Introduction to Algorithms September 12, 2003
Massachusetts Institute of Technology 6.046J/18.410J
Professors Shafi Goldwasser and Silvio Micali Handout 8

Problem Set 2

This problem set is due **in lecture** on **Wednesday, September 24.**

Reading: Chapters §2.3, §5.1–5.3, §8.1–8.3, §28.2, §30.1.

There are **four** problems. Each problem is to be done on a separate sheet (or sheets) of three-hole punched paper. 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 your essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudocode. English explanations should be used liberally in your pseudocode. For convenience, pseudocode may use standard arithemetic and logical operations, as well as loops and data structures like arrays. Pseudocode should be understandable to anyone who can program–it should **not** be ready to compile!
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. Graders will be instructed to take off points for convoluted and obtuse descriptions.

Problem 2-1. Goldilocks and the n bears.

> *Once upon a time, there was a little girl named Goldilocks. She went for a walk in the forest. Pretty soon, she came upon a house. She knocked and, when no one answered, she walked right in. At the table in the kitchen, there were three bowls of porridge. Goldilocks was hungry. She tasted the porridge from the first bowl. "This porridge is too hot!" she exclaimed. So, she tasted the porridge from the second bowl. "This porridge is too cold," she said. So, she tasted the last bowl of porridge. "Ahhh, this porridge is just right," she said happily and she ate it all up.*

In an unfortunate accident at a laboratory in Building 68, a little girl named Goldilocks wandered into a human cloning machine. As a result, there are now n little Goldilocks (Goldilocks$_1$, Goldilocks$_2$, $\ldots$) wandering around the halls of MIT. Because cloning remains an imperfect technology, each Goldilock$_i$ has a *temperature preference* t_i distinct from that of all the other Goldilocks.

Luckily, you also have managed to find n bowls of porridge, where bowl j is kept at temperature b_j. It is a great stroke of fortune that for each Goldilocks$_i$, there is exactly one bowl j so that $b_j = t_i$, and for each bowl j there is exactly one Goldilocks$_i$ whose temperature preference $t_i = b_j$. That is, the two sets $\{b_j : 1 \leq j \leq n\}$ and $\{t_i : 1 \leq i \leq n\}$ are equal. Your job is to match Goldilocks$_i$ to a bowl j of porridge such that $b_j = t_i$.

When you give bowl j to Goldilocks$_i$, she says:

$$\begin{cases} \text{“This porridge is too hot!”} & \text{if } t_i < b_j \\ \text{“This porridge is too cold!”} & \text{if } t_i > b_j \\ \text{“This porridge is just right!”} & \text{if } t_i = b_j . \end{cases}$$

Call any one such tasting a *trial*. You may only use trials to get information about temperatures of the bowls or temperature preferences of the Goldilocks. *You may **not** directly compare the temperatures of two bowls or the temperature preferences of two Goldilocks.*

(a) Give a randomized algorithm for which the expected number of trials is $O(n \log n)$.

(b) Suppose that when you give Goldilocks$_i$ bowl j with $b_j \neq t_i$, she spits out a spoonful and says "Yuck!" (without indicating whether it's too hot or too cold). Give the fastest algorithm you can to match Goldilocks to bowls in this scenario.

Problem 2-2. Hitting the target.

Give an $O(n)$ algorithm for the following problem: Given a target integer T and an array A of n integers such that $0 \leq A[i] \leq 65536$ for every $1 \leq i \leq n$, determine if there exists a pair $i, j \in \{1, \ldots n\}$ such that $A[i] + A[j] = T$.

Problem 2-3. Beating the sorting lower bound?

(a) You are a spammer. You have an array A with n entries, each containing information about a person. Each person is from one of k different families, and there are exactly n/k members of each family in the array. (You may assume that each family has a unique last name.) You would like to sort the n entries in alphabetical order according to $\langle$*last name, first name*$\rangle$, using a comparison-based sorting algorithm. You know how to sort using $\Theta(n \log n)$ comparisons, but since A is truly massive in size, you would like to sort the data faster. So you hire King Arthur Anderson Consulting to address your problem.

The Anderson Consultant claims that he can sort A using $o(n \log n)$ comparisons! He says he can do this because the array is partially sorted in the following sense:

(i) Entries $A[1, \ldots, n/k]$ correspond to the members of one of the families, entries in $A[n/k + 1, \ldots, 2n/k]$ correspond to the members of another family, etc. In other words, entries in $A[(i-1)n/k + 1, \ldots, i \cdot n/k]$ for $i, 1 \leq i \leq k$ correspond to all the n/k members of one of the k famillies.

(ii) The families are *already* in alphabetical order according to last name. In other words, the last name of the family $A[1, \ldots, n/k]$ comes alphabetically before the last name of the family $A[n/k+1, \ldots, 2n/k]$, etc. In general, the last name of the family $A[(i-1)n/k + 1, \ldots, i \cdot n/k]$ comes alphabetically before the last name of the family $A[i \cdot n/k + 1, \ldots, (1+1)n/k]$ for all $i, 1 \leq i < k$.

You verify that (i) and (ii) are indeed correct. Should you believe his claim for all values of k? If not, for what values of k is it possible to sort A using $o(n \log n)$ comparisons? Prove your answer.

(b) Anderson goes bankrupt, and loses all of your data. You have a scrambled backup copy of A, so you can no longer assume that (i) and (ii) hold. You hire another consultant from ConsultingAgency.com to sort the data. She invents new algorithms that are not comparison based. She claims that she can sort your data in $o(n)$ time. Should you believe her? Prove your answer.

(c) After proceedings in bankruptcy court, you recover your partially sorted array. The ConsultingAgency.com consultant says that she can also use her algorithms on this partially sorted array for any values of k. For what values of k should you believe her claim? Prove your answer.

Problem 2-4. Optimal investment strategies (with insider trading).

You are given an array A of n integers. Entry $A[i]$ is the stock price of BIM on day i. Your goal is to make the most money that you can by buying BIM once and subsequently selling once during this n-day period.

For any $r \geq \ell$, you can buy shares of BIM on day ℓ at price $A[\ell]$ and sell on day r at price $A[r]$. You need to find two indices ℓ and r such that $r \geq \ell$ and $A[r] - A[\ell]$ is maximized.

It is easy to find the best ℓ and r in time $O(n^2)$, simply by iterating over all possible pairs ℓ and r. This question asks you to give a more efficient algorithm.

(a) Give an $O(n \log n)$ algorithm to solve this problem. Your algorithm should take A as input, and produce two indices ℓ and $r \geq \ell$ such that $A[r] - A[\ell]$ is maximized.

(b) **Optional extra credit:** Give a linear time algorithm for this problem.