CS 473: Algorithms, Fall 2023 Sariel Har-Peled

midterm i: september 28, 7-9pm, 2023, Loomis 151 Version: **0.9999999**

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. The cheat sheet can not be printed or a photocopy.
- 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.
- 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 Runners. (25 pts.)

(Based on a homework question.)

There are n runners $r_1, \ldots, r_n$ sorted in decreasing order by their initial speed (so initially r_1 is the leader when the race starts, r_2 is in second place, etc). However, runner r_i, get tired and sooner or later all the (initially) slower runners are going to pass it by (and it is also going to pass all the $i-1$ initially faster runners). Two runners pass each other exactly once during the race. In particular, assume that you are given a function $t(i, j) > 0$, defined for all $i < j$, that in constant time, tells you the time when runner i passes runner j (you can safely assume all the values of t are unique).

Given times $\alpha_1 < \alpha_2 < \cdots < \alpha_n$, describe an algorithm, as fast as possible, that outputs the leading runner at each time α_i, for $i = 1, \ldots, n$. What is the running time of your algorithm? Provide pseudo-code for your algorithm. Argue/prove that your algorithm is correct.

(hint: First describe how to compute the leader of the race at a specified time $\alpha > 0$.)

2 3SUM (25 pts.).

(Taken from the review questions.)

Consider two sets $B, C \subseteq [\![10n]\!] = \{1, 2, \ldots, 10n\}$, where $|B| = n$ and $|C| = n$ (they are not sorted). We wish to compute the ***Cartesian sum*** of B and C, defined by

$$D = \{b + c \mid b \in B \text{ and } c \in C\}.$$

Note that the integers in D are in the range from 2 to $20n$. Describe an algorithm, **as fast as possible**, that computes the set D. Argue/prove your algorithm is correct. What is the running time of your algorithm?

3 Beware of oracles. (25 pts.)

[Similar to question from the example midterm + stuff coveerd in class.)

You are given an oracle $f(\mathsf{H}, k)$, which in constant time either returns yes or no. In particular, $f(\mathsf{H}, k)$ returns yes $\iff$ H has a vertex-cover of size k.

You are given a graph G (with n vertices and m edges), a parameter k, and the oracle f. Describe an algorithm, as fast as possible, that returns a set $X \subseteq \mathsf{V}(\mathsf{G})$ that is a vertex-cover of G, where $|X| \leq k$, if such a set exists. As a reminder, the set X is a ***vertex-cover*** for G, if for all $\{u, v\} \in \mathsf{E}(\mathsf{G})$, we have that $|\{u, v\} \cap X| > 0$. Explain why your algorithm is correct.

For full credit, your algorithm should also perform as few oracle queries as possible, how many oracle queries does your algorithm perform?

(Hint: Consider a single edge uv of G and what the vertex-cover of this edge is.)

4 Quasi-convex. (25 pts.)

(Second part on the other side of the page.)

4.A. (10 pts.) An array $A[1..n]$ of real numbers is ***quasi-convex*** (QC), if for any value α, the values in A that are at most α form a consecutive block. Formally, for all $\alpha \in \mathbb{R}$, we have that the set

$$F(\alpha) = \{i \mid A[i] \leq \alpha\}$$

has the property that $F(\alpha) = [\![\beta \mathinner{..} \gamma]\!] = \{\beta, \beta+1, \ldots, \gamma\}$, where β, γ are some integers that depends on α.

Describe an algorithm, as fast as possible, that computes the minimum value in A. Provide pseudo-code for your algorithm. Explain why your algorithm is correct. What is the running time of your algorithm? (You can assume all the values in A are distinct.)

(Second part on the other side of the page.)

4.B. (15 PTS.) You are given a matrix $M[1 \mathinner{..} n][1 \mathinner{..} m]$ that is quasi-convex (QC). A matrix is ***quasi-convex*** if it is QC on all rows, and all columns (individually), and in addition, for any value α, the set

$$G(\alpha) = \{(i,j) \mid M[i,j] \leq \alpha\}$$

is connected. A set $X \subseteq [\![n]\!] \times [\![n]\!]$ is ***connected***, if for any $p, q \in X$, there exists a sequence $p = p_1, p_2, \ldots, p_t = q$, such that $p_i \in X$, for all i, and p_i and p_{i+1} are adjacent (i.e., $\|p_i - p_{i+1}\| = 1$). Here is an example of such a matrix M (with the set $G(24)$ in gray):

i,j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	55	60	64	66	68	70	77	101	116	120	121
2	49	33	28	26	**23**	**22**	45	81	100	114	119
3	48	30	**14**	**9**	**7**	**18**	41	73	93	107	118
4	50	29	**12**	**4**	**2**	**16**	39	67	89	105	117
5	53	31	**11**	**3**	**1**	**10**	35	63	87	103	115
6	54	32	**13**	**5**	**6**	**8**	27	57	84	102	113
7	56	34	**15**	**17**	**19**	**20**	**24**	51	83	98	112
8	58	36	**21**	25	37	40	43	46	80	97	111
9	59	38	42	44	47	61	69	74	78	96	110
10	62	52	65	71	75	79	85	88	91	94	109
11	72	76	82	86	90	92	95	99	104	106	108

Describe an algorithm, as fast as possible, that computes the minimum value in M. **Prove** that your algorithm is correct. What is the running time of your algorithm? You can assume all the values in M are distinct.

Provide pseudo-code of your algorithm.