CS 473: Algorithms, Fall 2021

Sariel Har-Peled

Midterm II: November 9, 15:30-17:25, 2021, SC 1404

Version: **1.0**

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 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.

At the sight of the still intact city, he remembered his great international precursors and set the whole place on fire with his artillery in order that those who came after him might work off their excess energies in rebuilding.
– The tin drum, Gunter Grass

1 Short questions/answers.

Provide **as short as possible** answers to the following questions (no need to provide explanations). Specifically, you answer should be at most one line per question.

1.A. (5 pts.) Given two arrays of numbers $A[1 \,..\, n]$ and $B[1 \,..\, n]$, let

$$N = \{(i, j) | i, j \in [\![n]\!]\,,\ \text{and}\ \ i + j \text{ is even}\}$$

how fast can one compute $\sum_{(i,j)\in N} A[i]B[j]$? And using which technique?

1.B. (5 pts.) If a sorting network sorts correctly all inputs from $\{2, 7\}^n$, then it sorts correctly all inputs. This statement is correct or false?

1.C. (5 pts.) In class, we have seen algorithms for computing a min-cut with running time $O(n^2 \log^3 n)$. Using the algorithms seen in class, one can (with high probability) compute all the min-cuts in a graph in what time? [Faster is better.]

1.D. (5 pts.) Using hashing one can sort real numbers in linear time. This statement is true or false?

1.E. (5 pts.) Let $x_1, \ldots, x_n$ be a sequence of n distinct numbers, and let π be a random permutation. Let $F(i)$ be the set of k smallest numbers among $x_{\pi(1)}, \ldots, x_{\pi(i)}$. What exactly is the probability that $F(i) \neq F(i-1)$, for $i \geq k$?

2 Slide me down. (25 pts.)

[A question about probability.]

2.A. (5 pts.) Let X and Y be two random variables. The variable X is picked uniformly from $[0, n]$, and then Y is picked uniformly at random from $[0, X]$. What is the expected value of Y – that is $\mathbb{E}[Y]$?

2.B. (20 pts.) Consider an undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ over n vertices, where $\mathsf{V} = \{v_1, \ldots, v_n\}$. The algorithm randomly assigns each vertex $v \in \mathsf{V}$ a random priority $f(v)$ uniformly and independently from the range $[0, n]$.

For an input vertex v, consider the following walk. Initially $u_1 = v$.

In the ith iteration, if $f(u_i)$ is smaller than 1, or smaller than $\min_{x \in N(u_i)} f(x)$, then we stop (we reached a local minimum), where $N(u_i)$ is the set of the vertices in G that are adjacent to u_i. Otherwise, the algorithm computes the minimum index α, such that $v_\alpha \in N(u_i)$ and $f(v_\alpha) < f(u_i)$. The algorithm then sets u_{i+1} to be v_α, and continues to the next iteration.

Prove an upper bound, as tight as possible, on the length of the walk $u_1, u_2, \ldots$. The upper bound should hold with high probability.

[Hint: Let $Y_i = f(u_i)$. Consider the expectations $\mathbb{E}[Y_1], \mathbb{E}[Y_2], \ldots$.]

3 Independent sets. (25 pts.)

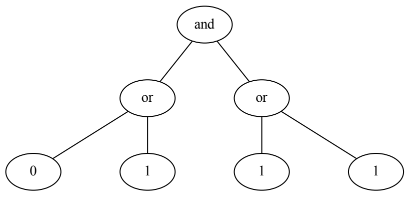
(Similar to things seen in class.)

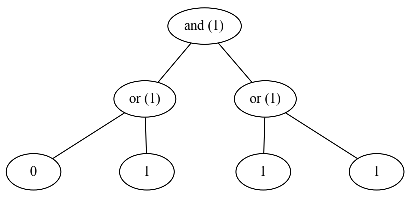
You are given a set system $(\llbracket n \rrbracket, \mathcal{F})$, where $\mathcal{F} = \{F_1, \ldots, F_m\}$, where $F_i \subseteq \llbracket n \rrbracket = \{1, \ldots, n\}$ for all i. A subset $\mathcal{I} \subseteq \mathcal{F}$ is ***independent***, if for all $i \in \llbracket n \rrbracket$, there is at most one set in $\mathcal{I}$ that contains it. The *size* of $\mathcal{I}$ is the number of sets in $\mathcal{I}$ (that is $|\mathcal{I}|$). Assume that every set in $\mathcal{F}$ is of size at most four. Describe a polytime time algorithm (in n and m) that approximates the largest set $\mathcal{I} \subseteq \mathcal{F}$ that is independent. What is the running time of your algorithm? What is the quality of approximation provided by your algorithm? Prove your bound.

4 AND OR OR? (25 PTS.)

(An easier version of a question given in homeworks.)

You are given a complete binary tree of height $2h$. The input to the tree are bits stored at the leafs (there are $n = 2^{2h}$ such inputs). The nodes alternate between `and`/`or` gates as one goes down the tree, where the root is an `and` gate, its children are an `or` gates, and so on. The value of a node, is the value of its two children, and the appropriate boolean operation (`or`/`and`) applied to these values – according to the label of the node. As usual, our purpose is to read as few leafs of the tree as possible, while still evaluating correctly the value output by the root node. Here is an example:





One can show (but you do not have to do it!) that any deterministic algorithm needs to read all the leafs of the tree, to compute the value at the root.

4.A. (15 PTS.) Describe a randomized algorithm that computes the value in the root of the tree, for the case $h = 1$ (i.e., this is a tree with four leafs). Let γ be the expected number of leafs your algorithm reads. For credit, your algorithm should have $\gamma < 4$. Prove your answer.

[Hint: The analysis here is a bit tricky. First consider the case that the tree evaluates to 0, and then consider the case that the tree evaluates to 1.]

4.B. (10 PTS.) Describe an algorithm that computes the value in the root of the tree, for the general case ($h > 1$). Your algorithm should read the value stored in as few leafs as possible.

What is the expected number of leafs your algorithm read as function of $n = 2^{2h}$?

[You can state the running time as a function of γ, even if you had not solved the first part. You just need to provide a short formula for the constants involved – no need to provide an exact value. Finally, feel free to use the formula $\log_x y = (\ln y)/(\ln x)$.]