HW 6

Due on **Friday, October 13, 2023 at 9pm**

CS 473: Algorithms, Fall 2023 Version: **2.0**

Submission instructions: As in previous homework.

16 (100 PTS.) Concentration of mass.

16.A. (40 PTS.) Let $k > 0$ be a positive integer, and consider a sequence $\mathsf{s} = (s_1, s_2, \ldots, s_{2k}) \in [\![n]\!]^{2k}$, where $[\![n]\!] = \{1, 2, \ldots, n\}$. Such a sequence s is ***even*** if a every value in s appears exactly even number of times. Thus, for example, $\mathsf{s} = (1, 17, 13, 17, 1, 13)$ is even, while $\mathsf{s} = (1, 13, 13, 17, 1, 13)$ is not.

Let $F(n, 2k)$ be the number of even sequences in $[\![n]\!]^{2k}$. Prove that $F(n, 2k) \leq n^k f(k)$, where $f(k)$ is a function that depends only on k. Your upper bound on $f(k)$ should be as small as possible. Your argument should be self contained and elementary.

16.B. (20 PTS.) Following on what was seen in class, consider the independent random variables $X_1, \ldots, X_n \in \{-1, +1\}$, where $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$. Let $Y = \sum_{i=1}^{n} X_i$. Prove that $\mathbb{E}\left[Y^{2k}\right] = F(n, 2k)$, and that $\mathbb{P}\left[|Y| \geq t\big(F(n, 2k)\big)^{1/2k}\right] \leq 1/t^{2k}$, for any $t \geq 1$.

(Hint: In class, see scribbles/notes/lecture, we did the case $k = 1$. Go through the argument and extend it to larger values of k.)

16.C. (40 PTS.) Using the previous part (and only the previous part), prove that there exists some absolute constant $c > 0$, such that for any even integer $\alpha > 0$, we have

$$\mathbb{P}\left[|Y| \geq c\sqrt{\alpha n \log_2 n}\right] \leq \frac{1}{n^\alpha}.$$

(For simplicity, you can assume that $\log_2 n$ is an integer.)

Partial credit would be given to proofs of a weaker bound $\mathbb{P}\left[|Y| \geq c\sqrt{\alpha n}(\log_2 n)^\beta\right] \leq \frac{1}{n^\alpha}$, where β is some absolute constant.

17 (100 PTS.) As simple as 1,2,3.

You are given a full tree T_h of degree 7 and height h – every node has 7 children. Each leaf of the tree stores a value that is either $1, 2$ or 3. To compute the value of an internal node v, recursively, consider the value of all its children. This form a multiset S_v with 7 elements, with numbers in $\{1, 2, 3\}$. The value of v is the value in S_v that appears the largest number of times. There are two additional assumptions we are going to make about these 7 values:

(i) If two values appear the same number of times (and the other value appear strictly less number of times), then the larger of the two is the returned value.

(ii) No two values appear exactly twice. (Thus, $2, 2, 3, 3, 1, 1, 1$ is not legal.)

17.A. (50 PTS.) Consider an algorithm that evaluates the children of a node recursively in a random order, and returns the value the node as soon as it is determined. Prove that for any fixed values of the children, there is always an ordering such that the algorithm does not need to evaluate all the children before the returned value is determined. In particular, consider a tree T_1 (of height 1, naturally), and let Y_1 be the random variable that is the number of leafs the algorithm reads before computing the value of the tree (i.e., the value of the root). Prove that $\mathbb{E}[Y_1] < 7$.

17.B. (50 PTS.) Let $n = 7^h$. Prove that the above randomized algorithm computes the value of T_h in time $O(n^\alpha)$, where $\alpha < 1$. What is the value of α?

(Importantly, you can assume nothing (beyond what specified above) on the values stored in the leafs of the tree – they are fixed not random!)

18 (100 PTS.) With a little help from an oracle.

Let $P \subseteq \mathbb{R}^4$ be a set of n points. The ***diameter*** of P is $\Delta(P) = \max_{p,q \in P} \|p - q\|$. Clearly, one can compute the diameter in quadratic time, but here we are interested in a faster algorithm.

18.A. (20 PTS.) Describe an algorithm, as fast as possible, that for any integer t, $1 < t < \lfloor n/2 \rfloor$, computes sets $P_1, \ldots, P_m \subseteq P$, such that:

(i) $\Delta(P) = \max_i \Delta(P_i)$,
(ii) for all i: $|P_i| \leq n/t$, and
(iii) $m = O(t^2)$.

What is the running time of your algorithm?

(Remark: This readily gives a recursive algorithm for computing the diameter – indeed, compute the above partition, and recursively compute $\Delta(P_i)$, for all i, and return the maximum. Unfortunately, this algorithm still has quadratic running time.)

18.B. (20 PTS.) Let $\alpha_1, \alpha_2, \ldots, \alpha_t$ be a sequence of numbers. Let $\pi : [\![t]\!] \to [\![t]\!]$ be a random permutation. Let

$$\beta_i = \max_{\ell=1}^{i} \alpha_{\pi(\ell)}.$$

Let $Y_i = 1 \iff \beta_i > \beta_{i-1}$. Let $Z = \sum_{i=1}^{t} Y_i$. Provide an upper bound, as small as possible, on $\mathbb{E}[Z]$. In words, when randomly permuting a sequence of t numbers (not necessarily distinct), and computing their maximum by scanning the permuted sequence, how many times (in expectation) does the maximum change?

18.C. (60 PTS.) Assume you have an oracle Verify, that given a set $U \subseteq P$, and a real number r, can tell you, say in $O(u \log u)$ time, whether $\Delta(U) > r$, $\Delta(U) = r$, or $\Delta(U) > r$, where $u = |U|$.

Describe a randomized algorithm, as fast as possible, that computes $\Delta(P)$. Specifically, use the scheme from the previous part, to break the problem into subproblems, and then start solving the subproblems recursively, but only if you need to, by calling Verify first. Finally, use randomization to make sure the expected number of recursive calls is small. **Prove** the upper bound on the expected running time of your algorithm.

(Hint: It is enough to take t to be a very large constant.)