CS302 - Problem Set 2

Due: Monday, Sep. 25. Must be uploaded to Canvas before the beginning of class.

Please be familiar with the sections of the syllabus on problem sets and honor code before starting this homework. You may also want to look at the grading rubrics.

1. *Proof Practice*

   (a) **[11 points]** Prove that Algorithm 1 multiplies a non-negative n and an integer b.

   Algorithm 1: `Mult`(n, b)

   Input : Non-negative integer n, and integer b
   Output: $n \times b$

```
   /* Base Case                                                    */
1  if n == 0 then
2  |  return 0
3  |
4  else
   |  // Recursive step
5  |  return b + Mult(n − 1, b)
6  end
```

   (b) **[11 points]** Prove Algorithm 2 for binary search is correct.

   Algorithm 2: `BinarySearch`(A, V, s, f)

   Input : Sorted increasing array A, a value V, a starting index s, and a final index f
   Output: Index i such that $A[i] = V$, and $s \leq i \leq f$ or 0 if no V exists in A with index betwen s and f inclusive.

```
   /* Base Case                                                    */
1  if f − s == 0 then
2  |  if A[s] == V then
3  |  |  return s
4  |  else
5  |  |  return 0;
6  |  end
7  else
   |  // Recursive step
8  |  mid = ⌊(f + s)/2⌋
9  |  if A[mid] ≤ V then
10 |  |  return BinarySearch(A, V, mid, f)
11 |  else
12 |  |  return BinarySearch(A, V, s, mid − 1)
13 |  end
14 end
```

2. *3-D Closest Points*

In this problem, you will design an algorithm for the closest points problem in 3D. That is, suppose you are given an array of n points, and for each point p_i, we are given values (x_i, y_i, z_i) which are its x- y- and z-coordinates. The distance between two points is given by $d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$. We would like to find the distance between the pair of closest points.

In the next few paragraphs and questions, I will lead you through one approach to this problem. However, I encourage you to take some time to try to come up with some strategies on your own first.

Let `CP_3D`(X, Y, Z) be an algorithm that takes as input 3 arrays X, Y, Z, each of which contains the same n points $P = \{p_1, \ldots, p_n\}$, but sorted by x-, y-, and z-coordinates respectively. `CP_3D`(X, Y, Z) outputs the distance between the closest pair of points. *Note: the output is not the pair of points, but the distance!*

Let `CP_Almost2D`(X, Y, Z, δ) be an algorithm that takes as input a real positive number δ, and 3 arrays X, Y, Z, each of which contains the same n points $P = \{p_1, \ldots, p_n\}$, but sorted by x-, y-, and z-coordinates respectively. P is assumed to have the property that all points have z-coordinate values between $\bar{z} - \delta$ and $\bar{z} + \delta$, for some real number $\bar{z}$. (That is, they are all in a narrow region within distance δ of the plane at $z = \bar{z}$.) Furthermore, if two points p_i and p_j both have z-coordinates on the same side of $\bar{z}$, then $d(p_i, p_j) \geq \delta$. `CP_Almost2D`(X, Y, Z, δ) returns the distance between the closest points in P.

You may assume that all x- y- and z-coordinates for all points are distinct.

(a) **[9 points]** Write psuedocode for an algorithm for `CP_3D` that uses `CP_Almost2D` as a subroutine. (You do not need to write psuedocode for `CP_Almost2D`.)

(b) **[11 points]** Consider the pseudocode in algorithm 3 for `CP_Almost2D`. Prove this algo-

rithm is correct.

Algorithm 3: `CP_Almost2D`(X, Y, Z, δ)

Input : 3 arrays X, Y, Z, each of which contains the same n points $P = \{p_1, \ldots, p_n\}$, but sorted by x-, y-, and z-coordinates respectively. Real number $\delta > 0$, such that P satisfy the conditions relative to δ laid out above.

Output: The distance between the closest pair of points in P.

```
/* Base Case                                                    */
if |P| ≤ 3 then
    minDist = ∞
    for p_i, p_j ∈ P, i ≠ j do
        if d(p_i, p_j) < minDist then
            minDist = d(p_i, p_j)
        end
    end
    return minDist
else
    // Divide and Conquer
    Let x̄ divide the points in half by x-coordinate
    Create Z_L, X_L, Y_L, (sorted arrays containing the "left" half, those with x_i < x̄)
    Create Z_R, X_R, Y_R,(sorted arrays containing the "right" half, those with x_i > x̄)
    δ' = min{CP_Almost2D(X_L, Y_L, Z_L, δ), CP_Almost2D(X_R, Y_R, Z_R, δ)}
    // Combine
    Create Y_δ', which is a sorted array containing those points in P with x-coordinate within
    δ' of x̄.
    Let p_i be the i^th point in Y_δ'.
    for i = 1 to |Y_δ'| − 31 do
        for j = 1 to 31 do
            if d(p_i, p_{i+j}) < δ* then
                δ* = d(p_i, p_{i+j})
            end
        end
    end
    return δ*
end
```