cs473: Algorithms Assigned: *Wed., Sep. 25, 2019*

Problem Set #4

Prof. Michael A. Forbes
Prof. Chandra Chekuri

Due: *Thu., Oct. 3, 2019 (10:00am)*

All (non-optional) problems are of equal value.

1. (Multiplicative Chernoff Bound). Let $\mathsf{X}_1, \ldots, \mathsf{X}_n$ be independent random variables taking values over $[0,1]$. Let $\mathsf{X} = \sum_i \mathsf{X}_i$. Show the following.

   (a) For $r \in (-\infty, \ln 2]$, prove that $\mathbb{E}[\mathrm{e}^{r\mathsf{X}}] \le \mathrm{e}^{r\mathbb{E}[\mathsf{X}]+r^2\mathbb{E}[\mathsf{X}]}$, where you may use (without proof) that $1+x \le \mathrm{e}^x \le 1+x+x^2$ for such r.

   (b) Explain how the above used the independence of the X_i.

   (c) Apply Markov's inequality ($\Pr[\mathsf{Y} \ge a] \le \mathbb{E}[\mathsf{Y}]/a$) to $\mathrm{e}^{r\mathsf{X}}$, and optimize over r, to conclude that:

      i. For $0 \le \epsilon \le \ln 4$, $\Pr[\mathsf{X} \ge (1+\epsilon)\mathbb{E}[\mathsf{X}]] \le \mathrm{e}^{-\epsilon^2\mathbb{E}[\mathsf{X}]/4}$

      ii. For $\epsilon \ge \ln 4$, $\Pr[\mathsf{X} \ge (1+\epsilon)\mathbb{E}[\mathsf{X}]] \le 2^{-\epsilon\mathbb{E}[\mathsf{X}]/2}$

      iii. For $0 \le \epsilon \le 1$, $\Pr[\mathsf{X} \le (1-\epsilon)\mathbb{E}[\mathsf{X}]] \le \mathrm{e}^{-\epsilon^2\mathbb{E}[\mathsf{X}]/4}$

      iv. (Additive Chernoff Bound) For $\epsilon \ge 0$, $\Pr[|\mathsf{X} - \mathbb{E}[\mathsf{X}]| \ge \epsilon \cdot n] \le 2\mathrm{e}^{-\epsilon^2 n/4}$

   Note that the additive Chernoff bound suffices for applications such as estimating the errors in polling, but the multiplicative bound is in general stronger and often needed (e.g. consider $\mathbb{E}[\mathsf{X}] = \lg n$ and the resulting bound for $\Pr[\mathsf{X} \ge 2\mathbb{E}[\mathsf{X}]]$). Note also that the above omits one range of parameters, where one can show that $\Pr[\mathsf{X} \ge (1+\epsilon)\mathbb{E}[\mathsf{X}]] \le \mathrm{e}^{-(1+\epsilon)\ln(1+\epsilon)\mathbb{E}[\mathsf{X}]/4}$ if $\epsilon \ge 1$.

2. Consider a balls and bins experiment with $2n$ balls but only two bins. Each ball is thrown independently into a bin chosen uniformly at random. Let X_1 be the random variable for the number of balls in bin 1 and X_2 for bin 2. It is easy to see that $\mathbb{E}[\mathsf{X}_1] = \mathbb{E}[\mathsf{X}_2] = n$. We would like to have a handle on the difference $\mathsf{X}_1 - \mathsf{X}_2$. Our goal is to prove that for any fixed $\epsilon > 0$ there is a fixed constant $c > 0$ such that $\Pr[\mathsf{X}_1 - \mathsf{X}_2 \ge c\sqrt{n}] \le \epsilon$. By symmetry we can then argue that $\Pr[|\mathsf{X}_1 - \mathsf{X}_2| \ge c\sqrt{n}] \le 2\epsilon$.

   The below explores this random process through two different bounding methods in order to compare these methods.

