CS 473: Algorithms, Fall 2021

Sariel Har-Peled

Midterm I: September 28, 15:30-17:25, 2021, SC 1404

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 q 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 page of this booklet is 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.
- Please submit your cheat sheet together with your exam.
- 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.
- There is **NO** credit given for IDK ("I don't know").
- Write your exam using a pen (please do not use an invisible ink or a pencil).
- Time limit: 115 minutes.
- Relax.

'Would you know how to calculate the diameter of the globe?'

'No, I'm afraid I wouldn't,' answered Svejk, 'but I'd like to ask you a riddle myself, gentlemen. Take a three-storied house, with eight windows on each floor. On the roof there are two dormer windows and two chimneys. On every floor there are two tenants. And now, tell me, gentlemen, in which year the house-porter's grandmother died?'

– The Good Soldier Svejk, Jaroslav Hasek

1 Beware of oracles traveling in 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 the input graph H has a Hamiltonian cycle. Describe an algorithm, using this oracle, as efficient as possible, that outputs for a given graph G with n vertices and m edges, the Hamiltonian cycle in the graph if it exists. Namely, it either outputs that G contains no Hamiltonian cycle, or it outputs a valid Hamiltonian cycle of G.

What is the running time of your algorithm? Argue why your algorithm is correct.

2 Maximum sub-array revisited. (25 pts.)

It turns out that the divide & conquer algorithm seen in class for maximum sub-array is pretty silly. As a reminder, the input is an array $X[1 \,..\, n]$ of n numbers. The *value* of $X[i \,..\, j]$ is

$$v(i,j) = \sum_{t=i}^{j} X[t].$$

The task at hand is to compute the maximum such value – that is output the value $\max_{i,j} v(i,j)$.

Give an algorithm, **as fast as possible**, that uses dynamic programming that solves this problem.

Argue the correctness, and bound the running time of your algorithm. How much space does your algorithm uses if one ignores the space used by the input (the less space it uses, the better).

3 A bottleneck distance. (25 pts.)

[Similar to stuff seen in class.]

The input is an undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be with n vertices and m edges, and a start vertex s and a target vertex t (you can assume G is connected). The edges of G are weighted – an edge $e \in \mathsf{E}$ has weight $\omega(e)$. Given a parameter ℓ, let

$$\mathsf{G}_{\leq \ell} = \Big(\mathsf{V}, \{uv \in \mathsf{E}(\mathsf{G}) \mid \omega(\mathsf{G}) \leq \ell\}\Big)$$

be the subgraph including only the edges of weight $\leq \ell$. A value ℓ is ***good***, if s is connected to t in $\mathsf{G}_{\leq \ell}$.

Describe an algorithm, as fast as possible, that computes the minimum value ℓ which is still good.

Prove/argue that your algorithm is correct.

[A correct sub-optimal (but close to optimal) solution (with a correct argument about correctness) is worth at least 15 points. There are many solutions to this problem.]

4 SUMba. (25 pts.)

4.A. (10 pts.) Given two sets X and Y of at most n numbers, describe an algorithm, as fast as possible, that decides if there is a number $x \in X$ and $y \in Y$, such that $x + y = 0$.

4.B. (15 pts.) You are given a full binary tree T with a root r of height h with n nodes. (Thus T has exactly 2^h leafs, and $2^h - 1$ internal nodes.) There are weights on the edges (the weights can be positive and negative [but they are not zero]). Describe an algorithm, as fast as possible, that decides if there is a non-empty path in T that has weight exactly zero (non-empty here means the path must include at least one edge).