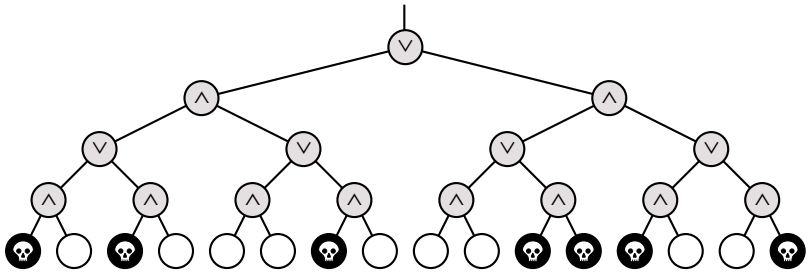
CS 473: Undergraduate Algorithms, Spring 2009
Homework 9

Due Tuesday, April 28, 2009 at 11:59:59pm.

- Groups of up to three students may submit a single, common solution for this and all future homeworks. Please clearly write every group member's name and NetID on every page of your submission.

1. Death knocks on your door one cold blustery morning and challenges you to a game. Death knows that you are an algorithms student, so instead of the traditional game of chess, Death presents you with a complete binary tree with 4^n leaves, each colored either black or white. There is a token at the root of the tree. To play the game, you and Death will take turns moving the token from its current node to one of its children. The game will end after $2n$ moves, when the token lands on a leaf. If the final leaf is black, you die; if it is white, you will live forever. You move first, so Death gets the last turn.



You can decide whether it is worth playing or not as follows. Imagine that the nodes at even levels (where it is your turn) are OR gates, the nodes at odd levels (where it is Death's turn) are AND gates. Each gate gets its input from its children and passes its output to its parent. White and black leaves stand represent TRUE and FALSE inputs, respectively. If the output at the top of the tree is TRUE, then you can win and live forever! If the output at the top of the tree is FALSE, you should challenge Death to a game of Twister instead.

   (a) Describe and analyze a deterministic algorithm to determine whether or not you can win. *[Hint: This is easy.]*

   (b) Prove that *every* deterministic algorithm must examine *every* leaf of the tree in the worst case. Since there are 4^n leaves, this implies that any deterministic algorithm must take $\Omega(4^n)$ time in the worst case. Use an adversary argument; in other words, assume that Death cheats.

   (c) ***[Extra credit]*** Describe a *randomized* algorithm that runs in $O(3^n)$ expected time.

2. We say that an array $A[1..n]$ is *k-sorted* if it can be divided into k blocks, each of size n/k, such that the elements in each block are larger than the elements in earlier blocks, and smaller than elements in later blocks. The elements within each block need not be sorted.

   For example, the following array is 4-sorted:

1	2	4	3	7	6	8	5	10	11	9	12	15	13	16	14

(a) Describe an algorithm that k-sorts an arbitrary array in time $O(n \log k)$.

(b) Prove that any comparison-based k-sorting algorithm requires $\Omega(n \log k)$ comparisons in the worst case.

(c) Describe an algorithm that completely sorts an already k-sorted array in time $O(n \log(n/k))$.

(d) Prove that any comparison-based algorithm to completely sort a k-sorted array requires $\Omega(n \log(n/k))$ comparisons in the worst case.

In all cases, you can assume that n/k is an integer.

3. UIUC has just finished constructing the new Reingold Building, the tallest dormitory on campus. In order to determine how much insurance to buy, the university administration needs to determine the highest safe floor in the building. A floor is consdered *safe* if ~~a drunk student~~ ***an egg*** can fall from a window on that floor and land without breaking; if the egg breaks, the floor is considered *unsafe*. Any floor that is higher than an unsafe floor is also considered unsafe. The only way to determine whether a floor is safe is to drop an egg from a window on that floor.

You would like to find the lowest unsafe floor L by performing as few tests as possible; unfortunately, you have only a very limited supply of eggs.

(a) Prove that if you have only one egg, you can find the lowest unsafe floor with L tests. *[Hint: Yes, this is trivial.]*

(b) Prove that if you have only one egg, you must perform at least L tests in the worst case. In other words, prove that your algorithm from part (a) is optimal. *[Hint: Use an adversary argument.]*

(c) Describe an algorithm to find the lowest unsafe floor using *two* eggs and only $O(\sqrt{L})$ tests. *[Hint: Ideally, each egg should be dropped the same number of times. How many floors can you test with n drops?]*

(d) Prove that if you start with two eggs, you must perform at least $\Omega(\sqrt{L})$ tests in the worst case. In other words, prove that your algorithm from part (c) is optimal.

(e) **[Extra credit!]** Describe an algorithm to find the lowest unsafe floor using k eggs, using as few tests as possible, and prove your algorithm is optimal for all values of k.