CSE 202: Design and Analysis of Algorithms Spring 2016

Problem Set 4

Instructor: Kamalika Chaudhuri **Due on:** Fri Jun 3

Instructions

- For your proofs, you may use any lower bound, algorithm or data structure from the text or in class, and their correctness and analysis, but please cite the result that you use.
- All problems are worth 10 points.
- If you do not prove that your algorithm is correct, we will assume that it is incorrect. If you do not provide an analysis of the running time, we will assume you do not know what the running time is.

Problem 1

Provide brief answers to the following questions. Detailed proofs are not necessary.

1. You are given an array A of n data elements, which come from k clusters. You don't know which element of A belongs to which cluster, but you do know that there are exactly n/k elements in A that belong to each cluster. Suppose you pick a random subset S of m elements from A with replacement. How large does m need to be so that with probability $1/2$, S includes at least one element from every cluster in A?
2. Consider the following variation of the coupon collector's problem. Each cereal box contains one of $2n$ different coupons. The coupons are in pairs – coupons 1 and 2 are a pair, 3 and 4 are a pair, and so on. You get a prize if you get at least one coupon from each pair. Assuming that the coupons in each box are chosen independently and uniformly at random from the $2n$ coupons, what is the expected number of cereal boxes you must buy before you get a prize?

Problem 2

Given an undirected unweighted graph, the goal of the longest path problem is to find the simple path in it with the maximum number of nodes. (Recall that a simple path is one that does not contain any node twice). In contrast with the shortest path problem which we looked at in class, the longest path problem is usually NP Hard. We will now look at a randomized algorithm for this problem.

1. Suppose you have a directed acyclic graph (DAG). First show that the longest path in a DAG can be found easily via dynamic programming in $O(m + n)$ time where $m = |E|$ is the number of edges and $n = |V|$ is the number of vertices in the graph. (It is ok to give the subtask and the recurrence and running time analysis and skip the proof of correctness in this case.)
2. Now, consider the following algorithm:
   (a) Let π be a random permutation of the vertices in the graph $G = (V, E)$, drawn uniformly at random from the set of all permutations of V.
   (b) Construct a DAG $G' = (V, E')$ from G as follows. For each edge $(u, v) \in E$, if $\pi(u) > \pi(v)$, then add (u, v) to E', else add (v, u) to E'.
   (c) Return the longest path in G'.

   If the longest path in G has length k, find a lower bound on the probability that this algorithm gives the correct path.

3. Suppose the length of the longest path $k = \frac{\log n}{\log \log n}$. How many times do you have to repeat the algorithm in part (2) so that your probability of success is $\geq \frac{1}{2}$?

Problem 3

Blood samples from n people are being tested for a disease. It can be costly to test each person separately, so we pool blood from a group of k people and test them. If the test is negative, we know noone in the pool have the disease. If the test is positive, all the k people in the pool have to be tested again individually to find out who has the disease, and thus $k + 1$ total tests are required for the k people in the pool. Suppose n is a multiple of k, and we create n/k disjoint pools of k people to test. Moreover, assume that each one of n people has the disease independently with probability p.

1. What is the probability that the test will be positive for a pooled sample of k people?
2. What is the expected number of tests needed?
3. Given n and p, write down an equation that describes the value of k so that the expected number of tests is minimized.
4. Notice that pooling is not always better than doing n individual tests. Give an inequality to show for what values of p, pooling is better than doing individual tests.