CSE 103: Probability and statistics **Fall 2010**

Homework 7

1. A *dominating set* in an undirected graph $G = (V, E)$ is a subset of nodes $U \subset V$ such that every node is either in U or adjacent to (that is, has an edge to) U. The problem of finding the *smallest* dominating set is of practical importance, but is NP-complete; no efficient algorithm is known for it. We'll use randomization to construct a pretty good solution, in the case where each node of G has degree d.

   $U = \emptyset$
   For each $v \in V$:
   Add v to U with probability p

   (a) Pick any node $v \in V$. We will say that v is *covered* if either $v \in U$ or if v is adjacent to U. What is the probability that v is not covered, as a function of p and d?

   (b) Show that for $p = (\ln 2n)/(d+1)$, where $n = |V|$, the expected number of uncovered nodes is less than 1/2. *Hint:* use the inequality $1 - x \leq e^{-x}$.

   (c) For this setting of p, what is the expected size of U?

   (d) **(Challenging)** Show that for this setting of p, the probability that U is a valid dominating set is at least 1/2. That is, the algorithm fails with probability less than 1/2.

   (e) Assuming part (d) is true, how can the probability of failure be reduced to δ?

2. Barbara Smith is interviewing candidates to be her secretary, one at a time. After each interview she is able to determine the true competence level of the candidate (which can be thought of as some positive real number). However, she needs to make a spot decision whether or not to hire a candidate, before interviewing the remaining ones. If the candidates appear in random order, what is a good hiring strategy for Barbara?

   Suppose there are n candidates. Barbara decides to interview the first r candidates, noting down their scores but not hiring any of them. Let s be the largest score she records during this time. Then, she starts interviewing the remaining $n - r$ candidates, and hires the first one who scores more than s (if none of them do, then she simply picks the last candidate).

   (a) Show that the probability that Barbara ends up with the best secretary is at least $r(n - r)/n^2$. *Hint:* Consider the event that the second-best secretary is in the first group of r people, while the best secretary is the second group.

   (b) What is a good setting for r, and what is the probability of success in this case?

3. Suppose you have access to two different algorithms for testing whether a number is prime. The first is a randomized algorithm $\mathcal{A}$ that runs in time $T(n)$ and has the following behavior:

   - If x is prime, $\mathcal{A}(x)$ always says "prime"
   - If x is not prime, $\mathcal{A}(x)$ says "not prime" with probability 1/2; otherwise it says "prime"

   Algorithm $\mathcal{B}$ is deterministic, runs in time $100T(n)$, and always returns the correct answer.

   You wish to create an algorithm that is always correct and is as efficient as possible.

(a) Specify your algorithm.

(b) What is the expected running time of your algorithm when the input is prime?

(c) When the input is not prime?

4. Suppose you have a linked list of n elements *in sorted order*. Here's the obvious way to look up an element x, given a pointer to the first node in the list:

   ```
   Function lookup(L, x)
   if L is NULL: return NULL
   if value(L) = x:  return L
   if value(L) > x:  return NULL
   return lookup(next(L), x)
   ```

   Notation: given the pointer P to a node, value(P) and next(P) are the value in the node and the pointer to the next node.

   (a) Suppose an element of L is chosen at random. What is (roughly) the expected time (in terms of n) to look up this element?

   (b) One way to get faster lookups is to add a second pointer jump(P) that points to the node k steps down the list from P (that is, following a "jump" pointer is like following k "next" pointers). The procedure above is then altered by adding the following line just before the final return statement:

   ```
   if value(jump(L)) ≤ x:  return lookup(jump(L), x)
   ```

   With this addition, the data structure is called a *skip list*. Now suppose an element of L is chosen at random. What is, roughly, the expected time to look it up, as a function of n and k?

   (c) What is the best choice of k?

5. For cryptographic applications, it is necessary to generate random n-bit prime numbers. One way to do this is to repeatedly generate random n-bit numbers until you find one that is prime. Suppose that we check primality using an algorithm $\mathcal{A}$ with the following behavior on n-bit numbers:

   - If x is prime, $\mathcal{A}(x)$ always says "prime"
   - If x is not prime, $\mathcal{A}(x)$ says "prime" with probability at most $1/(10^6 n)$; otherwise "not prime"

   The overall scheme for generating a random n-bit prime then looks like this:

   - Repeat:
     - Generate a random n-bit number x.
     - If $\mathcal{A}(x)$ = "prime": return x and halt.

