CSOR W4231.002 – Spring, 2018

Homework 2

Out: Monday, February 5, 2018
Due: 8pm, Monday, February 19, 2018

You should always describe your algorithm clearly in English. You should also give pseudocode for all algorithms that you design. For all problems asking you to "design and analyze an algorithm", you must prove correctness and give the best upper bound that you can for the running time.

Collaboration is limited to discussion of ideas only. You should write up the solutions entirely on your own. You should list your collaborators on your write-up. If you do not type your solutions, be sure that your hand-writing is legible, your scan is high-quality and your name is clearly written on your homework.

1. (15 points) Let S and T be two disjoint subsets of size n 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).

2. (15 points) In this problem, we will analyze an algorithm that finds an item *close enough* to the median item of a set $S = \{a_1, \ldots, a_n\}$ of n distinct numbers. Specifically, the algorithm finds an item a_i such that at least $n/4$ items are smaller than a_i and at least $n/4$ items are greater than a_i.

Algorithm 1

Randomized Approximate Median(S)

```
Select an item a_i ∈ S uniformly at random
rank = 1
for j = 1 to n do
    if a_j < a_i then rank = rank + 1
    end if
end for
if n/4 ≤ rank ≤ 3n/4 then return a_i
else return error
end if
```

   (a) What is the running time of this algorithm?

   (b) What kind of algorithm is `Randomized Approximate Median` and why? What is the success probability of this algorithm?

   (c) How can you improve the success probability of the algorithm to over 99%? What is the running time of the new algorithm?

3. (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.

   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.

4. (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.

   (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 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 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?

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

5. (30 points) This problem considers the following variant of Randomized Quicksort. On input a set S of n distinct integers, the algorithm repeatedly selects an input item uniformly at random until it finds a *good item to partition* its input.

   Intuitively, a good item to partition the input is one that generates a *balanced* partition, that is, a partition where each side contains a constant fraction of the input items.

   We will call an item a *good partitioning item* if it is greater than at least $n/4$ of the input items and smaller than at least $n/4$ of the input items. Once such an item a_i is found, the algorithm recursively runs on inputs S^- and S^+, which are the sets of input elements that

are smaller and greater than a_i respectively. For small constant size $n \leq 3$, the algorithm sorts its input explicitly.

We call this algorithm `Randomized Quicksort-v1`. The pseudocode follows.

Algorithm 2

Randomized Quicksort-v1(S)

```
1: if |S| ≤ 3 then
2:     sort S
3:     output sorted list
4: end if
5: while no good partitioning element has been found do
6:     Select an element a_i ∈ S uniformly at random
7:     for each element a_j ∈ S do
8:         Put a_j in S^- if a_j < a_i
9:         Put a_j in S^+ if a_j > a_i
10:    end for
11:    if |S^-| ≥ |S|/4 and |S^+| ≥ |S|/4 then
12:        a_i is a good partitioning element
13:    end if
14: end while
15: Recursively call Randomized Quicksort-v1 on S^- and S^+
16: Output the sorted set S^-, then a_i, then the sorted S^+
```

We will now analyze the expected running time of this algorithm.

(a) What is the expected time to find a good partitioning element?

(b) What is the expected time of `Randomized Quicksort-v1` on a subproblem of size $|S|$, excluding the time spent on recursive calls?

(c) We will say that a subproblem is of type j if its input consists of at most $n\left(\frac{3}{4}\right)^j$ and at least $n\left(\frac{3}{4}\right)^{j+1}$ items.

i. For a fixed j, how much time is spent on a subproblem of type j?

ii. For a fixed j, how many subproblems of type j are there?

iii. For a fixed j, how much time is spent on all subproblems of type j?

(d) What is the expected running time of `Randomized Quicksort-v1`?

6. (20 points) Use the ideas from the previous problem to design and analyze the expected running time of a recursive randomized algorithm that returns the k-th smallest number in a set S of n distinct integers, for any k.

For example, for $k = \lceil n/2 \rceil$, your algorithm will return the median item.

Hint: Randomly select a partitioning item x. Check if x is the k-the smallest item; if so, return x, otherwise, recurse as appropriate. Analyze the expected time to find a good partitioning item and thus shrink the size of the subproblem by a factor of at least $3/4$.

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. (30 points) In this problem, you are given $n \times n$ matrices A, B, C and you wish to check whether $AB = C$.

   (a) (6 points) Give a $o(n^3)$ algorithm for the above task.

   (b) You will now design a faster randomized test for this task.

      i. (14 points) Let $\mathbf{x}$ be an n-dimensional vector with entries randomly and independently chosen to be 0 or 1, each with probability 1/2. Prove that if M is a non-zero $n \times n$ matrix, then $\Pr[M\mathbf{x} = \mathbf{0}] \leq 1/2$.

      ii. (10 points) Show that $\Pr[AB\mathbf{x} = C\mathbf{x}] \leq 1/2$ if $AB \neq C$. Then design and analyze a randomized algorithm to check whether $AB = C$.