
Homework 1 – Due Thursday, January 30 *before 11am* on Gradescope

Instructions

- Solutions written in L^AT_EX are strongly preferred, but you can upload any pdf files, including scanned hand-written solutions. Template latex files are on the course webpage.
- Collaboration is allowed and encouraged. However, each of you should think about a problem before discussing it with others and write up your solution independently. You may consult books and online sources to get information about well-known theorems, such as the Chernoff bound. But you are not allowed to look up solutions directly in papers or any other sources. And you *must* list all collaborators and sources! (See full details in the General Information Handout.)
- Correctness, clarity, and succinctness of the solution will determine your score.

Problems

1. A *tournament* is a directed graph that contains exactly one edge for each pair of vertices. There are two possible orientations for each edge. (Think of vertices as representing competitors in a tournament where every pair of competitors plays exactly one match, and the direction of the edge encoding the outcome of the match.) Suppose an n -vertex tournament G is represented by a $n \times n$ matrix in which an entry (u, v) is 1 if G contains the edge (u, v) and -1 otherwise (that is, if G contains the edge (v, u)). A *sink* is a node u such that u 's row contains only 1s.

Give an algorithm to find a sink in a tournament, if it exists, with $O(n)$ queries to the matrix.

2. In class, we saw an algorithm for testing whether an image specified by an $n \times n$ matrix of 0-1 entries represents a half-plane. Consider a 1-dimensional version of this problem: You are given an n -bit string, and you have to test whether it belongs to the set $0^*1^* \cup 1^*0^*$, that is, has 0s followed by 1s or 1s followed by 0s.
 - (a) Give a property tester for this problem with query complexity $O(1/\epsilon)$ following the Testing by Implicit Learning paradigm.
 - (b) Give a different property tester for this problem, with the restriction that the tester can only make queries to uniformly random positions in the string (independently). Try to get the best query complexity you can (in the asymptotic sense).
3. Your friend Kabir proposes the following tester for sortedness of a list of n numbers: *Query s positions from $[n]$ uniformly and independently at random and **reject** if the numbers in these positions are out of order.*
 - (a) Show that one of the two examples of hard lists for sortedness is also hard for this tester. Specifically, prove a lower bound of the form $s = \Omega(n^c)$ on the number of samples (i.e., queries) you need, where c is as large as you can make it.

Hints: 1. It resembles the birthday paradox.
2. You may use the inequality $1 - x \geq e^{-1.5x}$ for all $x \in (0, 1/2]$ without proof.

- (b) Analyze a property tester for sortedness based on Kabir's idea. If you can, make your upper bound on the number of queries match the lower bound from part (a) in terms of the asymptotic dependence on n .

Hints: 1. Consider the following undirected graph G , called *the violation graph*: the set of nodes is $[n]$, and for each pair $(i, j) \in [n]^2$, where $i < j$, the graph G contains the edge (i, j) iff $x_i > x_j$. (Recall that, in the lecture, we represented the input as x_1, \dots, x_n .)

2. Relate the distance of (x_1, \dots, x_n) to sortedness to the size of the minimum vertex cover of the violation graph built from this input.
3. Use the fact that the size of the minimum vertex cover is at most twice the size of the maximum matching in any graph. (You don't need to prove it).
4. The birthday paradox to the rescue again!