CS41 Lab 11: Randomized and Approximation Algorithms

Thursday, November 17

1. **Finding the Median.** Given a set $S = \{a_1, \ldots, a_n\}$ of numbers, the *median* of S, denoted $\mathrm{med}(S)$, is the k-th smallest element of S, where $k = \lfloor \frac{n+1}{2} \rfloor$. In this problem, you will analyze a randomized algorithm to output the median. Consider the following algorithm for finding the median:

   FindMedian(S)
   Return Select($S, \lfloor \frac{n+1}{2} \rfloor$)

   Select(S, k)
   Choose pivot $a_i \in S$
   Initialize $S^-, S^+ := \{\}$
   for each $j \neq i$
       if $a_j < a_i$ add a_j to S^-
       if $a_j > a_i$ add a_i to S^+
   if $|S^-| = k - 1$ **return** a_i
   else if $|S^-| > k - 1$
       return Select(S^-, k)
   else
       return Select($S^+, k - (1 + |S^-|)$)

   - First, show that FindMedian always returns the median.
   - Next, analyze the running time of FindMedian when the pivot element is chosen uniformly from S[1]. The following structure will help guide you. Say that the algorithm is in *phase* j if there are between $n(3/4)^j$ and $n(3/4)^{j+1}$ elements in the set S being considered. So, for example, we are in phase 0 the first time Select is called.

     Call an element $a_i \in S$ *central* to S if (i) at least $|S|/4$ of the elements of S are less than a_i and (ii) at least $|S|/4$ elements of S are greater than a_i.

     (a) Show that there are $|S|/2$ central elements.
     (b) Show that if the pivot element is central, the phase ends i.e., the next recursive call that gets made will be in a different phase.
     (c) Give an upper bound on the expected number of recursive calls to Select before a round ends.
     (d) Give an upper bound on the running time of each Select call, not including recursive calls.
     (e) Give an upper bound on the number of phases that are run before FindMedian terminates.
     (f) Give an upper bound on the expected runtime of FindMedian when the pivot is chosen uniformly.

[1] An element is chosen *uniformly* if each element is equally likely to be picked.

2. **Three-Coloring Revisited.** Recall the THREE-COLORING problem: Given a graph $G = (V, E)$, output YES iff the vertices in G can be colored using only three colors such that the endpoints of any edge have different colors. In homework 8, you showed that THREE-COLORING is NP-COMPLETE. In this lab, we'll look at several approximation and randomized algorithms for the optimization version of THREE-COLORING.

   Let THREE-COLOR-OPT be the following problem. Given a graph $G = (V, E)$ as input, color the vertices in G using at most three colors in a way that maximizes the number of *satisfied* edges, where an edge $e = (u, v)$ is satisfied if u and v have different colors.

   Hardness of Three-Color-OPT. Show that if there is a polynomial-time algorithm for THREE-COLOR-OPT then P = NP.

3. **Approximation Algorithm.** Give a deterministic, polynomial-time (3/2)-approximation algorithm for THREE-COLOR-OPT. Your algorithm must satisfy at least $2c^*/3$ edges, where for an arbitrary input $G = (V, E)$, c^* denotes the maximum number of satisfiable edges.

4. **Randomized Algorithms.** Give randomized algorithms for THREE-COLOR-OPT with the following behavior:

   (a) An algorithm with expected polynomial runtime that always outputs a three-coloring that satisfies at least $2c^*/3$ edges.

   (b) An algorithm that runs in worst-case (i.e., not expected) polynomial time and produces a three-coloring such that the expected number of satisfied edges is at least $2c^*/3$.

   (c) An algorithm that runs in worst-case polynomial time, and with probability at least 99% outputs a three-coloring which satisfies at least $2c^*/3$ edges. What is the running time of your algorithm? The following inequality might be helpful: $1 - x \leq e^{-x}$ for any $x > 0$.