**The closest pair of points**

**CODING:**

#include <iostream>

#include<cmath>

#include<algorithm>

using namespace std;

struct point {

int x, y;

};

int cmpX(point p1, point p2) { //to sort according to x value

return (p1.x < p2.x);

}

int cmpY(point p1, point p2) { //to sort according to y value

return (p1.y < p2.y);

}

float dist(point p1, point p2) { //find distance between p1 and p2

return sqrt((p1.x - p2.x)\*(p1.x - p2.x) + (p1.y - p2.y)\*(p1.y - p2.y));

}

float findMinDist(point pts[], int n) { //find minimum distance between two points in a set

float min = 9999;

for (int i = 0; i < n; ++i)

for (int j = i+1; j < n; ++j)

if (dist(pts[i], pts[j]) < min)

min = dist(pts[i], pts[j]);

return min;

}

float min(float a, float b) {

return (a < b)? a : b;

}

float stripClose(point strip[], int size, float d) { //find closest distance of two points in a strip

float min = d;

for (int i = 0; i < size; ++i)

for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)

if (dist(strip[i],strip[j]) < min)

min = dist(strip[i], strip[j]);

return min;

}

float findClosest(point xSorted[], point ySorted[], int n){

if (n <= 3)

return findMinDist(xSorted, n);

int mid = n/2;

point midPoint = xSorted[mid];

point ySortedLeft[mid+1]; // y sorted points in the left side

point ySortedRight[n-mid-1]; // y sorted points in the right side

int leftIndex = 0, rightIndex = 0;

for (int i = 0; i < n; i++) { //separate y sorted points to left and right

if (ySorted[i].x <= midPoint.x)

ySortedLeft[leftIndex++] = ySorted[i];

else

ySortedRight[rightIndex++] = ySorted[i];

}

float leftDist = findClosest(xSorted, ySortedLeft, mid);

float rightDist = findClosest(ySorted + mid, ySortedRight, n-mid);

float dist = min(leftDist, rightDist);

point strip[n]; //hold points closer to the vertical line

int j = 0;

for (int i = 0; i < n; i++)

if (abs(ySorted[i].x - midPoint.x) <dist) {

strip[j] = ySorted[i];

j++;

}

return min(dist, stripClose(strip, j, dist)); //find minimum using dist and closest pair in strip

}

float closestPair(point pts[], int n) { //find distance of closest pair in a set of points

point xSorted[n];

point ySorted[n];

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

xSorted[i] = pts[i];

ySorted[i] = pts[i];

}

sort(xSorted, xSorted+n, cmpX);

sort(ySorted, ySorted+n, cmpY);

return findClosest(xSorted, ySorted, n);

}

int main() {

point P[] ={{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};

int n = 6;

cout<< "The Closest distance is " <<closestPair(P, n);

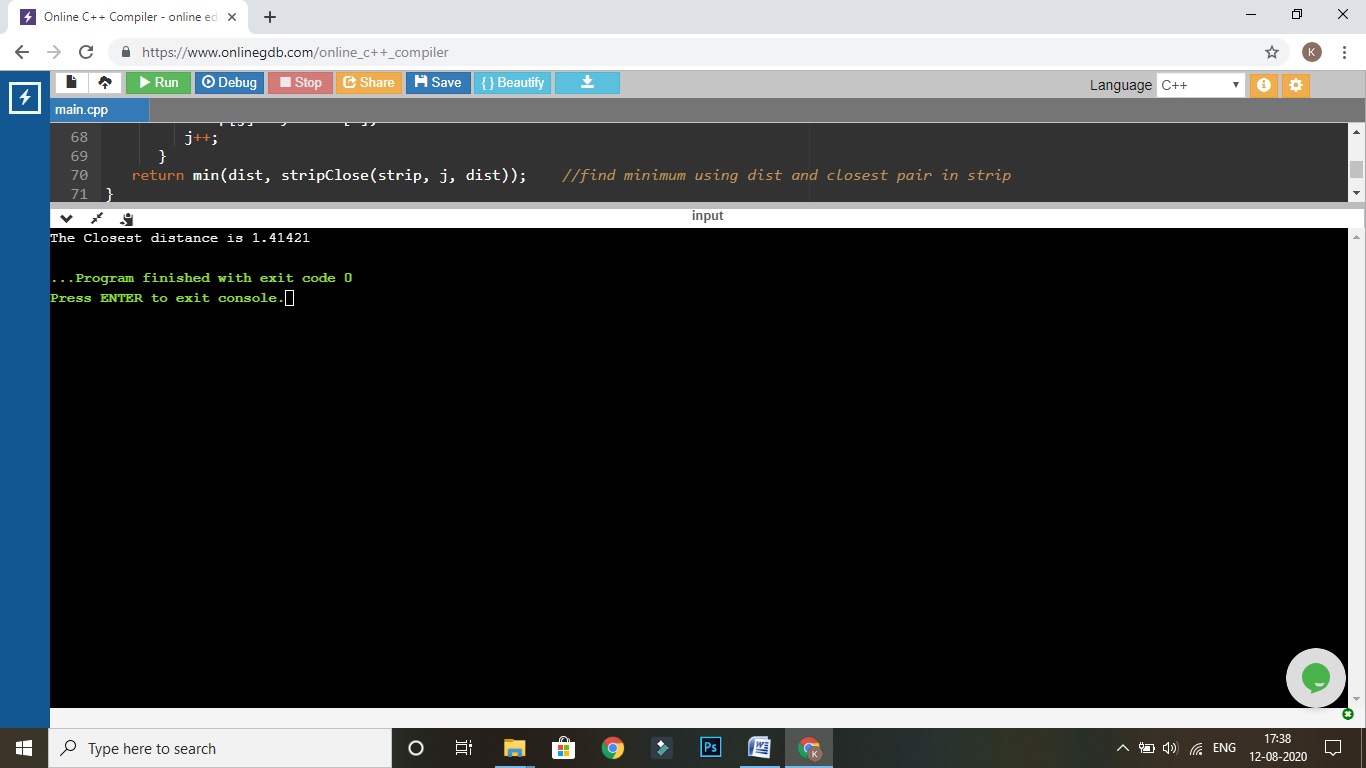
}

**EXPLANATION:**

To solve this problem, we take it as x axis and y axis and x axis is divided into two halves, after that smallest distance between two points is calculated in a recursive way. Using distances from the middle line, the points are separated into some strips. We will find the smallest distance from the strip array. At first two lists are created with data points, one list will hold points which are sorted on x values, another will hold data points, sorted on y values.

The time complexity of this algorithm will be O(n log n).

**OUTPUT:**

****