CS41 Lab 1

September 7, 2020

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students each. You will not be handing in solutions; the primary purposes of these labs are to have a low-pressure space to discuss algorithm design, and to gain experience collaborating with others on algorithm design and analysis.
However, it will be common to have some overlap between lab exercises and homework sets.

1. **Lower Bound for the Hiking Problem** Show that *any solution* to the Hiking Problem must take at least m miles.

2. In lecture earlier today we discussed the following algorithm (Algorithm #2) for the Hiking Problem:

   ```
   Hiking()
   1  $k = 1$.
   2  while you haven't arrived at your friend:
   3      hike $k$ miles north
   4      return to start
   5      hike $k$ miles south
   6      return to start
   7      $k = k + 1$.
   ```

   We quickly went over analysis showing that in the worst case, we travelled $O(m^2)$ miles if your friend is m miles away.

   The purpose of this exercise is to practice/formalize the analysis for the distance traveled in Hiking in the worst case, similar to the second algorithm we saw in lecture.

   (a) First, assume that your friend is m miles north of you. How many iterations of the loop do you execute before reaching your friend?
   (b) For any loop iteration *except* the final loop iteration, how far do you hike during this iteration? (express your answer in terms of k)
   (c) How far do you hike during the final loop iteration?
   (d) Now, calculate the total distance traveled by adding up the distance hiked over all loop iterations.
   (e) Next, repeat the steps above for the case where your friend is m miles *south* of you.
   (f) In the worst case, how far do you travel if your friend is m miles away?

3. Is Algorithm #2 the most efficient algorithm you can do for the Hiking problem? Either argue why this is the most effecient, or try to understand **why** this algorithm is not efficient, and design and analyze a more efficient algorithm.

4. **Counterfeit Coins.** You are given n coins and a balance scale. To use this scale, you put a number of coins in a pile on the left part of the scale, and a number of coins in a pile on the right. The scale indicates which pile is heavier.

   Most of the coins are identical in every aspect; however, one of the coins is counterfeit and much heavier than the rest. Design an algorithm to identify the counterfeit coin that uses the scale at most $\log_3 n$ times.