CS 573: Algorithms, Fall 2013

Homework 2, due Monday, October 7, 23:59:59, 2013

Version 1.1

Neatly print your name(s), NetID(s) on each submitted question. Remember that you have to submit each question on a separate page. each student should submit his own homework. If you are on campus, submit the homework by submitting it in the homework boxes in the basement of SC.

To acknowledge the corn – This purely American expression means to admit the losing of an argument, especially in regard to a detail; to retract; to admit defeat. It is over a hundred years old. Andrew Stewart, a member of Congress, is said to have mentioned it in a speech in 1828. He said that haystacks and cornfields were sent by Indiana, Ohio and Kentucky to Philadelphia and New York. Charles A. Wickliffe, a member from Kentucky questioned the statement by commenting that haystacks and cornfields could not walk. Stewart then pointed out that he did not mean literal haystacks and cornfields, but the horses, mules, and hogs for which the hay and corn were raised. Wickliffe then rose to his feet, and said, "Mr. Speaker, I acknowledge the corn".
– Funk, Earle. "A Hog on Ice and Other Curious Expressions

Required Problems

1. Greedy algorithm do not work. (40 pts.)
A natural algorithm, **GreedyIndep**, for computing maximum independent set in a graph, is to repeatedly remove the vertex of lowest degree in the graph, and add it to the independent set, and remove all its neighbors.

(A) (5 pts.) Show an example, where this algorithm fails to output the optimal solution.

(B) (5 pts.) Let G be a $(k, k+1)$-uniform graph (this is a graph where every vertex has degree either k or $k+1$). Show that the above algorithm outputs an independent set of size $\Omega(n/k)$, where n is the number of vertices in G.

(C) (5 pts.) Let G be a graph with average degree δ (i.e., $\delta = 2\left|E(G)\right| / \left|V(G)\right|$). Prove that the above algorithm outputs an independent set of size $\Omega(n/\delta)$.

(D) (5 pts.) For any integer k, present an example of a graph G_k, such that **GreedyIndep** outputs an independent set of size $\leq \left|OPT(G_k)\right|/k$, where $OPT(G_k)$ is the largest independent set in G_k. How many vertices and edges does G_k has? What it the average degree of G_k?

(E) (20 pts.) Coloring. Let G be a graph defined over n vertices, and let the vertices be ordered: $v_1, \ldots, v_n$. Let G_i be the induced subgraph of G on $v_1, \ldots, v_i$. Formally, $G_i = (V_i, E_i)$, where $V_i = \{v_1, \ldots, v_i\}$ and

$$E_i = \left\{ uv \in E \;\middle|\; u, v \in V_i \text{ and } uv \in E(G) \right\}.$$

The greedy coloring algorithm, colors the vertices, one by one, according to their ordering. Let k_i denote the number of colors the algorithm uses to color the first i vertices.
In the ith iteration, the algorithm considers v_i in the graph G_i. If all the neighbors of v_i in G_i are using all the k_{i-1} colors used to color G_{i-1}, the algorithm introduces a new color (i.e., $k_i = k_{i-1} + 1$) and assigns it to v_i. Otherwise, it assign v_i one of the colors $1, \ldots, k_{i-1}$ (i.e., $k_i = k_{i-1}$).
Give an example of a graph G with n vertices, and an ordering of its vertices, such that even if G can be colored using $O(1)$ (in fact, it is possible to do this with two) colors, the greedy algorithm would color it with $\Omega(n)$ colors. (Hint: consider an ordering where the first two vertices are not connected.)

2. Find kth smallest number. (20 pts.)
This question asks you to design and analyze a *randomized incremental* algorithm to select the kth smallest element from a given set of n elements (from a universe with a linear order).
In an *incremental* algorithm, the input consists of a sequence of elements $x_1, x_2, \ldots, x_n$. After any prefix $x_1, \ldots, x_{i-1}$ has been considered, the algorithm has computed the kth smallest element in $x_1, \ldots, x_{i-1}$

(which is undefined if $i \le k$), or if appropriate, some other invariant from which the kth smallest element could be determined. This invariant is updated as the next element x_i is considered.

Any incremental algorithm can be *randomized* by first randomly permuting the input sequence, with each permutation equally likely.

(A) (5 PTS.) Describe an incremental algorithm for computing the kth smallest element.

(B) (5 PTS.) How many comparisons does your algorithm perform in the worst case?

(C) (10 PTS.) What is the expected number (over all permutations) of comparisons performed by the randomized version of your algorithm? (Hint: When considering x_i, what is the probability that x_i is smaller than the kth smallest so far?) You should aim for a bound of at most $n + O(k \log(n/k))$. Revise (A) if necessary in order to achieve this.

Most of the points (i.e., 8) would be given to a solution that achieves $n + O(k \log^2 n)$ expected number of comparisons. The better bound is quite challenging.

3. Strange tree (40 PTS.)

Consider a uniform rooted tree of height h (every leaf is at distance h from the root). The root, as well as any internal node, has 5 children (as such, the tree has $n = 5^h$ leafs). Each leaf has a boolean value associated with it. Each internal node returns the value returned by a boolean function $f(b)$, where $b = (b_1, \ldots, b_5)$ are the values returned by its children. Assume that f has the property that for any binary vector $b \in \{0,1\}^5$, there exists a vector $b' \in \{0,1\}^5$, that differs only in a single bit, such that $f(b) = f(b')$. For example, if f returns the majority vote of its five input bits, then it has it property. For example, in this case, we have $f(1,0,1,0,1) = f(1,1,1,0,1) = 1$.

The evaluation problem consists of determining the value of the root; at each step, an algorithm can choose one leaf whose value it wishes to read.

You can assume f is given to you, and you know its value for every possible input.

(A) (5 PTS.) Prove that indeed the majority function has the desired property. Specifically, $f(b_1, \ldots, b_5)$ returns one, if and only if at least 3 bits of $b_1, \ldots, b_5$ are one.

(B) (35 PTS.) Present a recursive randomized algorithm that computes the value of a node. Prove that the expected number of leaves read by your algorithm on any instance is at most n^c, where c is some constant smaller than 1. Provide a precise bound on the value of c.

(Hint: (A) Think about the case where the tree has only a single internal node. Come up with a randomized algorithm that in expectation does not read all 5 leafs. (B) Try to solve this first for case that f is indeed the majority function.)

Important: The input is not random! The randomness is only made by the choices of the algorithm. In particular, your algorithm should have expected sublinear running time for any input.