   (a) Compute the variance of X_1. Then use Chebyshev's inequality to show that $\Pr[|\mathsf{X}_1 - n| \ge c\sqrt{n}] \le \epsilon$ for suitable choice of c for a given ϵ. What is the dependence of c on ϵ?

   (b) Use the Chernoff bound to show that $\Pr[|\mathsf{X}_1 - n| \ge c\sqrt{n}] \le \epsilon$. You need to use the bound separately for computing $\Pr[\mathsf{X}_1 \ge n + c\sqrt{n}]$ and for $\Pr[\mathsf{X}_1 \le n - c\sqrt{n}]$. What is the dependence of c on ϵ?

   (c) Using the preceding show that $\Pr[\mathsf{X}_1 - \mathsf{X}_2 \ge c\sqrt{n}] \le \epsilon$.

   (d) (**optional**, *not* for submission) A one-dimensional random walk on the integer line starts at position 0 on the number line. In each step we move from the current position one unit step to the left or one unit step to the right with equal probability (independent of the previous choices). Let Z_n be the position of the walk after n steps (it is an integer in the range $[-n, n]$). Using a simple connection to the problem of throwing balls into two bins show that for any fixed ϵ, there is a c that depends only on ϵ such that $\Pr[|\mathsf{Z}_n| \geq c\sqrt{n}] \leq \epsilon$. Also derive that $\mathbb{E}[|\mathsf{Z}_n|] \leq O(\sqrt{n})$.

3. Consider the following geometric problem: given a set P of n points in two-dimensions, with integer coordinates from $\{0, 1, \ldots, U-1\}$, find a *closest pair* — two points $p \neq q \in P$ such that the (euclidean) distance between p and q is the smallest. We denote the distance of the closest pair by $\delta(P)$.

   An $O(n^2)$-time algorithm for this problem is trivial, and you can find an $O(n \log n)$-time divide-and-conquer algorithm for two-dimensions in some textbooks. In this question, we give a different, faster randomized algorithm (which has the added advantage that it can be extended to higher dimensions and to other problems).

   (a) First give an $O(n)$-expected-time (Las Vegas) algorithm for the easier *decision problem*: given a value r, decide whether $\delta(P) < r$.

   Hint: Build a uniform grid where each cell is an $\frac{r}{2} \times \frac{r}{2}$ square. Use hashing. How many points can a grid cell have? For each grid cell, how many grid cells are of distance at most r?

   (b) Now, consider the following recursive Las Vegas algorithm to compute $\delta(P)$:

```
closest-pair($P$):
    if $|P| \leq 100$ then return answer by brute force
    partition $P$ into subsets $P_1, \ldots, P_{20}$ each with at most $\lceil n/20 \rceil$ points
    $S = \{(i,j) \mid 1 \leq i < j \leq 20\}$
    $r = \infty$
    for $(i,j) \in S$ in random order
        if $\delta(P_i \cup P_j) < r$
            $r =$closest-pair($P_i \cup P_j$)
    return $r$
```

   Explain why the algorithm is always correct, and analyze its expected running time by solving a recurrence.

   Hint: Where is the first part used? What is the size of S? How many times (in expectation) does the function recursively call itself?

4. (**optional**, *not* for submission) Consider the following variant of quicksort. Given an array A of n numbers (which we assume are distinct for simplicity) the algorithm picks a pivot x uniformly at random from A and computes the rank of x. If the rank of x is between $n/4$ and $3n/4$ (call such a pivot a *good* pivot) it behaves like the normal quicksort in partitioning the array A and recursing on both sides. If the rank of x does not satisfy the desired property (the pivot picked is not *good*) the algorithm simply repeats the process of picking the pivot until it finds a good one. Note that in principle the algorithm may never terminate!

   (a) Write a formal description of the algorithm.

   (b) Prove that the expected run time of this algorithm is $O(n \log n)$ on an array of n numbers.

   (c) Prove that the algorithm terminates in $O(n \log n)$ time with high probability. Does this immediately imply that the expected run time is $O(n \log n)$?