Homework 1 – Due Thursday, January 26 *before* 10am on Angel

(For students taking candidacy exams: due Tuesday, January 31 *before* 10am on Angel)

Instructions

- Solutions written in LaTeX 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 on-line 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!
- 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, based on spanners, for testing if a list of numbers $x_1, \ldots, x_n$ is sorted. Now we will design another algorithm for this problem, based on binary search.

   BinarySearchSortednessTest(n, ϵ)
   1 Pick an index i from $\{1, 2, \ldots, n\}$ uniformly at random and read the number x_i.
   2 Perform a binary search for x_i and **reject** if you find any numbers out of order.

   (a) Analyze the probability that BinarySearchSortednessTest(n, ϵ) rejects a list that is ϵ-far from sorted. **Hint:** Call a number x_i *bad* if it "fails" the binary search, i.e., Step 2 performed on x_i would reject.

   (b) Prove that, with enough repetitions, BinarySearchSortednessTest(n, ϵ) is an ϵ-tester for sortedness.

   (c) How does the query complexity and running time of this test compare to those of the test we saw in class?

   (d) An algorithm is called *nonadaptive* if it makes its queries in advance, before getting any answers. Otherwise, it is called *adaptive*. Was the spanner-based algorithm adaptive? Is your new algorithm adaptive? **Hint:** One of them is adaptive and the other is nonadaptive. Can you modify the adaptive algorithm to make it nonadaptive without changing its running time?

3. In class we saw a tester for connectedness that made $O(\frac{1}{\epsilon^2 d})$ queries. Give a tester for connectedness that makes $O(\frac{1}{\epsilon}\text{polylog}\frac{1}{\epsilon^2 d})$ queries, where polylog m means that there is a constant c such that the expression is $\log^c m$.

   Hint: In class we proved that if a graph is ϵ-far from connected, it has many small connected components. ("Many" was $\geq \frac{\epsilon dn}{8}$ and "small" was of size $\leq \frac{8}{\epsilon d}$.) Try to do a more careful accounting by considering small components of different sizes separately. I.e., break components into *buckets* according to their size (1, 2 to 3, 4 to 7, etc.) and prove that at least one of the buckets contains "enough" components. Modify the test accordingly.

4. An *image* is an $n \times n$ matrix of 0/1 pixel values where 0 represents white and 1 represents black. To keep the correspondence with the plane, index the matrix by $\{0, 1, \ldots, n-1\}^2$, with the lower left corner being $(0, 0)$ and the upper left corner being $(0, n-1)$. In class, we gave an algorithm for testing if an image is a half-plane.

   (a) The *image graph* $G_M = (V, E)$ of an image matrix M has vertex set $V = \{(i, j) | M_{ij} = 1\}$ and edge set $E = \{((i_1, j), (i_2, j)) \mid |i_1 - i_2| = 1\} \cup \{((i, j_1), (i, j_2)) \mid |j_1 - j_2| = 1\}$. In other words, the image graph consists of black pixels connected by the grid lines. The image is *connected* if its image graph is connected.

   Prove that an $n \times n$ image contains at most p connected components, they can be linked into one component by changing at most $n(\sqrt{2p} + O(1))$ pixel values from white to black.

   (b) Adopt the tester for connectedness of graphs to test for connectedness of images. (Be careful: the tester is almost the same, but its analysis is different because the distance between two images is measured differently than the distance between two graphs in the bounded degree model.) Note that the trick from problem 3 can be applied here, too, but you can use the slower algorithm, explained in class.

   (c) Define a property of an image that would be interesting to test. Make sure your property is not trivially testable, i.e., for example, not all images are close to having the property. Just to get you started, some interesting things we might want to find out about images are whether they fit a certain template or whether they can be partitioned in a certain way.

   Hint: You don't need to give an algorithm, so you might try to impress me with how interesting your property is. Imagine you are a professor and you are trying to come up with a research problem for your graduate student. (If you say "having the same number of black and white pixels", it does answer the question, but is not likely to get your graduate student a paper: for one thing, she is going to solve it on the spot, and she is going to wonder why you needed to explain the pixel model to her instead of formulating the question as "the same number of 0s and 1s in the string".)

   (d) **(Optional)** Give an ϵ-tester for your property and analyze it.

   Hint: Leave it for the graduate student.