HW 6 (due Monday, at noon, March 11, 2013)

CS 473: Fundamental Algorithms, Spring 2013 Version: **1.1**

Make sure that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: The homework can be worked in groups of up to 3 students each.

1. (30 PTS.) The climbing vampires of Champaign.

Every year the vampires of Champaign elect their leader using a somewhat bizarre process. They all go and travel to Peru and try to climb Siula Grande[1]. The rule of the game is that they all start from the base camp and climb for 5 hours, and then they stop (it is a competition, so each vampire probably reached a different location in the mountain). After that, the base station start calling the vampires one after the other, and ask them for their location, and more significantly how high are they in the mountain (the vampires have each a unique ID between 1 and n which they can use to call them). The vampire that is highest on the mountain is the new leader.

To avoid a vacuum in leadership, the leader is being updated as the results come in. Specifically, if after contacting the first i vampires the last vampire contacted is the highest, then the base station sends n messages to all the vampires telling them that this vampire is the new leader, where n is the number of vampires participating.

(A) (10 PTS.) Given that the given vampires are $v_1, \ldots, v_n$, describe a strategy that minimizes the overall number of messages sent. How many messages does you scheme sends in the worst case? If you decide on a randomized strategy, how many messages does your scheme sends in expectation? The smaller the number of messages, the better. Prove your answer.

(B) (10 PTS.) How many times does the leader changes in the worst case (and also in expectation if your scheme is randomized) under your scheme (the fewer the better). Prove your answer.

(C) (10 PTS.) A new scheme was suggested that minimizes the number of messages sent: Whenever a new leader is discovered, you announce it only to the vampires already contacted. Describe a scheme that orders the vampires, such that the number of messages sent is minimized. What is the number of messages sent by your scheme (either in the worst case, and in expectation if your scheme is randomized). The fewer the better.

For this question, if you have two bounds x and y, then if $x < y$ then x is preferable to y, even if x holds only in expectation, and y is a worst case bound.

2. (40 PTS.) Collapse and shrink.

(A) (5 PTS.) Consider the procedure that receives as input an undirected weighted graph G with n vertices and m edges, and weight x, and outputs the graph $\mathsf{G}_{<x}$ that results from removing all the edges in G that have weight larger than (or equal to) x. Describe (shortly – no need for pseudo code) an algorithm for computing $\mathsf{G}_{<x}$. How fast is your algorithm?

The graph $\mathsf{G}_{<x}$ might not be connected – how would you compute its connected components?

[1]See http://en.wikipedia.org/wiki/Touching_the_Void_%28film%29.

(B) (5 PTS.) Consider the procedure that receives as input an undirected weighted graph G, and a partition $\mathcal{V}$ of the vertices of G into k disjoint sets $V_1, \ldots, V_k$. The ***meta graph*** $\mathsf{G}(\mathcal{V})$ of G induces by $\mathcal{V}$ is a graph having k vertices, $v_1, \ldots, v_k$, where $v_i v_j$ has an edge if and only if, there is an edge between some vertex of V_i and some vertex of V_j. The weight of such an edge $v_i v_j$ is the minimum weight of any edge between vertices in V_i and vertices in V_j.

Describe an algorithm, as fast as possible, for computing the meta-graph $\mathsf{G}(\mathcal{V})$. You are not allowed to use hashing for this question, but you can use that **RadixSort** works in linear time (see wikipedia if you do not know **RadixSort**). How fast is your algorithm?

(C) (10 PTS.) Consider the randomized algorithm that starts with a graph G with m edges and n vertices. Initially it sets $\mathsf{G}_0 = \mathsf{G}$. In the ith iteration, it checks if G_{i-1} is a single edge. If so, it stops and outputs the weight of this edge. Otherwise, it randomly choose an edge $e_i \in \mathsf{E}(\mathsf{G}_{i-1})$. It then computes the graph $H_i = (\mathsf{G}_{i-1})_{<w(e_i)}$, as described above.

- If the graph H_i is connected then it sets $\mathsf{G}_i = H_i$ and continues to the next iteration.
- Otherwise, H_i is not connected, then it computes the connected components of H_i, and their partition $\mathcal{V}_i$ of the vertices of G_{i-1} (the vertices of each connected component are a set in this partition). Next, it sets G_i to be the meta-graph $\mathsf{G}_{i-1}(\mathcal{V}_i)$.

Let m_i be the number of edges of the graph G_i. Prove that if you know the value of m_{i-1}, then $\mathbf{E}[m_i] \leq (7/8)m_{i-1}$ (a better constant is probably true). Conclude that $\mathbf{E}[m_i] \leq (7/8)^i m$.

(D) (15 PTS.) What is the expected running time of the algorithm describe above? **Prove** your answer. (The better your bound is, the better it is.)

(E) (5 PTS.) What does the above algorithm computes, as far as the original graph G is concerned?

3. (30 PTS.) Selection revisited.

You are given two arrays of numbers $X[1 \ldots n]$ and $Y[1 \ldots m]$, where n is smaller than m.

(A) (20 PTS.) Given a number k, and assuming both X and Y are sorted (say in increasing order), describe an algorithm, as fast as possible, for finding the k smallest number in the set $X \cup Y$ (assume all the numbers in X and Y are distinct).

(B) (10 PTS.) Solve the same problem for the case that X is not sorted, but Y is sorted.