CS 5800: Algorithms V. Pavlu, K.Lieberherr

Due: Feb. 17, 2014

Homework Module 6

1 Submission Rules

`http://www.ccs.neu.edu/home/lieber/courses/algorithms/`
`cs5800/sp14/homeworks/submission-rules.pdf`

2 Problems

1. Jars on a ladder problem. Given a ladder of n rungs and k identical glass jars, one has to design an experiment of dropping jars from certain rungs, in order to find the highest rung on the ladder from which a jar doesn't break if dropped.

   Idea: With only one jar (k=1), we can't risk breaking the jar without getting an answer. So we start from the lowest rung on the ladder, and move up. When the jar breaks, the previous rung is the answer; if we are unlucky, we have to do all n rungs, thus n trials. Now lets think of k=log(n): with log(n) or more jars, we have enough jars to do binary search, even if jars are broken at every rung. So in this case we need log(n) trials. Note that we can't do binary search with less than log(n) jars, as we risk breaking all jars before arriving at an answer in the worst case.

   Earlier, we used this equivalent description, called HSR-Intro:

   `http://www.ccs.neu.edu/home/lieber/courses/algorithms/cs5800/sp14/`
   `labs/HSR-problem-CS5800-1.pdf`

   Your task is to calculate $q = MinT(n, k)$= the minimum number of dropping trials any such experiment has to make, to solve the problem even in the worst/unluckiest case (i.e., not running out of jars to drop before arriving at an answer). $MinT$ stands for Minimum number of Trials. $MinT$ corresponds to HSRnk-min(n,k) in HSR-Intro.

   A(5 points). Explain the optimal solution structure and write a recursion for $MinT(n, k)$.

   B(5 points). Write the alternative/dual recursion for $MaxR(q, k)$ = the Highest Ladder Size n doable with k jars and maximum q trials. Explain how $MinT(n, k)$ can be computed from the table $MaxR(q, k)$. $MaxR$ stands for the Maximum number of Rungs. Using the terminology of HSR-Intro, we would call $MaxR$ to be HSRqk-max(q,k).

   C(10 points). For one of these two recursions (not both, take your pick) write the bottom-up non-recursive computation pseudocode. *Hint: the recursion $MinT(n, k)$ is a bit more difficult and takes more computation steps, but once the table is computed,* *the rest is easier on points E-F below. The recursion in $MaxR(q, k)$ is perhaps easier, but trickier afterwards: make sure you compute all cells necessary to get $MinT(n, k)$— see point B.*

D(10 points). Redo the computation this time top-down recursive, using memoization.

E(10 points). Trace the solution for $MinT(n, k)$. While computing bottom-up, use an auxiliary structure that can be used to determine the optimal sequence of drops for a given input n, k. The procedure TRACE(n, k) should output the ladder rungs to drop jars, considering the dynamic outcomes of previous drops. *Hint: its recursive. Somewhere in the procedure there should be an **if statement** like "if the trial at rung x breaks the jar... else ..."*

F(10 points). In A you use a minimum over a range 0..n-1. Can you efficiently predict the value to be chosen in this range without having to do a linear search for it? Compare with section 16.1 in the text book where a dynamic programming solution is simplified using a greedy choice.

G(20 points). Implement in your favorite programming language a perfect avatar for the debate associated with claim MinHSR() (see HSR-Intro). To test your avatar run a debate in your teams of three. Before you run the debates, set reasonable boundary conditions, say that your perfect avatar needs to handle only values of n less than one million. Feed the objects obtained from your partner manually to your avatar. Test them for correctness to make sure there is no cheating. Use JSON as described in HSR-Intro to exchange objects during the debate.

Output the entire decision tree produced by your perfect avatar using JSON to express the tree, for the following test cases : (n=9,k=2); (n=11, k=3); (n=10XYZ,k=9) where XYZ are the last three digits of your NU-ID. Turn in your avatar that produces the optimum decision tree for given n and k using JSON. Turn in a zip folder that contains: (1) all files required by your avatar (2) instructions how to run your avatar (3) the three decision trees in files t-9-2.json, t-11-3.json and t-10000-9.json (4) the answers to all other questions of this homework.

H(extra credit, 20 points). Solve a variant of this problem for q = MinT(n,k) that optimizes the average case instead of the worst case: now we are not concerned with the worst case q, but with the average q. Will make the assumption that all cases are equally likely (the probability of the answer being a particular rung is the same for all rungs). You will have to redo points A,C,E specifically for this variant.