

Practice Problems for Homework 2

1. Consider the problem of searching for x in an array A of n elements:

```
boolean search(n,A,x){    \\searches for x in the unsorted array A[0..n-1]
    int i=0;
    while (i <= n-1 && A[i] != x)    \\searches for x
        i++;
    if (A[i] == x) return true;
    else return false;
```

You are to give an exact answer for number of times that $A[i] != x$ is executed.

Along with the arithmetic sum $\sum_{i=1}^k i = k(k+1)/2$, another useful summation (obtained by integrating both sides of the geometric series $[\sum_{i=0}^k x^i] = (1-x^{k+1})/(1-x)$ and then setting $x = 1/2$ is

$$\sum_{i=0}^k i(1/2)^i = \sum_{i=1}^k i(1/2)^i = 2 - \frac{1}{2^{k-1}} - \frac{k}{2^k}$$

- (a) Assume that with probability $1/2$, x is in $A[0]$, with probability $1/4$, x is in $A[1]$ and with probability $1/4$, x is not in A .
- (b) Assume that with probability $1/2$, x is not in A and with probability $1/(2n)$, x is position i of the list for $i = 0, 1, \dots, n-1$.
- (c) Assume that for $i = 0, 1, \dots, n-1$, that the probability that x is in position i of the array is $1/(2^{i+1})$ and with probability $1/(2^n)$ that x is not in the array.
2. Here we consider a situation in which calls will be repeatedly made to search for an item in a unsorted list L . Further, you know that items searched for recently are more likely to be searched for again. Thus, to reduce the search time, whenever an item is found it is moved to the front of L . Here's pseudo-code for the search procedure where `ptr.value` gives the value of the item pointed to by `ptr` and `ptr.next` is a reference to the list item after that referenced by `ptr`.

```
boolean search(L,x){    \\searches for x in the unsorted list L
    initialize ptr to be a reference to the first item in L
    while (ptr != nil && ptr.value != x)    \\searches for x
        ptr = ptr.next;
    if (ptr != nil) {    \\if found
        move item referenced by ptr to the front of L
        return true;
    }
    else
        return false;
}
```

You are given that for $1 \leq i \leq n - 1$, that the probability that x is in position i of the list is $\frac{1}{2^i}$ and the probability that x is in position n is $\frac{1}{2^{n-1}}$. (The first item in the list is item 1, the second item in the list is item 2, and so on with the last item in the list being item n .)

What is the expected number of times that the comparison (`ptr.value != x`) is executed? You are to give an EXACT answer.

Note: Be sure to show your work starting from the definition of expected value.

3. (15 pts) You are to implement a caller-id system (for all US phone numbers) with the three below operations supported.

Your implementation is to be designed for the following conditions. There are 10^8 phone numbers that will be inserted with 10^5 different area code/exchange (first 6 digits of the number) combinations occurring. Further, suppose you only have space in main memory (a.k.a. RAM) to hold 8,000,000 bytes (roughly what is in an inexpensive PDA).

I've added these constraints on the memory to make this a more interesting practice problem. The key facts you need to know to think about this are:

- An integer takes 4 bytes (32 bits) – the phone number can be stored as an integer
- A long takes 8 bytes – when you need to store something on disk you can represent the location of a page using a long integer.
- All information stored on the disk (i.e not in main memory) is grouped into pages which are brought into RAM as a whole. To keep it simple, assume you can hold 1,000,000 bytes on a page and in constant time a page can be read from disk (or written to disk) when given its location (a long integer).

You need to support the following methods. All must run in expected constant time.

- Insert a new area code/exchange combination into the system. (This will be used infrequently).
- Insert into the system a new item that consists of a phone number and name. (You can assume that the area code/exchange from the number corresponds to one of the area code/exchanges already in the system.)
- Given a phone number, return the name.
- Given a phone number, remove the corresponding item from the system.