Problem Set 4

Data Structures and Algorithms, Fall 2022

Due: October 6, in class.

Problem 1

In this problem we will develop a divide-and-conquer algorithm for the following geometric task.

The Closest Pair Problem:
Input: A set of n points in the plane, $\{p_1 = (x_1, y_1), p2 = (x_2, y_2), \cdots, p_n = (x_n, y_n)\}$.
Output: The closest pair of points. That is, the pair $p_i \neq p_j$ that minimizes $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

For simplicity, assume that n is a power of two, and that all the x-coordinates x_i are distinct, as are the y-coordinates. Here's a high-level overview of the algorithm:

- Find a value x for which exactly half the points have $x_i < x$, and half have $x_i > x$. On this basis, split the points into two groups, L and R.
- Recursively find the closest pair in L and in R. Say these pairs are $p_L, q_L \in L$ and $p_R, q_R \in R$, with distances d_L and d_R respectively. Let d be the smaller of these two distances.
- It remains to be seen whether there is a point in L and a point in R that are less than distance d apart from each other. To this end, discard all points with $x_i < x - d$ or $x_i > x + d$ and sort the remaining points by y-coordinate.
- Now, go through this sorted list, and for each point, compute its distance to the *seven* subsequent points in the list. Let p_M, q_M be the closest pair found in this way.
- The answer is one of the three pairs $\{p_L, q_L\}, \{p_R, q_R\}, \{p_M, q_M\}$, whichever is closest.

(a) Prove that the algorithm is correct. *(Hint: You may find the following property helpful: any square of size $d \times d$ in the plane contains at most four points of L.)*

(b) Write down the pseudocode for the algorithm and argue its running time is $O(n \log^2 n)$.

(c) Can you improve the algorithm so that its running time is reduced to $O(n \log n)$?

Problem 2

Given a list I of n intervals, specified as (x_i, y_i) pairs, return a list where the overlapping intervals are merged. For example, for $I = \{(1, 3), (2, 6), (8, 10), (7, 18)\}$ the output should be $\{(1, 6), (7, 18)\}$. Your algorithm should have $O(n \log n)$ time complexity.

Problem 3

Suppose you are given a stack of n pancakes of different sizes. You want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation you can perform is a *flip*—insert a spatula under the top k pancakes, for some integer $k \in [1, n]$, and flip them all over. (See Figure 1.)

(a) Describe an algorithm to sort an arbitrary stack of n pancakes using $O(n)$ flips.

(b) Describe a stack of n pancakes that requires $\Omega(n)$ flips to sort.

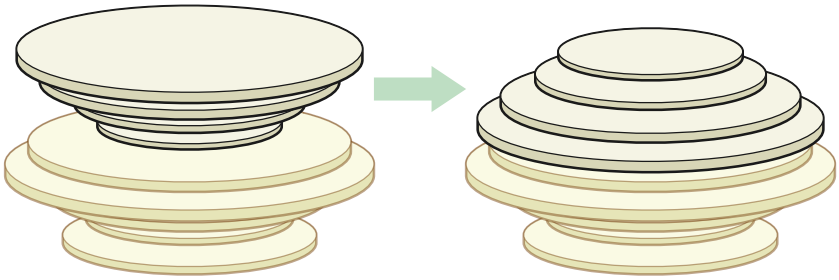

Figure 1

Problem 4

`StrangeSort`$(A[0, \cdots, n-1])$

1: **if** $(n == 2$ **and** $A[0] > A[1])$ **then**
2: `Swap`$(A[0], A[1])$.
3: **else if** $(n > 2)$ **then**
4: $m \leftarrow \lceil 2n/3 \rceil$.
5: `StrangeSort`$(A[0, \cdots, m-1])$.
6: `StrangeSort`$(A[n-m, \cdots, n-1])$.
7: `StrangeSort`$(A[0, \cdots, m-1])$. ▷ Yes, this line is identical to Line 5.

(a) What is the running time of the algorithm? Prove your answer.

(b) Prove that the algorithm actually sorts its input.

(c) Is the algorithm still correct if we replace $m \leftarrow \lceil 2n/3 \rceil$ with $m \leftarrow \lfloor 2n/3 \rfloor$. Prove your answer.

(d) Prove that the number of swaps executed by the algorithm is at most $\binom{n}{2}$.

Problem 5

One way to improve randomized quicksort is to partition around a pivot that is chosen more carefully than by picking a random element from the subarray. One common approach is the median-of-3 method: choose the pivot as the median of a set of 3 elements randomly selected from the subarray. For this problem, let us assume that the elements in the input array $A[1, \cdots, n]$ are distinct and that $n \geq 3$. We denote the sorted output array by $A' = [1, \cdots, n]$. Using the median-of-3 method to choose the pivot element x, define $p_i = \Pr(x = A'[i])$.

(a) Give an exact formula for p_i as a function of n and i for $i = 2, 3, \cdots, n-1$. (Note that $p_1 = p_n = 0$.)

(b) By what amount have we increased the likelihood of choosing the pivot as $A'[\lfloor (n+1) \rfloor / 2]$, the median of $A[1, \cdots, n]$, compared with the ordinary implementation? Assume that $n \to \infty$, and give the limiting ratio of these probabilities.

(c) If we define a "good" split to mean choosing the pivot as $x = A'[i]$, where $n/3 \leq i \leq 2n/3$, by what amount have we increased the likelihood of getting a good split compared with the ordinary implementation? *(Hint: Approximate the sum by an integral.)*

(d) Does the median-of-3 method reduce the (asymptotic) best-case or worst-case running time of randomized quicksort? If your answer is negative, discuss why the method could be considered as an "improvement" for randomized quicksort.

Problem 6 [OJ Problem]

(Solve this problem on the OJ platform, do not hand in written solutions!)

Suppose you are given two sets of n points, one set $\{p_1, p_2, \cdots, p_n\}$ on the line $y = 0$ and the other set $\{q_1, q_2, \cdots, q_n\}$ on the line $y = 1$. Create a set of n line segments by connect each point p_i to the corresponding point q_i. Devise an efficient divide-and-conquer algorithm to determine how many pairs of these line segments intersect. (For example, see Figure 2.) *(Hint: Modify* `MergeSort`*.)*

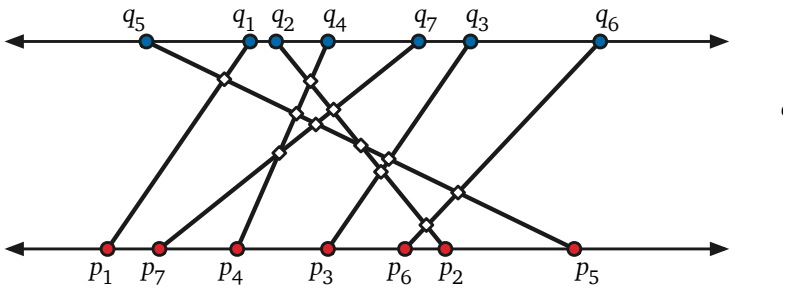

Figure 2