CS 473: Algorithms, Fall 2018

Midterm I: October 2, 2018, 7pm-9pm, DCL 1320

Version: **1.1**

Name:	
Netid:	

- This is a closed-book, closed-notes, open-brain exam. If you brought anything with you besides writing instruments and your **handwritten** $8\frac{1}{2}'' \times 11''$ cheat sheet, please leave it at the front of the classroom.
- Please **print** your name, and netid in the boxes above. Print your name at the top of every page (in case the staple falls out!).
- **You should answer all the questions on the exam.**
- The last few pages of this booklet are blank. Use that for a scratch paper. Please let us know if you need more paper.
- If your cheat sheet is not hand written by yourself, or it is photocopied, please do not use it and leave it in front of the classroom.
- Submit your cheat sheet together with your exam. An exam without your cheat sheet attached to it will not be checked or even graded.
- If you are NOT using a cheat sheet you should indicate it in large friendly letters on this page.
- There are 4 questions on the exam. Each question is worth 25 points.
- Answers containing *only* the expression: "I don't know", will get 25% of the points of the question. If you write anything else, it would be ignored. Overall, points given for "I don't know" will not exceed 10 points for the whole exam.
- Write your exam using a pen (please do not use an invisible ink or a pencil).
- Time limit: 110 minutes.
- Relax.

I feel like a spinning top for a dreidel
The spinning don't stop when you leave the cradle
You just slow down
Round and around this world you go
Spinning through lives of the people you know
We all slow down
– Don McLean

1 Beware of oracles coloring graphs. (25 pts.)

[A question from the example midterm with a small twist]

You are given an oracle $f(\mathsf{H})$, which in constant time can tell you whether or not the input graph H is 7-colorable. Describe an algorithm, using this oracle, as efficient as possible, that outputs for a given graph G with n vertices and m edges, the 3-coloring (note the $\mathbf{3}$ in the desired coloring vs. the $\mathbf{7}$ in the oracle) of the graph if it exists. Namely, it either outputs that G can not be colored by three colors, or it output a valid 3-coloring for the vertices of G. What is the running time of your algorithm?

2 Locating burger joints. (25 pts.)

[Similar to homework problem.]

The McYucky chain wants to open several restaurants along Red street in Shampoo-Banana. The possible locations are at $L_1, L_2, \ldots, L_n$ where L_i is at distance m_i meters from the start of Red street. Assume that the street is a straight line and the locations are in increasing order of distance from the starting point (thus $0 \leq m_1 < m_2 < \ldots < m_n$). McYucky has collected some data indicating that opening a restaurant at location L_i will yield a profit of p_i independent of where the other restaurants are located. However, the city of Shampoo-Banana has a zoning law which requires that any two McYucky locations should be D or more meters apart. Moreover, McYucky has realized that its budget constraints imply that it can open at most $k < n$ restaurants. Describe an algorithm that McYucky can use to figure out the locations for its restaurants so as to maximize its profit while satisfying its budget constraint and the city's zoning law.

3 A point above all given lines. (25 pts.)

[A combination of two algorithms seen in class.]

Let L be a set of n lines in the plane. Assume the lines are in general position (no line is horizontal, no line is vertical, no three lines meet in a point). Let $\ell^-, \ell^+ \in L$ be the two lines with the maximum and minimum slope in L, respectively. Furthermore, assume that ℓ^+ has a positive slope, and ℓ^- has a negative slope. The task at hand is to compute the lowest point that is above all the lines of L.

Consider the algorithm that randomly permute the lines of L, and let $\ell_1, \ldots, \ell_n$ be the resulting ordering. Let p_i be the lowest point that is above all the lines of $L_i = \{\ell^+, \ell^-\} \cup \{\ell_1, \ldots, \ell_i\}$. The algorithm works as follows. It computes $p_0 = \ell^- \cap \ell^+$. In the ith iteration, for $i = 1, \ldots, n$, it does the following:

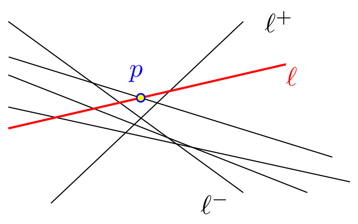
(I) Check if p_{i-1} is above ℓ_i. If so, it sets $p_i \leftarrow p_{i-1}$ and continue to the next iteration. (That is, p_{i-1} is still the lowest point above all the lines of L_i.)

(II) Otherwise, p_i must be on ℓ_i. It computes in $O(i)$ time the lowest point on ℓ_i that is above all the lines of L_{i-1}. It sets p_i to be this point.

The purpose of this question is to describe and analyze this algorithm in more detail.

3.A. (5 pts.) Given a set T of i lines (in general position), such that $\ell^+, \ell^- \in T$, and a line ℓ (which has, say, a positive slope), describe **shortly** how to compute in $O(i)$ time, the lowest point p (i.e., point with minimum value of the y axis) that lies on ℓ, and is above all the lines of T (you can safely assume that such a point exists). (This shows how to implement step (I) above.)

(Hint: First consider the case that ℓ has a positive slope.)



3.B. (5 pts.) Give an upper bound to the probability that $p_i \neq p_{i-1}$.

3.C. (10 pts.) Let R_i be the running time of the ith iteration. Prove that $\mathbb{E}[R_i] = O(1)$.

3.D. (5 pts.) Provide and prove an upper bound on the expected running time of the algorithm.

4 2SUM, 3SUM, Fast Sum. (25 pts.)

[Similar to homework problem.]

4.A. (5 pts.) Describe a *simple* algorithm that determines whether a given set S of n distinct integers contains two elements whose sum is zero, in $O(n \log n)$ time.

4.B. (10 pts.) Describe an algorithm that determines whether a given set S of n distinct integers contains three (distinct) elements whose sum is zero, in $O(n^2)$ time. For simplicity, you can assume that S does not contain the number 0.

[A correct full solution with running time $O(n^2 \log n)$ would get 12 points. A solution with $O(n^3)$ time would get as many points as IDK.]

4.C. (10 pts.) Now suppose the input set X contains only integers between 1 and $1,000n$. Describe an algorithm, as fast as possible (way faster than (B)), that determines whether X contains three (distinct) elements $x, y, z \in S$, such that $x + y = z$. What is the running time of your algorithm?

*** Scratch paper - do not detach from the booklet ***