ANLY550–Spring, 2017 Homework 4 Out: April 7, 2017
Due: April 25, 2017

1. Recall that a hash family $\mathcal{H}$ of hash functions mapping $[n] \to [r]$ is pairwise independent if for every 2 distinct values $x_1, x_2 \in [n] = \{1, \ldots, n\}$, and any $y_1, y_2 \in [r]$,

$$\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = 1/r^2.$$

   Also, recall that $\mathcal{H}$ is universal if for every pair of distinct values $x_1, x_2 \in [n]$,

$$\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = h(x_2)] \leq 1/r.$$

   Construct a specific family $\mathcal{H}$ that is universal, but not pairwise independent, and justify your answer. Write down the family as a table, with one column per input to the hash function (i.e., one column for each number in $[n]$), and one row per hash function. Try to make $|\mathcal{H}|$, n, and r as small as possible! (There is a way to accomplish this with all of these quantities equal to 2).

2. You are given a bag of n balls, with at least 10% of the balls being blue, at least 5% of the balls being yellow, and no more than 80% of the balls being red. Asymptotically, as n grows, how many balls do you have to draw at random from the bag to see a blue ball with probability at least $2/3$? (Assume that the balls are drawn with replacement, meaning that as soon as a ball is drawn, it is placed back in the bag before the next draw.)

3. Let A be a randomized algorithm for a decision problem (where the correct output is always YES or NO). Suppose that on YES instances the algorithm is always correct, but on NO instances the algorithm outputs YES with probability at most $1/2$. Derive a new algorithm B that is always correct on YES instances, and on NO instances outputs YES with probability at most 2^{-100}. How much slower is Algorithm B than Algorithm A?

4. Sometimes for special classes of inputs, NP-hard problems are solvable in polynomial time. For example, we have seen on an earlier problem set that computing the size of a maximal independent set in a tree can be done in polynomial time. Here, we will consider counting the number of independent sets in even more specialized graphs.

   Consider a graph that is a line on n vertices. (That is, the vertices are labelled 1 to n, and there is an edge from 1 to 2, 2 to 3, etc.) How many independent sets are there on a line graph?

   Similarly, describe how you could quickly compute the number of independent sets on a complete binary tree. (Here, just explain how to compute this number.) Calculate the number of independent sets on a complete binary tree with 127 nodes.

5. Recall that a clique in a graph G is a collection of mutually adjacent vertices. The decision version of the clique problem is to take a graph G and an integer k and decide if G has a clique of size k or not. The optimization version of the problem takes a graph G, and returns a largest clique in G. Show that if the decision problem has a polynomial time algorithm, then the optimization problem also has a polynomial time algorithm.

6. Consider the following problem, called 2Clique:

   INPUT: A undirected graph G and an integer k.

OUTPUT: 1 if G has two vertex disjoint cliques of size k, and 0 otherwise. (Two cliques are vertex disjoint if they do not share any vertices in common).

Show that this problem is NP-hard. Use the fact that the decision version of the clique problem in NP-complete.

7. We know that that all of NP-complete reduce to each other. It would be nice if this meant that an approximation for one NP-hard problem would lead to another. But this is not the case.

   Given a graph G, the Minimum Vertex Cover problem is to find the smallest set of vertices such that each edge of the graph is incident to at least one vertex of the set. Minimum Vertex Cover has a polynomial 2-approximation algorithm; that is, the algorithm always outputs a vertex cover whose size is within a factor of 2 of the optimal solution. The algorithm is to repeatedly choose an edge, add both of its endpoints into the cover, throw the vertices and its adjacent edges out of the graph, and continue. It is a 2-approximation algorithm because each edge that gets chosen during the course of the algorithm must have one of its endpoints in the cover; hence we have merely always thrown two vertices in where we might have gotten away with throwing in 1.

   We know (from the NP-completeness notes, which you may want to check) that C is a cover in a graph $G = (V,E)$ if and only if $V - C$ is an independent set in V. Explain why this does not yield an approximation algorithm that is within a constant factor of optimal for Maximum Independent Set. That is, show that for any constant c, there exists a graph for which even if we obtain a 2-approximation of the Minimum Vertex Cover, the corresponding independent set is not within a factor of c of the Maximum Independent Set.

   The Maximum Independent Set problem and the Maximum Clique problem are related in the following way: an independent set of a graph G is a clique in the complement of G. (The complement of G is the graph that contains exactly the edges that are not in G.) Does an approximation algorithm (that is, an algorithm within a constant factor of the optimal) for the Maximum Clique problem yield an approximation algorithm (within a constant factor of optimal) for Maximum Independent Set?