<202135760 김현지 >

[main]

read number of dot

read dots

sort dots ascending order based on x-coordinate

result <- closetPair(0, numOfDot -1)

print result

[closetPair]

if (last - start == 0) //one dot

return 0

else if (last - start == 1) //two dot

return distance(dot[start], dot[last]);

else if (last - start == 2) //three dot

return min(distance(dot[start], dot[start + 1]), distance(dot[start], dot[start+2]), distance(dot[start + 1], dot[start+2]));

midIdx <-(start + last) / 2

ds <- closetPair(start, midIdx)

dl <- closetPair(midIdx +1, last)

minDis <- min(ds, dl);

for I <- start to last

if (dot[midIdx].x - minDis <= dot[i].x <= dot[midIdx].x + minDis) {

S[s\_count++] = dot[i]

sort S ascending order based on y-coordinate

for i<-0 to s\_count-1

for j <- 1 to 7

if (i + j >= s\_count)

break;

if (minDis > distance(S[i], S[i + j])) {

minDis <- distance(S[i], S[i + j]);

코드

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

typedef struct Dot {

float x;

float y;

};

Dot dot[30];

int cmpX(const void\*, const void\*);

int cmpY(const void\*, const void\*);

float closetPair(int, int);

float distance(Dot, Dot);

float min(float, float);

void main() {

int numOfDot;

float x, y;

while (true) {

//input

scanf("%d", &numOfDot);

if (numOfDot <= 0 || numOfDot > 30)

break;

for (int i = 0; i < numOfDot; i++) {

scanf("%f %f", &x, &y);

if (x > 40000 || y > 40000 || x < 0 || y < 0) {

printf("please enter number non-negative and less than 40000\n");

i--;

continue;

}

dot[i].x = x;

dot[i].y = y;

}

//sort dot ascending order based on x-coordinate

qsort(dot, numOfDot, sizeof(Dot), cmpX);

float result = closetPair(0, numOfDot-1);

if (result > 10000)

printf("Infinity\n");

else

printf("[result] %0.2f\n", result);

}

}

int cmpX(const void\* a, const void\* b) {

Dot\* d1 = (Dot\*)a;

Dot\* d2 = (Dot\*)b;

if (d1->x > d2->x) return 1;

else return -1;

}

int cmpY(const void\* a, const void\* b) {

Dot\* d1 = (Dot\*)a;

Dot\* d2 = (Dot\*)b;

if (d1->y > d2->y) return 1;

else return -1;

}

float closetPair(int start, int last) {

if (last - start == 0) {//one dot

return 0;

}

else if (last - start == 1) {//two dot

return distance(dot[start], dot[last]);

}

else if (last - start == 2) {//three dot

float minDis = min(distance(dot[start], dot[start + 1]), distance(dot[start], dot[start+2]));

return min(minDis, distance(dot[start + 1], dot[start+2]));

}

//four~

int midIdx = (start + last) / 2;

float ds = closetPair(start, midIdx);

float dl = closetPair(midIdx +1, last);

float minDis = min(ds, dl);

Dot S[15];

int s\_count = 0;//S에 들어간 값 개수

for (int i = start; i <= last; i++) {

if (dot[midIdx].x - minDis <= dot[i].x && dot[i].x <= dot[midIdx].x + minDis) {

//printf("S[%d](%f %f)", s\_count, dot[i].x, dot[i].y);

S[s\_count++] = dot[i];

}

}

//y좌표 기준으로 오름차순 정렬

qsort(S, s\_count, sizeof(Dot), cmpY);

//S중 minDis보다 작은 거리를 갖는 쌍이 있으면 misDis에 그 거리 넣기

for (int i = 0; i < s\_count; i++) {

for (int j = 1; j <= 7; i++) {

if (i + j >= s\_count)

break;

if (minDis > distance(S[i], S[i + j])) {

minDis = distance(S[i], S[i + j]);

}

}

}

return minDis;

}

float distance(Dot d1, Dot d2) {//두 점 사이 거리 반환

float x = d2.x - d1.x;

float y = d2.y - d1.y;

return sqrt(x \* x + y \* y);

}

float min(float a, float b) {//더 작은 값 반환

if (a > b)

return b;

else return a;

}

