CSOR W4231.002 – Spring, 2020

Homework 2

Out: Monday, February 10, 2020

Due: 11:59pm, Monday, February 24, 2020

You should always describe your algorithm clearly in English. You should always give pseudocode for all algorithms that you design.

For all deterministic algorithms you design, you must prove correctness and give the best upper bound that you can for the running time. You are also encouraged to analyze the space required by your algorithm.

You must submit your assignment as a pdf file. Other file formats will not be graded, and will automatically receive a score of 0.

I recommend you type your solutions using LaTeX. For every theoretical assignment, you will earn 5 extra credit points if you type your solutions. If you do not type your solutions, make sure that your hand-writing is very neat, your scan is high-quality and your name and UNI are clearly written on your homework.

You should write up the solutions **entirely on your own**. Collaboration is limited to discussion of ideas only (see also the course syllabus). Similarity between your solutions and solutions of your classmates, or solutions posted online will result in receiving a 0 in this assignment and possibly further disciplinary actions.

1. (15 points) Let S and T be two disjoint subsets of size $n > 1$ of a finite universe U. Suppose we select a random subset $R \subseteq U$ by independently including each element of U with probability p. We say that the sample is *good* if it contains an element of T but no element of S.

   Show that when $p = 1/n$, the probability our sample is good is larger than some positive constant (independent of n). You may use the fact that for all $x, n \in \mathbf{R}$ such that $n \geq 1$ and $|x| \leq n$,

$$e^x \left(1 - \frac{x^2}{n}\right) \leq \left(1 + \frac{x}{n}\right)^n \leq e^x. \tag{1}$$

2. (25 points) Suppose that a sequence of items passes by one at a time. We want to maintain a sample of one item with the property that it is uniformly distributed over all the items that we have seen so far. Moreover we do not know the total number of items in advance and we cannot store more than one item at any time.

   (a) (15 points) Consider the following algorithm. When the first item appears, we store it. When the k-th item appears, we replace the stored item with probability $1/k$. Show that this algorithm solves the problem.

   (b) (10 points) Now suppose that when the k-th item appears, we replace the stored item with probability $1/2$. What is the distribution of the stored item in this case?

3. (30 points) The following randomized algorithm returns the k-th smallest number from a set S of n distinct integers, for any k.

Algorithm 1

```
k-th order statistic $(S, k)$
 1: Select an item $a_i \in S$ uniformly at random
 2: for each item $a_j \in S$ do
 3:     Put $a_j$ in $S^-$ if $a_j < a_i$
 4:     Put $a_j$ in $S^+$ if $a_j > a_i$
 5: end for
 6: if $|S^-| = k - 1$ then            // $a_i$ is the k-th smallest item
 7:     return $a_i$
 8: else if $|S^-| \geq k$ then          // the k-th smallest item lies in $S^-$
 9:     k-th order statistic $(S^-, k)$
10: else                                 // the k-th smallest item lies in $S^+$
11:     k-th order statistic $(S^+, k - 1 - |S^-|)$
12: end if
```

(a) (5 points) Show how to compute the median of S using this algorithm.

(b) (25) Analyze the expected running time of `k-th order statistic` (S, k).

Optional hint: We will say that a subproblem is of type j if its input consists of at most $n\left(\frac{3}{4}\right)^j$ items but more than $n\left(\frac{3}{4}\right)^{j+1}$ items.

Upper bound the expected time of `k-th order statistic` on a subproblem of type j, excluding the time spent on recursive calls. Equivalently, upper bound the expected time until an item a_i is selected such that $1/4$ of the input can be thrown out (thus the input shrinks by a factor of $3/4$).

Then upper bound the expected running time of the entire algorithm by summing up the expected times spent on all subproblems.

4. (30 points) Consider the following experiment that proceeds in a sequence of *rounds*. For the first round we have n balls, which are thrown independently and uniformly at random into n bins. After round i, for $i \geq 1$, we discard every ball that fell into a bin by itself in round i. The remaining balls are kept for round $i + 1$, in which they are again thrown independently and uniformly at random into the n bins.

(a) Suppose that in some round we have $k = \epsilon n$ balls. At most how many balls should you expect to have in the next round?

Hint: use $e^x \geq 1 + x$ for real x.

(b) Assuming that everything proceeded according to expectation, prove that we would discard all the balls within $O(\log \log n)$ rounds.

Hint: Use the result in part (a) to derive a recurrence for the expected number of balls in the next iteration. Use $e^x \geq 1 + x$ for real x to simplify the recurrence. Note that the process ends when no balls are left to throw.

5. (30 points) Suppose that time is divided in discrete steps. There are n people and a single shared computer than can only be accessed by one person in a single step: if two or more people attempt to access the computer at the same step, then everybody is "locked out" during that step.

   At every step, each of the n people attempts to access the computer with probability p.

   (a) Determine the probability that a fixed person i succeeds in accessing the computer during a specific step j.

   (b) How would you set p to maximize the above probability?

   (c) For the choice of p in part (b), upper bound the probability that a fixed person i did not succeed to access the computer in *any* of the first $t = en$ steps.

   Hint: use the inequality $e^x \geq 1 + x$, *for all real* x.

   (d) What is the number of steps t required so that the probability that a fixed person i did not succeed to access the computer in *any* of the first t steps is upper bounded by an inverse polynomial in n (that is, by $1/n^k$ for some constant $k > 0$)?

   (e) How many steps are required to guarantee that *all* people succeeded to access the computer with probability at least $1 - 1/n$ (that is, with high probability)?

Recommended exercises: do NOT return, they will not be graded

1. Problem 7-2 in the textbook.

2. Problem 7-4 in the textbook.
   (If necessary, read pages 232-233 to refresh your memory on the definition of a stack.)

3. Consider the following online auction system. There are n bidding agents; agent i has an integer bid $b_i > 0$. All bids are distinct. The bidding agents appear in an order chosen uniformly at random. Each agent proposes its bid b_i in turn, and at all times the system maintains a variable b_{max} equal to the highest bid seen so far.

   How many times do you expect to update b_{max} when this process is executed?