演算法Homework 20190926

山下夏輝（Yamashita Natsuki）

R08922160

A screenshot of a social media post

Description automatically generated

//preparing

//n = (x, y, z) // origin is the center of the cylinder’s bottom.

// = (r cos θ, r sinθ, h)

//|n1n2| =

//reference

<https://www.youtube.com/watch?v=_nPvJpjKBmw>

slides of the classes

guidance of TA.吳士綸

**(1)**

Pre-Processing(P, N)

//sort

Sort P by x-, y-, z-coordinate and store in Px, Py and Pz

Closest-Pair(P, N)

//base case

If |N| <= 3

return Brute-Force-ClosestPair-and-Distance(P)

//recursive case

//divide

Find a vertical line L s.t.both planes contain half of the points

//conquer

left-min = Closest-Pair(points in the left, the num of the points in the left)

right-min = Closest-Pair(Points in the right, the num of the points in the right)

//combine

delta = min(left-min, right-min)

//cross two regions case

Remove points that are delta or more away from L

for point pi in candidates sorted by y-coordinate.

compute distances with pi+1, pi+2, …, pi+15

update delta if a closer pair is found

return the closest pair and its distance

**(2)**

Proof

In base case, |N| ≤ 3, the answer is returned by the brute force algorithm

In recursive case, |N| > 4, Closest-Pair(P,N) will be recalled recursively. The answer is in the variable delta and will be returned at last when the closest pair is not crossing two region divided by L.

In recursive case and in cross two regions case, |N| > 4, the value of delta will be updated when a closer pair is found and the answer is in the variable delta and will be returned at last.

**(3)**

Time Complexity for Finding Closest Pair

Pre-Processing(P, N)

sort: O(nlogn)

Closest-Pair(P, N)

base case: O(1)

divide: O(n)

conquer: 2T(n/2)

combine: O(1)

cross two regions case: O(n)

**T(n) = 2T(n/2)+O(nlogn)+O(n)+O(n)**

**= 2T(n/2) + O(nlogn)**

Theorem

T(n) = O(1) if n≤3

2T(n/2) + O(n) if n>3

T(n) = 2T(n/2) + cn

≤1

**≤** 2[2T(n/4)+cn/2]+cn = 4T(n/4)+2cn

≤ 4[2T(n/8)+cn/4]+2cn = 8T(n/8)+3cn

…

≤ 2kT(n/2k)+kcn

The expansion stops when 2k= n.

Then,

T(n) ≤ nT(1)+cnlogn

=O(n) + O(nlogn)

= O(nlogn)

Also, the complexity of the pre-processing sort algorithm is O(nlogn).

So, it can be proved that the complexity of this algorithm is O(nlogn).