Design and Analysis of Algorithms: Homework #3

Due in class on March 1, 2018

Professor Kasturi Varadarajan

Alic Szecsei

Problem 1

Consider the following randomized algorithm for generating biased random bits. The subroutine **FairCoin** returns either 0 or 1 with equal probability; the random bits returned by **FairCoin** are mutually independent.

```
function OneInThree
    if FairCoin = 0 then
        return 0
    else
        return 1−OneInThree
    end if
end function
```

(a) Prove that **OneInThree** returns 1 with probability $\frac{1}{3}$.

(b) What is the *exact* expected number of times that this algorithm calls **FairCoin**?

(c) Now suppose you are *given* a subroutine **OneInThree** that generates a random bit that is equal to 1 with probability $\frac{1}{3}$. Describe a **FairCoin** algorithm that returns either 0 or 1 with equal probability, using **OneInThree** as your only source of randomness.

(d) What is the *exact* expected number of times that your **FairCoin** algorithm calls **OneInThree**?

Solution

Problem 2

Consider the following algorithm for finding the smallest element in an unsorted array:

```
function OneInThree(A[1..n])
    min ← ∞
    for i ← 1 to n in random order do
        if A[i] < min then
            min ← A[i]                                        ▷ (⋆)
        end if
    end for
    return min
end function
```

(a) In the worst case, how many times does **RandomMin** execute line (⋆)?

(b) What is the probability that line (⋆) is executed during the nth iteration of the for loop?

(c) What is the *exact* expected number of executions of line (⋆)?

Solution

Problem 3

Suppose we have a circular linked list of numbers, implemented as a pair of arrays, one storing the actual numbers and the other storing successor pointers. Specifically, let $X[1..n]$ be an array of n distinct real numbers, and let $N[1..n]$ be an array of indices with the following property: If $X[i]$ is the largest element of X, then $X[N[i]]$ is the smallest element of X; otherwise, $X[N[i]]$ is the smallest among the set of elements in X larger than $X[i]$. For example:

i	1	2	3	4	5	6	7	8	9
$X[i]$	83	54	16	31	45	99	78	62	27
$N[i]$	6	8	9	5	2	3	1	7	4

Describe and analyze a randomized algorithm that determines whether a given number x appears in the array X in $O(\sqrt{n})$ expected time. ***Your algorithm may not modify the arrays X and N.***

Solution

Problem 4

A *majority tree* is a complete binary tree with depth n, where every leaf is labeled either 0 or 1. The *value* of a leaf is its label; the *value* of any internal node is the majority of the values of its three children. Consider the problem of computing the value of the root of a majority tree, given the sequence of 3^n leaf labels as input. For example, if $n = 2$ and the leaves are labeled 1, 0, 0, 0, 1, 0, 1, 1, 1, the root has value 0.

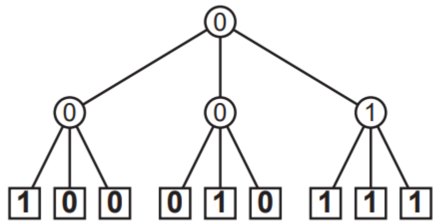

Figure 1: A majority tree with depth $n = 2$.

(a) Prove that *any* deterministic algorithm that computes the value of the root of a majority tree *must* examine every leaf. *[Hint: Consider the special case $n = 1$. Recurse.]*

(b) Describe and analyze a randomized algorithm that computes the value of the root in worst-case expected time $O(c^n)$ for some constant $c < 3$. *[Hint: Consider the special case $n = 1$. Recurse.*

Solution

Problem 5

Suppose you are given a graph G with weighted edges, and your goal is to find a cut whose total weight (not just number of edges) is smallest.

(a) Describe an algorithm to select a random edge of G, where the probability of choosing edge e is proportional to the weight of e.

(b) Prove that if you use the algorithm from part (a), instead of choosing edges uniformly at random, the probability that **GuessMinCut** returns a minimum-weight is still $\Omega(1/n^2)$.

(c) What is the running time of your modified **GuessMinCut** algorithm?

Solution