CS648 : Randomized Algorithms
Semester I, 2011-12, CSE, IIT Kanpur

Assignment - 1 (due on 26th August, 9 AM)

August 6, 2011

Note: Give complete details of the analysis of your solution. Be very rigorous in providing any mathematical detail in support of your arguments. Also mention any Lemma/Theorem you use.

1. **Randomized quick select**

   Let S be a set of n real numbers. Consider the randomized algorithm *Rand-QSelect*(k, S) as described below that finds the k^{th} smallest element from the S.

   Select a pivot element x uniformly randomly from set S. Find its rank in the set S (by comparing x with every other element of set S). Let r be the rank of x. If $r = k$, we report x as the output. Otherwise we proceed recursively as follows:

   If $r > k$ then *Rand-QSelect*$(k, S_{<x})$

   else *Rand-QSelect*$(k - r, S_{>}x)$

   Where $S_{<}x$ and $S_{>}x$ are the sets consisting of all those elements that are respectively smaller and greater than the element x. Observe that the running time of the above algorithm is dominated by the number of comparisons performed. Therefore, in order to get a bound on the expected running time of the algorithm, our aim is essentially to find out the expected number of comparisons performed in *Rand-QSelect*(k, S). Prove the following statements.

   (a) The expected number of comparisons is at most $4n$ (3 bonus points to prove an upper bound of $3.5n$).

   (b) There are elements in set S which will be compared $\Theta(\log n)$ times on expectation by the algorithm. Can you characterize these elements (this part of the problem is not to be submitted).

   Note that proof by induction will not be accepted as a solution to this problem. Proofs by induction are based on making a right guess and not based on insight into the problem/algorithm. You should take inspiration from the analysis we did for randomized quick sort and proceed along similar lines.

2. **Careful application of partition theorem**

   Recall the partition theorem, which states that if events $\mathcal{E}_1, ..., \mathcal{E}_\ell$ form a partition of a sample space Ω, and A is any event, then

   $$\mathbf{P}[A] = \sum_{j=1}^{\ell} \mathbf{P}[A|\mathcal{E}_j] \ \cdot \ \mathbf{P}[\mathcal{E}_j]$$

   This theorem can sometimes be used very effectively to calculate probability of event A when any direct method of calculating $\mathbf{P}[A]$ appears difficult. But applying this theorem effectively requires some creative skills and a better insight into the problem. In general, it works when the partition formed is such that calculating $\mathbf{P}[A|\mathcal{E}_j]$ is very easy for each $\mathcal{E}_j$ (in fact usually it turns out to be independent of j). We had seen one application of partition theorem in the red-blue balls problem. We used it to calculate the expected number of red balls preceding all the blue balls when the balls are taken out uniformly randomly from a bag containing r red balls and b blue balls. The following problem will provide you an opportunity to apply partition theorem.

Professor Dixon needs to hire a research assistant. She has arranged interviews with n applicants and would like to base her decision solely on their qualifications. Unfortunately, university regulations require that after each interview she immediately rejects or offers the position to the applicant. Professor Dixon decides to adopt the strategy of selecting a positive integer $k < n$, interviewing and then rejecting first k applicants, and hiring the first applicant thereafter who is better qualified than all preceding applicants. If the best qualified applicant is among the first k interviewed, she will hire the nth applicant.

(a) Assuming the applicants appear in a uniformly random order (all permutations are equally likely), what is the probability in terms of k and n that Dixon will select the best qualified applicant ? This is how you should proceed. Let $\mathcal{E}$ be the event that Dixon selects the best qualified applicant. Let p_i be the probability that Professor Dixon selects the best candidate given that the best qualified applicant is at ith place. Give arguments to show that

$$\mathbf{P}[\mathcal{E}] = \frac{1}{n}\sum_{i=1}^{n} p_i$$

So all you need to solve the problem is to calculate p_i. The sample space you need to focus for calculating p_i consists of all those elementary events (permutations) where the best qualified person occupies ith position. To come up with a neat and calculation free way to calculate p_i, it makes sense to try out the partition theorem. But which partition of the sample space should we use ? Try out the partitioning defined by

- the sets of persons occupying the first k positions.
- the positions occupied by the second best qualified applicant.

None of them work! Take a break and then try the partitioning defined by the sets of persons occupying first $i-1$ positions. Show with careful arguments that this will work and thus provide a a calculation free way of calculating p_i. Now you should try to investigate why it worked whereas none of the above two partitioning worked. Find out the final expression for $\mathbf{P}[\mathcal{E}]$

(b) What asymptotic value of k (as a function of n) would maximize the chances of selecting the best qualified person ?

3. **Using Randomization to Rectify Errors**

We have a function $F : \{0, \ldots, n-1\} \to \{0, \ldots, m-1\}$. We know that, for $0 \leq x, y \leq n-1$, $F((x+y) \mod n) = (F(x) + F(y)) \mod m$. F can be evaluated only by using a lookup table that stores values of F. Unfortunately, an adversary has changed values of $1/5^{\text{th}}$ of the table entries when we were not looking.

Describe a simple randomized algorithm that, given an input z, outputs a value that equals $F(z)$ with probability at least $1/2$. Your algorithm should work for every value of z, regardless of what values the adversary changed. It should use as few lookups and as little computation as possible.

Suppose you are allowed to repeat your initial algorithm three times. What should you do in this case, and what is the probability that your enhanced algorithm returns the correct answer?

4. **Set Balancing**

Given an $n \times n$ matrix A all of whose entries are 0 or 1, our aim is to compute a column vector $b \in \{-1, +1\}^n$ minimizing $\|Ab\|_\infty$. Here is a surprisingly simple algorithm: Construct the vector b as follows: each entry of b is independently and equiprobably chosen from $\{-1, +1\}$. Show that with very high probability, $\|Ab\|_\infty$ is going to be $O(\sqrt{n\ ln\ n})$.

(**Note:** $\|Ab\|_\infty$ evaluates to the element of the vector Ab with largest absolute value.)