   A useful fact from number theory: a random n-bit number has approximately a c/n probability of being prime, for some small constant c. To keep things simple, assume this probability is exactly $1/n$.

   (a) Give an upper bound on the probability that the algorithm ends up returning a number that is not prime. *Hint:* use Bayes' rule.

   (b) What is the expected running time of the algorithm, assuming $\mathcal{A}$ runs in time $O(n^3)$?

6. You are dealt ten cards, one at a time, from the top of a randomly shuffled deck.

   (a) What is the probability that the tenth card you get is an ace?

(b) What is the probability that the tenth card is an ace, given that the third card is an ace?

(c) What is the probability that the third card is an ace, given that the tenth card is an ace?

7. Pick a random permutation of $(1, 2, \ldots, n)$. Let X_i be the number that ends up in the ith position. For instance, if the permutation is $(3, 2, 4, 1)$ then $X_1 = 3$, $X_2 = 2$, $X_3 = 4$, and $X_4 = 1$.

   (a) What is the expected number of positions at which $X_i \neq i$?

   (b) What is the expected number of positions at which $X_i = i + 1$?

   (c) What is the expected number of positions at which $X_i \geq i$?

   (d) What is the expected number of positions at which $X_i > \max(X_1, \ldots, X_{i-1})$?

8. A set of n people are lined up along a wall in a random order. Among them are Alice, Bob, and Chet.

   (a) What is the probability that Alice appears somewhere to the left of Bob, and Bob appears somewhere to the left of Chet?

   (b) What is the expected number of people between Alice and Bob?

9. In class, we saw how to generate fair coin flips given only a coin of unknown bias. But suppose we have a coin whose bias, p, we *know*. Can we then do better, in terms of using fewer coin flips in expectation? Here's a potential scheme which assumes (without loss of generality) that $p < 1/2$. In fact, this scheme can generate a coin flip of any desired bias q, given the coin of bias p. To get a fair coin flip, call it with $q = 1/2$.

```
Function simulate(q)
If q < 1/2:
   Return switch(simulate(1 − q)) // 'switch' converts H to T and T to H
Flip the coin of bias p
If it turns up heads:
   Return H
else:
   Return simulate((q − p)/(1 − p))
```

   (a) Can you see why this algorithm will output H with probability exactly q? To show this, assume the recursive calls work correctly, and show that the top-level algorithm will do the right thing. Separately consider the two cases where $q < 1/2$ and $q \geq 1/2$.

   (b) What is the expected number of times the biased coin is flipped?

10. **(Challenging)** *From Las Vegas to Monte Carlo.* Suppose you have an algorithm $\mathcal{A}$ for your problem that always returns the correct answer, but takes different amounts of time each time it runs. Its *expected* time on an input of size n is $T(n)$. What you would prefer, however, is an algorithm that *always* finishes in time $O(T(n))$, but may have up to a 5% probability of returning the wrong answer. Show how to construct such an algorithm from $\mathcal{A}$. *Hint:* Start by showing that for any $c > 1$, the probability that $\mathcal{A}$ takes longer than $cT(n)$ is at most $1/c$. Then use a timer while running $\mathcal{A}$!

Answer key 1 (a) $(1-p)^{d+1}$ (c) $(n \ln 2n)/(d+1)$ (e) Repeat the algorithm $\log 1/\delta$ times, and return the smallest valid dominating set found (if any). 2. Choosing $r = n/2$ gives at least a 1/4 probability of success. 3 (a) If $\mathcal{A}(x) =$ "not prime" return "not prime" else return $\mathcal{B}(x)$ (b) $101T(n)$ (c) $51T(n)$. 4 (a) roughly $n/2$ (b) roughly $n/2k + k/2$ (c) $\sqrt{n}$. 5 (a) 10^{-6} (b) $O(n^4)$. 6 (a) 1/13 (b) 1/17 (c) 1/17. 7 (a) $n-1$ (b) $1 - 1/n$ (c) $(n+1)/2$ (d) approximately $\ln n$. 8 (a) 1/6 (b) $(n-2)/3$. 9 (b) $1/p$. 10. Run $\mathcal{A}$ for $20T(n)$ time steps; if it is still going, halt and return some generic answer.