Algorithms Homework - Day 1

Instructor: Pat Devlin — `prd41@math.rutgers.edu`

Summer, 2016

(1) Prove the generalized information theory lower bound.

(2) Determine (with proof) how many "yes/no" questions are needed to solve the guessing game if $|S| = n$. (Assume *any* "yes/no" question is allowed.) Try to generalize this result.

(3) The search problem was stated with a list of size $n = 31$. Determine (with proof) exactly how many questions are needed in general as a function of n. [Hint: this involves either proving a lower bound better than the information theory bound or finding a better algorithm (or both).]

(4) [**The group testing problem**] We have 10 bottles of wine, and we know that exactly 1 of them is poisoned. We decide to use our fancy poison testing machine to figure out which one it is, but the machine is very expensive to use. So instead of testing the bottles one at a time, we decide to take samples from several different bottles, mix the samples together, and then test this combined sample. This process only costs us using the machine once, and it will tell us whether or not there is any poison in our combined sample (though of course it would not tell us *which* bottle in our sample contained the poison).

 (a) Determine (with proof) the least number of times we need to run the machine in order to find the poison.

 (b) Now suppose that among the 10 bottles, we know that exactly 9 of them are poisoned. How many times must we run the machine in this situation?

 (c) Try to generalize your results (e.g., to n bottles of wine, or suppose exactly 2 are poisoned, et cetera).

(5) Bob likes dropping eggs out of buildings[1]. Bob is in a building with 100 floors, and he wants to know what's the highest floor he can drop the egg without it breaking [it's of course possible that it breaks from floor 1 or that it doesn't even break from floor 100].

 (a) Suppose he has as many eggs as he could possibly want. How many drops does he need to do (in the worst-case scenario)?

 (b) Now suppose he only has *one* egg. If he drops it, he can go outside and pick it back up. And he can reuse the egg if it didn't break (if it did break, then he can't use it any more). How many drops does he need to do (in the worst-case scenario) so that he can be *guaranteed* to know the highest floor they can be dropped without breaking?

 (c) Repeat part (b), but now assume he has *two* eggs he can use. (So if one breaks, he can't reuse it, but he still has the second one he can use.)

 (d) Try to generalize your results.

(6) You have spheres numbered 1 through 50 that all look and feel the same to you, but your friend Robin has some weird ranking of which sphere she likes the best, which is second best, and so on (all the way down to which she likes least). She wants to play a game where you figure out which of these she likes the best, but the only questions she'll answer are of the form "which of these two spheres do you like better?"

 (a) Try to come up with an algorithm that figures out which she likes the best by asking as few questions as possible. (Proving that your algorithm is best possible is probably difficult, so don't worry about that.)

[1] Don't try this at home.

(b) Try to come up with an algorithm that figures out which she likes the best *and also* which she likes the second best by asking as few total questions as possible. (Proving that your algorithm is best possible is certainly difficult, so definitely don't worry about that.)

(c) Try to generalize your results.

(7) The algorithms we discussed for the search problem were *adaptive*, in that the questions we asked were allowed to depend on the answers to previous questions we received. Imagine instead that you have an index card, and you need to write down all the positions that you would like to be uncovered. You then hand Pat this index card, and he uncovers all the positions you wrote down.

(a) In a completely *non-adaptive* algorithm, after Pat uncovers the positions written on the index card, you would have to determine where the target number is without asking any other questions. In this setting, how many positions would you need to write down on the index card to be certain that you can find the target number.

(b) Now suppose you're allowed to use *two* index cards. So after giving Pat the first card, you can think about it and then use the second index card to write down which positions you would like to be uncovered next. How many positions in total would you have to uncover in this setting? (Proving a lower bound might be hard.)

(c) Try to generalize your results.