. (HMM stands for Hierarchical Memory Model.) In this model, we assume that the cost of performing a computation (e.g., a comparison) is included in the cost of accessing its inputs.

(c) On the HMM$_\sqrt{x}$, what is the expected running time of RANDOMIZED-QUICKSORT? (Give your answer using $\Theta$ notation.) Assume that the elements of the array $A[1..n]$ occupy memory locations 1 through $n$, respectively, and that the $\Theta(1)$ index variables $(i, j, p, q, r)$ can be accessed for free.
(d) Modify RANDOMIZED-QUICKSORT so that it sorts \( n \) items in \( O(n^{3/2}) \) expected time on the HMM\( \sqrt{n} \). Analyze the expected time taken by your algorithm. Assume that the \( n \) inputs are initially in memory locations 1 through \( n \), and that the outputs must be placed in nondecreasing order into locations 1 through \( n \).

*Hint:* One way to analyze this algorithm is to modify the analysis from Lecture 4 of RANDOMIZED-QUICKSORT. This yields a recurrence of the form

\[
T(n) \leq \frac{2}{n} \sum_{k=1}^{n-1} T(k) + f(n).
\]

As in lecture, this recurrence can be solved by the method of substitution, now using the guess that, for some positive constants \( c \) and \( n_0 \), \( T(n) \leq c n^{3/2} \) for all \( n \geq n_0 \). Of course, you should verify the guess by induction. When evaluating tricky sums, you might find it helpful to use inequality (A.11) on page 1067 of CLRS.

(e) Is there an algorithm that sorts \( n \) distinct items in \( o(n^{3/2}) \) expected time on the HMM\( \sqrt{n} \)?