ADA Mini HW #2

Student Name: 林楷恩
Student ID: b07902075

1 Pseudo-code

NOTE: The pseudo-code of function `ClosestPair2D` is taken from the class slides.

Algorithm 1: Closest Pair On Cylindrical Surface

```
input: a set of points P
       where points are defined as (θ,h), 0 ≤ θ ≤ 2π is the angle, and h is the height,
       contant r indicates the radius of cylinder
output: int

function ClosestPairCylin(P)
    // convert P to P' which is a set of points on 2D plane ...O(n)
    P' ← { (x,y) = (rθ,h) | (θ,h) ∈ P }
    // preprocess P' ...O(nlogn)
    sort P' by x− and y−coordinates
    // Run normal closest pair on 2D plane algorithm ...O(nlogn)
    δ ← ClosestPair2D(P')
    // consider pairs that cross the vertical line of θ = 0 ...O(n)
    P_r ← p_i in P' if p_i.x lie in [0,δ] ∪ [2πr − δ, 2πr]
    // "move" p_i.x ∈ [2πr − δ, 2πr] to [−δ, 0] by subtracting their x−coordinates by 2π ...O(n)
    foreach p_i.x ∈ [2πr − δ, 2πr] in P_r
        p_i.x ← p_i.x − 2πr
    // iterate through all candidate pairs and update δ ...O(n)
    foreach p_i in P_r:
        compute distance with p_{i+1}, p_{i+2}, ..., p_{i+7}
        update δ if a closer pair is found
    return δ

function ClosestPair2D(P)
    // base case ...O(1)
    if |P| ≤ 3
        brute−force finding closest pair and return it
    // Divide ...O(n)
    find a vertical line L s.t. both left and right planes contain half of the points
    // Conquer ...2T(n/2)
    left−pair, left−min ← ClosestPair2D(points in the left)
    right−pair, right−min ← ClosestPair2D(points in the right)
    // Combine ...O(n)
    δ ← min(left−min, right−min)
    remove points that are δ or more away from L
    foreach p_i in sorted candidates:
        compute distance with p_{i+1}, p_{i+2}, ..., p_{i+7}
        update δ if a closer pair is found
    return δ
```

2 Justification of Correctness

To prove the correctness of function `ClosestPairCylin`, I have to prove the correctness of function `ClosestPair2D` function.

- Proof by induction:
  - + **Base Case**: when $|P| \leq 3$, because we enumerate all possible pairs and check if it is the closest pair in P, so the answer it gets is always true.

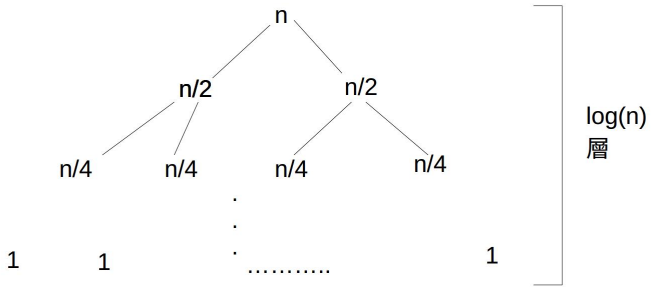
+ **Induction Step**: Assume the answer δ we get from the recursioin in left-plane and right-plane is correct, then we consider all the points within δ from the cut, and for every points we enumerating only pairs formed by the points whose y-coordinates are with δ from it(whose number is always ≤ 7 because pairs in the same side should have distance $\geq \delta$). Now we have tested all possible pairs that not included in the two subproblems, therefore the answer we get is correct.

Then, since we know that function `ClosestPair2D` is correct, I know the answer δ is correct on the 2D plane we convert from the cylindrical surface. Now we just need to consider pairs which cross the vertical line of $\theta = 0$, just as the method combining two subproblems in 2D closest pair problem. We consider the points whose x-coordinates lie in $[0, \delta]$ and $[2\pi r - \delta, 2\pi r]$, and for every points only consider the points whose y-coordinates are within δ from it. After enumerating all such pairs, we test all the pairs cross the cut, thus getting the correct answer.

3 Proof of Time Complexity

Let the time complexity of finding closest pair on 2D plane be $T(n)$, then the time complexity of the whole algorithm is $T(n) + O(n)$, so we must solve $T(n)$ first, by *recursion tree method*:

$$\begin{aligned} & T(n) = O(n) && \textit{...divide} \\ & \quad + 2T(n/2) && \textit{...conquer} \\ & \quad + O(n) && \textit{...combine} \\ \Longrightarrow\ & T(n) = 2T(n/2) + O(n) \end{aligned}$$



The sum of all nodes in above recursion tree is $nlogn$, thus the time complexity of `ClosestPair2D` is $O(nlogn)$. Therefore, the time complexity of the whole algorithm is $O(nlogn) + O(n) = O(nlogn)$.