Fall 2014 Farach-Colton

CS 513: Exam #1

You may consult your 1 page crib sheet. Any theorem used in class or algorithm presented in class or homeowrk can be used. Other theorems or algorithms can only be used if you prove them. 25% credit for clearly stating "I don't know." -25% for copying out a proof or algorithm that we've already done in class (that is, don't waste your time copying an old proof or algorithm just to have filler. If we already have an algorithm, you can use it as a black box.). For all algorithmic questions, the faster the algorithm, the more points (assuming that you've got a correct algorithm!). And you need to analyze any algorithm you describe.

1. Word algorithms. Give the run time of each, assuming that all integers are have $\log n$ bits, and that that is the size of the word. (So XOR, +, time, AND, ... are all constant time.) If you decide to do some preprocessing, seperately analyze the running time of the preprocessing and the query.

   (a) Given an algorithm for picking out the ith bit of a number. That is, PickBit(x, i) returns 1 if the ith bit of x is 1, and 0 otherwise.

   (b) Give an algorithm for finding the least significant 1 in a word. That is, LS1(x) returns i, if the PickBit$(x, i) = 1$ and, for all $j < i$, PickBit$(x, j) = 0$.

   (c) Given two words, x and y, give an algorithm to compute $\vee_{i=1}^{\log n} x_i \wedge y_i$, where $x_i =$ PickBit(x, i), and similarly for y_i.

2. The Selection Problem $SP(A, k)$ is the problem of finding the element in array A with rank k, that is, the element such that $k - 1$ elements are less that it (you may assume that all elements are distinct). The Quantile Problem $QP(A, c)$ is that of finding elements in A with rank $n/c, 2n/c, \ldots, cn/c$ (the last one is just the max, of course).

   (a) Give an algorithm for solving the Quantile Problem.

   (b) When is your algorithm faster than sorting?

3. Consider the Tree Meet Problem, defined as follows:

   Preprocess: An unrooted tree T.

   Query: TM(u, v, w) returns the unique node shared by the shortest uv, uw and vw paths.

   Give an algorithm for the Tree Meet Problem with constant query time. Give one with linear preprocessing time. In either case, minimize the running time of the other operation. Note: when you preprocess T, you don't know u, v or w (since you have to deal with lots of queries for lots of particular choices of u, v and w!).

4. A *k-clique* in a graph G is a set of k nodes, each pair of which has an edge between them. A *k-independent set* in a graph G is a set of k nodes, no pair of which has an edge between them.

   Consider the following problems:

   The k-Clique Problem:

   Input: A graph G and an integer k.

   Output: Yes, if G has a k-clique, No otherwise.

   The k-IS Problem:

   Input: A graph G and an integer k.

   Output: Yes, if G has a k-independent set, No otherwise.

   (a) Show that if the k-Clique problem has a polynomial time algorithm, so does the k-IS problem.

   (b) Show that if the k-Clique problem has a polynomial time algorithm, then you can actually find a largest clique (that is, give the list of node which form a clique of maximum size) in polynomial time.

   (c) Show that the 5-clique problem has a polynomial time solution.

Good Luck