CSE P521: Applied Algorithms (Winter, 2017)

Homework 1 Out: Thurs, 5-Jan. **Due**: Thurs, 12-Jan. (midnight in the **Dropbox**)

Instructions:

Your proofs and explanations should be clear, well-organized and as concise as possible.

You are allowed to discuss the problems with fellow students taking the class. However, you must write up your solutions completely on your own. Moreover, if you do discuss the problems with someone else, I am asking, on your honor, that you do not take any written material away from the discussion. In addition, for each problem on the homework, I ask that you acknowledge the people you discussed that problem with, if any.

Most of the problems require only one or two key ideas for their solution – spelling out these ideas should give you most of the credit for the problem even if you err in some finer details. So, make sure you clearly write down the main idea(s) behind your solution.

A final piece of advice: Begin work on the problem set early and don't wait until the deadline is only a few days away.

1. Maintaining a uniformly random element from a stream

Suppose you see a stream of items arriving one at a time: $a_1, a_2, a_3, \ldots$ You want to maintain a uniformly random item from the sequence, but you only have memory enough to store one item. Give an algorithm to solve the problem and prove that it's correct.

2. Min-cut modification

In lecture, we saw Karger's global min-cut algorithm and argued that given any n-vertex graph, it outputs a min-cut with probability at least $\frac{2}{n(n-1)}$. Consider the following variant: Instead of choosing an edge to contract uniformly at random, we choose two uniformly random vertices and merge them into a single vertex. Prove or disprove the assertion:

> This algorithm outputs a min-cut with probability at least n^{-c} for some constant $c > 0$.

3. The second-strongest Pokemon

Suppose you have n Pokemon $P_1, P_2, \ldots, P_n$ and you want to find the second-strongest. Since you don't know very much about Pokemon, the only way to compare is to have them battle each other. If P_i and P_j battle, you can see the outcome.

Your friend suggests that you should have all $\frac{n(n-1)}{2}$ pairs of Pokemon battle each other and then you can rank them. You tell your friend to stop being silly; you can sort them using only $O(n \log n)$ battles and then choose the second best.

Consider this randomized strategy: You choose a uniformly random index $t \in \{1,2, \ldots, n\}$, and you have Pokemon P_t battle the other $n-1$ Pokemon. Using the outcome, you put them in two sets L and W (the losers and winners). If there is exactly one winner, you return P_t as the second-strongest. If W is empty, you discard P_t and look for the **strongest** Pokemon among the remaining $n-1$. If $|W| > 2$, you recurse on W.

(a) Prove that this algorithm always returns the second-strongest Pokemon
(b) Show carefully that the **expected** number of battles needed to find the second-best is $O(n)$