Algorithm Analysis

CSI 3344 Name: Maiqi Hou

1.

Description

Default constructor of Points Class, and initial a point's x and y coordinate

Data Structure

None

Algorithm

Set up x and y coordniate equal to zero(zero is double type)

Analysis

Input N	None

Basic Operation	`Points() {`
	`x = 0.0;`
	`y = 0.0;`
	`}`

Summation or Recurrence Relation	1

Worst Case Analysis

T = O(1)

Best Case Analysis

T = O(1)

2.

Description

Get the x value of a point

Data Structure

None

Algorithm

Return the x's value.

Analysis

Input N	None

Basic Operation	`double gX() {`
	`return x;`
	`}`

Summation or Recurrence Relation	1

Worst Case Analysis

T = O(1)

Best Case Analysis

T = O(1)

3.

Description

Get the y value of a point

Data Structure

None

Algorithm

Return the y's value.

Analysis

Input N	None

Basic Operation	`double gY() {`
	`return y;`
	`}`

Summation or Recurrence Relation	1

Worst Case Analysis

T = O(1)

Best Case Analysis

T = O(1)

4.

Description

Store points' coordinates data.

Data Structure

Vector

Algorithm

Get the a point's x and y coordniate.

Store these data into a vector.

Analysis

Input N	N points(x and y coordinates)
Basic Operation	`void set(double x_c, double y_c, vector<Points>& v) { Points p; p.x = x_c; p.y = y_c; v.push_back(p); }`
Summation or Recurrence Relation	1

Worst Case Analysis

T = O(1)

Best Case Analysis

T = O(1)

4.

Description

Calculating the distance between two points. If they are less or equal to an equal D unit. Union them. In the end, counting the number of sets after points union.

Data Structure

Vector

Algorithm

If n greater than number of points

Show the number of sets

Return

If n less than number of points

Calculating all of the distance between the point and other points

If the distance is less or equal to D(which users given) and two points are not the same set

Union them

Analysis

Input N	N points(x and y coordinates)

Basic Operation	void dis(vector<Points> v, int& n, double d, disjointSet set) {
	if (n >= s) { // 1
	n = set.countSet(s);//1
	return;
	}
	for (int i = n+1; i < s; i++) { //n = s-i
	distance = sqrt(pow((v[i].gX() - v[n].gX()), 2.0) //n
	+ pow((v[i].gY() - v[n].gY()), 2.0));
	if (distance <= d) { //n
	if (!set.isSameSet(i, n)) { // n
	set.unionSet(i, n); // n
	}

	}
	}
	n += 1; //1
	dis(v, n, d,set); //1
	}
	}

Summation or Recurrence Relation	T (n) = O(n)

Worst Case Analysis

T(n) = O(n)

Best Case Analysis

T(n) = O(n)