CS648 : Randomized Algorithms
Semester I, 2018-19, CSE, IIT Kanpur

Assignment 1

Deadline : 6:00 PM, Thursday, 23rd August 2018

Important Guidelines:

- It is only through the assignments that one learns the most in any course on algorithms. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. **If you are cheating the instructor, you are cheating yourself first**. The onus of learning from a course lies first on you and then on the quality of teaching of the instructor. So act wisely while working on this assignment.
- Handwritten submissions will not be accepted. You must type your solution using any word processor (For example : Latex, Microsoft Word, Google Doc). You will have to upload the soft copy on moodle before the deadline. You will also have to submit its printed copy before the deadline.
- The answer of each questions must be formal, complete, and to the point. Do not waste your time writing intuition or idea. There will be penalty if you provide any such unnecessary details.
- Each question carries 50 marks. Total marks for this assignment are 150.

1 Randomized quick select

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

1. The expected number of comparisons is at most $3.5n$. (You will loose 20% marks in this question if you are able to get a bound greater than $3.5n$).
2. There are elements in set S which will be compared expected $\Theta(\log n)$ times during the algorithm. Can you characterize these elements (this part of the problem is not to be submitted) ?

2 Making an intelligent guess

We have a function $F : \{O, ..., n-1\} \rightarrow \{O, ..., m-1\}$. We know that, for $0 \leq x, y \leq n-1$,

$$F((x + y) \bmod n) = (F(x) + F(y)) \bmod m$$

The only way we have for evaluating F is to use a lookup table that stores the values of F. Unfortunately, an Evil Adversary has changed the value of 1/5 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. Your algorithm should use as few lookups and as little computation as possible.

Suppose I allow you to repeat your initial algorithm k times. What should you do in this case, and what is the probability that your *enhanced* algorithm returns the correct answer?

3 Reachability in a non-static DAG

(*marks = 25,30,5*)

Consider a directed graph $G = (V, E)$ on n vertices. T is a Boolean matrix storing all-pairs reachability information such that $T[i, j] = 1$ if there is a path from i to j, and 0 otherwise. We can compute T in $O(mn)$ time by carrying out DFS/BFS traversal from each vertex. Let us now discuss the problem that we need to solve.

There is an online sequence of edge insertions and deletions. Each such update may change the reachability information and hence the matrix T. We have to maintain T after each update. A trivial way to accomplish this task is to recompute T from scratch after every edge insertion or deletion. But it seems quite costly. So the aim is to maintain some clever data structure so that the time taken to update T for any edge insertion or deletion is much less than $O(mn)$. Ponder over this problem for a few minutes to get convinced about its non-triviality before moving ahead ...

Recall that a sequence of vertices $\langle (u =)x_1, x_2, \ldots, x_k (= v)\rangle$ is said to be a path from a vertex u to a vertex v if $(x_i, x_{i+1}) \in E$ for each $1 \leq i < k$ and no vertex is repeated in this sequence. Let P_{uv} denote any path from u to v in G at any time. It can be seen that if we remove any prefix (or suffix) of this path, we still get a path. However, converse is true **only** for a directed acyclic graph. Try to prove the following lemma.

Lemma 3.1 *Let $G = V, E)$ be a directed acyclic graph. If P_{ux} and $P_{y,v}$ are two paths and (x, y) is an edge, then is $P_{ux} :: (x, y) :: P_{y,v}$ also a path.*

Henceforth we shall assume that G is acyclic only. Let N_{uv} denote the number of paths from u to v. Now try to explore the answers to the following simple questions.

Question 1: Using Lemma 3.1, how can we express the number of paths from u to v that pass through edge (x, y) in terms of $N_{u,x}$ and N_{yv} ?

Question 2: How does N_{uv} get changed when an edge (x, y) gets deleted from the graph ?

Question 3: How does N_{uv} get changes when an edge (x, y) gets inserted into the graph ?

Ponder over the above questions and their answers for a few minutes and try to see how they can be used to solve the problem of maintaining T for a directed acyclic graph.

Most likely, you will get inspired to maintain T by maintaining matrix N such that $N[u, v]$ stores N_{uv} for all $u, v \in V$. Basically $T[u, v]$ will be 1 if and only if $N[u, v] \neq 0$. Design an algorithm that performs only $O(n^2)$ arithmetic operations to update N after any edge insertion or deletion. Now spot **a serious problem** in this algorithm. Try to see how you can overcome this problem by using prime numbers and design a Randomized Monte Carlo algorithm. As part of the assignment, you will have to submit only the following.

1. Design a randomized Monte Carlo algorithm that will take $O(n^2)$ time on word RAM model per update. Write pseudocode for handling edge insertion and edge deletions.

2. Focus on any pair (u, v). Analyse the probability that $T[u, v]$ will be incorrect at any stage. How will you reduce this error probability to less than n^{-4}.

3. Use some probability tool to ensure that $T[]$ will be completely correct at any stage with probability at least $1 - n^{-2}$.

4 OPTIONAL PROBLEM

This problem highlights the power of partition theorem. This problem will not be asked in any exam. However, you are strongly encouraged to solve it on your own.

Google USA is going to visit IITK to hire the best qualified (good algorithmic and programming skills) student in his/her final year. There are n applicants and n is obviously really huge since Google USA is offering a huge package. They will select a person based totally on his/her qualifications. Assume that there is a total order among all n applicants as far as their qualifications are concerned. Since n is huge, it is not possible to interview every applicant. Furthermore, the placement office requires that each applicant should be informed about his/her selection or rejection immediately after the interview. Therefore, the following strategy is followed by Google. They fix a number $k < n$. They interview and reject first k applicants. After that they continue taking interviews and stop as soon as they find an applicant better than the first k applicants. If they don't find any applicant better than the first k applicants, they return without hiring any one.

1. 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 Google USA will be successful in selecting the best qualified applicant ?

2. For what value of k, is the probability of selecting the best qualified applicant maximum ?