CS 473: Fundamental Algorithms, Fall 2011

Discussion 8

October 11, 2011

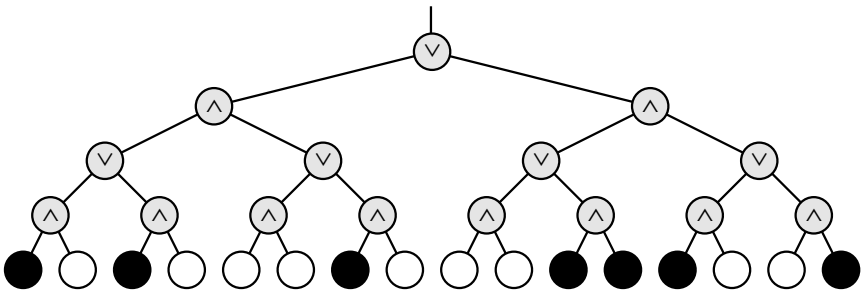
8.1 Game tree evaluation.

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's white, you will live forever. You move first, so Death gets the last turn.

You can decide whether it's worth playing or not as follows. Imagine that the nodes at even levels (where it's your turn) are OR gates, the nodes at odd levels (where it's Death's turn) are *and* gates. Each gate gets its input from its children and passes its output to its parent. White and black stand for TRUE and FALSE. 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) (2 pts) Describe and analyze a deterministic algorithm to determine whether or not you can win. *[Hint: This is easy!]*

(b) (8 pts) Unfortunately, Death won't let you even look at every node in the tree. Describe a *randomized* algorithm that determines whether you can win in $\Theta(3^n)$ expected time. *[Hint: Consider the case $n = 1$.]*



8.2 Selection on Sorted arrays.

Suppose you are given two sorted arrays $A[1 \ldots n]$ and $B[1 \ldots m]$ and an integer k. Describe an algorithm to find the kth smallest element in the union of A and B in $\Theta(\log(m + n))$ time. For example, given the input

$A[1 \ldots 8] = [0, 1, 6, 9, 12, 13, 18, 20] \quad B[1 \ldots 5] = [2, 5, 8, 17, 19] \quad k = 6$

Your algorithm should return 8. Assume the array contains no duplicates.
[Hint: What can we learn by comparing an element of A to an element of B?]

8.3 Nuts and Bolts.

Suppose we are given n nuts and n bolts of different sizes. Each nut matches exactly one bolt and vice versa. The nuts and bolts are all almost exactly the same size, so we can't tell whether one bolt is smaller than the other, or if one nut is bigger than the other. If we try to match a nut with a bolt, it would be either too big or too small for the bolt or it would be just right for the bolt. Find the correct matchings in worst case expected $O(n \log n)$ time.

[Hint:If a nut and bolt don't match how can you partition the nuts using the bolt and the bolts using the nut?]