CSC 2420 Fall 2017, Assignment 3
Due date: December 6, by 11:59pm

It is certainly preferable for you to solve the questions without consulting a published source. However, if you are using a published source then you must specify the source and you should try to improve upon the presentation of the result.

If you would like to discuss any questions with someone else that is fine BUT at the end of any collaboration you must spend at least one hour playing video games or watching two periods of Maple Leaf hockey or maybe even start reading a good novel before writing anything down.

If you do not know how to answer a question, state “I do not know how to answer this (sub) question” and you will receive 20% (i.e. 2 of 10 points) for doing so. You can receive partial credit for any reasonable attempt to answer a question BUT no credit or arguments that make no sense.

1. Recall the online bipartite matching problem from class, in which nodes on one side of a bipartite graph are present, nodes on the other side arrive online, and you must decide to match or not match them as they arrive. Show that any online algorithm A can be converted to a *greedy* online algorithm B with at least the same approximation ratio. Here, greedy means that B will always match an incoming vertex if one of its neighbours is still available.

2. Consider an input stream $A = a_1, \ldots, a_m$, where each $a_i \in [n]$. Define the rank of an element x as $rank(x) = |\{y \in A : y \leq x\}|$. The median of the stream is the element with rank $m/2$. We want to an ϵ-approximate median, i.e., an element x such that ${}^{m}\!/_{2} - \epsilon m \leq rank(x) \leq {}^{m}\!/_{2} + \epsilon m$.

   Suppose we use the most intuitive approach: select t random indices in $[m]$ in advance (with replacement), access the corresponding elements in the stream, and return their median. This uses $O(t \cdot (\log m + \log n))$-bit space. What value of t will be sufficient to ensure that the algorithm returns an ϵ-approximate median with probability at least $1 - \delta$?

   Hint: Partition the stream A into three disjoint sets: S_L contains elements of rank $< {}^{m}\!/_{2} - \epsilon m$, S_M contains ϵ-approximate medians, and S_H contains elements of rank $> {}^{m}\!/_{2} + \epsilon m$. Note that both S_L and S_H contain ${}^{1}\!/_{2} - \epsilon$ fraction of all elements in the stream, and for our approach to return a correct answer, we would like them to contain less than half of the sampled elements. Derive the required bound on t using the following Chernoff bound.

   Chernoff bound: If $X_1, \ldots, X_n$ are i.i.d. random variables taking values in $\{0, 1\}$, and $X = \sum_{i=1}^{n} X_i$, then for any $\gamma \in [0, 1]$,

   $$\Pr[X \geq (1 + \gamma)E[X]] \leq e^{-\frac{\gamma^2 E[X]}{3}}.$$

3. In this question, we want to design a one-sided property testing algorithm that checks whether an undirected input graph G is connected. Specifically, we want the algorithm to

   - always return 'Yes' if G is connected, and
   - return 'No' with probability at least ${}^{2}\!/_{3}$ if G is ϵ-far from being connected.

The graph is given in the *sparse* representation (i.e., adjacency list for every vertex). Let d denote the *average degree* in the graph. We say that a graph with m edges is ϵ-far from being connected if we must add at least ϵm edges to make it connected. Here is a proposed algorithm.

- Sample $\Theta({}^{1}\!/_{\epsilon d})$ vertices.
- Let $s = 1 + {}^{4}\!/_{\epsilon d}$. Start a BFS from each sampled vertex, and run it until it discovers at least s vertices.
- Return 'Yes' if every BFS discovers at least s vertices, and 'No' if even one BFS stops before discovering at least s vertices because it exhausted a connected component.

You will show that this algorithm satisfies our requirements.

(a) First, show that if a graph is ϵ-far from being connected, it has at least $\epsilon m + 1$ connected components.

(b) Next, show that if a graph is ϵ-far from being connected, then at least ${}^{\epsilon m}\!/_{2}$ connected components have size at most ${}^{4}\!/_{\epsilon d}$.

(c) Use the two results above to argue that the algorithm is indeed a valid connectivity tester.