15-451 Algorithms, Fall 2010

Homework # 2 **Due: Tue-Fri, September 21-24, 2010**

Ground rules:

- This is an oral presentation assignment. You should work in groups of three. At some point before **Saturday, September 18 at 11:59pm** your group should sign up for a 1-hour time slot on the signup sheet on the course web page.
- Each person in the group must be able to present every problem. The TA/Professor will select who presents which problem. The other group members may assist the presenter.
- You are not required to hand anything in at your presentation, but you may if you choose.

Problems:

(28 pts) 1. **JSort.** Jeremiah claims to have designed a new comparison based sorting algorithm called *JSort*, which makes at most n comparisons to sort a list L of length n. Jeremiah offers the following caveat: oftentimes JSort gives the wrong answer! However, Jeremiah claims to have proved that

$$\mathbf{Pr}[JSort(L) = Sort(L)] \geq \frac{1}{100},$$

when L is chosen uniformly at random from all permutation on n elements.

(a) (5 pts) Help Jeremiah improve his algorithm. Using *JSort* design a randomized algorithm *RandomizedJSort* with the following guarantee:

$$\forall L, \mathbf{Pr}[RandomizedJSort(L) = Sort(L)] \geq \frac{1}{100}.$$

Your new algorithm should not make any more than n comparisons. You may assume that Jeremiah's claim is correct.

(b) (5 pts) For a particular input L, you may want to know for certain if *RandomizedJSort(L)* gave you the right answer. How many comparisons do you need to make in the worst case? Give an upper and a lower bound.

(c) (5 pts) Suppose you try the following experiment: Run $RandomizedJSort(L)$, check if it is correct, if it not correct then keep trying until it is. Otherwise output the sorted list. In expectation, how many times will you have to run *RandomizedJSort* before you are successful?

(d) (13 pts) Prove that Jeremiah's claim is actually too good to be true.

(28 pts) 2. **Tight Upper/Lower Bound** In each of these questions, you are given $n > 0$ pieces of candy. One of the pieces of candy has a special flavor that you need to give your favorite 15451 TA. All of them look identical, but the special piece of candy has a slightly different weight (difference in weights between one normal piece of candy and the special piece is much less than the weight of one normal piece). Since you cannot

tell the differece in weights by hand, you will need to use a special weighing robot to make comparisons.

All the algorithms you present in these questions should be deterministic. For this question, an algorithm is considered optimal if its worst case is not worse than any other algorithms' worst case (You do not have to consider the average case or the best case). In each problem we will be looking for ***exact*** (not asymptotic) upper and lower bounds.

(a) You can present the robot with two *sets* of candies, A and B, and it will tell you one of three possible answers: $W(A) > W(B)$ or $W(A) = W(B)$ or $W(A) < W(B)$, where $W(X)$ denotes the sum of weights of candies in the set X. Additionally, the manufacturer tells you that the special piece of candy is slightly *heavier* than the others. We want you to find the optimal algorithm for finding the special piece of candy. To do this you will need to do two things: Present an algorithm and analyze the number of comparisons made in the worst case (upper bound: 3pts), Prove that no algorithm will make fewer comparisons in the worst case (lower bound: 4pts).

(b) Being sensitive to the smell of candies, the robot became a little stupid. Instead of three possible answers, it will now only give you one of the two possible answers: $W(A) = W(B)$ or $W(A) \neq W(B)$. You still know that the special candy is *heavier* than the others. Find the optimal algorithm (in terms of the number of comparisons made). For simplicity, assume that n is a power of 2. (3pts for upper bound, 4pts for lower bound).

(c) You attempt to fix the robot, but you are unsuccessful. Now the robot gives you one of the two possible answers: $W(A) > W(B)$ or $W(A) \leq W(B)$. To make matters worse, you receive a notice from the manufacturer telling you that they were mistaken. The special piece of candy might be lighter or heavier than the others. They don't know. (However, they assure you that its weight is still different from the others). Now in addition to identifying the special piece of candy your algorithm must also determine if the special piece of candy is heavier or lighter than regular candy. Find the optimal algorithm. For simplicity, assume that n is a power of 2 and $n > 2$. (4pts for upper bound, 3pts for lower bound).

(d) The manufacturer sends you a letter telling you that they figured it out. The special piece of candy is actually *lighter* than the others. You try to fixed the robot, and at first glance it appears to be working. It now gives the one of three possible answers as described in (a). However, the robot cannot stand the sweet smell of the candies anymore. Whenever it compares two sets of the equal weights, it eats up all the candies that were just compared. You still need to use the robot to find out the special candy. Now your goal is to minimize the number of candies consumed by the robot instead of minimizing number of comparisons. Assuming the n is *odd*, find the optimal algorithm. (3pts for upper bound, 4pts for lower bound).

(28 pts) 3. **The Lincoln Park Problem**

There are n barbies distributed across multiple play houses, p, where $p \geq n$.

At each step, an adversary picks a house and climbs in their window, snatching their barbies up (let k be the number of barbies in that house) and distributes these k bar-

bies to other houses, while leaving his t-shirt and finger prints behind. The adversary redistributes barbies to houses with the caveat that all these barbies must be in different houses after the step. The cost of the step is k^2.

For simplicity, you can assume that the adversary never stops and always continues to choose a house and redistribute barbies.

For example, consider houses h_0, h_1, h_2, h_3, and h_4 with 3 barbies in h_3, 1 in h_1, and 0 in the rest of the houses. If the adversary chooses to go into h_3, he then has the choice to redistribute the 3 barbies in any manner possible as long as those 3 barbies end up in different houses. Thus, he could choose to distribute them to h_0, h_1, h_2 giving us 1 barbie in h_0 and h_2, and 2 barbies in h_1. If he had chosen to distribute the barbies to h_2, h_3 and h_4, we would then arrive at a configuration where h_1, h_2, h_3 and h_4 have 1 barbie each, and h_0 has none. The cost of both of these steps would be 3^3 or 9.

(a) (13 pts) Find and prove an upper bound on the worst case amortized cost per a step (give asymptotic bounds).

(b) (15 pts) Find and prove a lower bound on the worst case amortized cost per a step (give asymptotic bounds).

(16 pts) 4. **The Curveball:** After your team present your solution to each of the above problems we reserve the right to make small changes to the problem, and ask how the answers would change. You will be expected to answer these questions on the fly. We are not going to tell you what these variations are in advance, that would defeat the purpose. Your best strategy is to make sure that everyone on your team *understands* each problem and its solution. If you have a good *understanding* of the solution then these variations should not be too difficult to answer.