CS302 - Problem Set 3

1. Read pages 2-5, 9-10 of the pdf of Chapter 3 of Weapons of Math Destruction by Cathy O'Neil on US News and World Report's College Ranking algorithm. (I encourage you to read the rest of the chapter if you are interested, but the key issues are in the pages listed.) This is a real world (and I suspect highly relevant) example of a fairly simple algorithm where weighting choices have large ethical consequences. Consider:

   (a) Brainstorm all stake-holders.

   (b) Who might benefit from this algorithm (applied to this domain)?

   (c) Who might be harmed by this algorithm (applied to this domain)?

   (d) Would this application likely reinforce or counteract existing inequalities?

   (e) **(Slightly Different from usual!!)** Are there ways that you could improve the weighting and create a better (ethically) ranking algorithm? Would any school ranking algorithm have negative ethical consequences?

   (f) What do you think of the new College Scorecard?

2. I've updated the 3D algorithm from the last problem set to have an input that is presorted, similar to what we did with the 2D algorithm. (See pseudocode below).

   (a) What is the runtime of this 3D closest points algorithm?

   (b) Comment on the runtimes of the 2D and 3D closest points algorithm. If you had to guess, what do you think the runtime is in d-dimensional space?

   (c) **(Challenge)** Create an algorithm and analyze the runtime for the d-dimensional closest points problem.)

Algorithm 1: `PreSort`(P)

Input : P, an unordered list of n points
Output: 3 copies of the points in P, sorted by x-, y-, and z-coordinate.

```
1 X ← Sort P by x ;
2 Y ←Sort P by y ;
3 Z ←Sort P by z ;
4 return {X, Y, Z};
```

Algorithm 2: `DivideFrontBack`(X, Y, Z)

Input : X, Y, and Z: Lists of n points sorted by x-, y-, and z-coordinate respectively
Output: The distance of the closest pair of points

1 If $|X| \leq 3$, brute force search;
2 Split points into front and back halves by z-coordinate around the midline z_{mid}, to get X_F, X_B, Y_F, Y_B, Z_F, and Z_B;
3 $\delta^* = \min\{$`DivideFrontBack`(X_F, Y_F, Z_F),`DivideFrontBack`$(X_B, Y_B, Z_B)\}$;
4 Create X_{δ^*}, Y_{δ^*} and Z_{δ^*}, which are sorted arrays of points whose z-coordinates are within δ^* of z_{mid}.;
5 Return `DivideLeftRight`$(X_{\delta^*}, Y_{\delta^*}, Z_{\delta^*}, \delta^*, z_{mid})$;

Algorithm 3: `DivideLeftRight`$(X, Y, Z, \delta^*, z_{mid})$

Input : X, Y, and Z: Lists of n points sorted by x-, y-, and z-coordinate respectively, where all n points have z-ccordinates within δ^* of z_{mid}.
Output: The distance of the closest pair of points

1 If $|X| \leq 3$, brute force search;
2 Split points into left and right halves by x-coordinate around the midline x_{mid}, to get X_L, X_R, Y_L, Y_R, Z_L, and Z_R;
3 $\delta =$ $\min\{\delta^*,$`DivideLeftRight`$(X_L, Y_L, Z_L, \delta^*, z_{mid})$,`DivideLeftRight`$(X_R, Y_R, Z_R, \delta^*, z_{mid})\}$;
4 Let Y_δ be the set of points sorted by y-coordinate whose x coordinate is within δ of x_{mid} or whose z coordinate is within δ of z_{mid};
5 Loop through the elements of Y_δ, checking the distance between each point and the next ?? points, and let δ' be the smallest distance found;
6 Return $\min\{\delta, \delta'\}$;

3. Suppose you have n events, where the ith event has a start time s_i and end time f_i, for $i \in \{1, \ldots, n\}$. Unfortunately, you only have one auditorium, and you can't schedule conflicting events (events where a start time of one is between the start time and end time of another.) We would like to maximize the number of events that are held.

   (a) Run through our ethical analysis questions. (If we practice this enough, you will hopefully remember to do this when you are putting algorithms out in the world and it really matters!)

   (b) For each of the following greedy algorithms, create an example of a series of events (with start and end times) where the algorithm does not perform optimally.

      i. At each iteration, pick the remaining event with the earliest start time.
      ii. At each iteration, pick the remaining event that has the shortest time ($f_i - s_i$ is smallest.)

4. Let $T(n)$ be the number of bit strings of length n where there are two consecutive 1s. Create a recurrence relation for $T(n)$.

   This goal of this problem is to remind you how to think recursively - a skill from 200 that will be crucial for our next paradigm, Dynamic Programming. I know it has been a while for some of you, so I've made a video about how to solve these types of problems. You don't have to watch, but it is available if you need it: Panopto

5. Friendly reminder to start working on your 1st programming assignment. The rough draft is due the same day as this pset.