

MERCY KIRUBA S

Bachelor of Computer Applications

First Round - Final Code Assessment Document

Due Date: August 12, 2020

Email: mercykirubas18bca031@skasc.ac.in

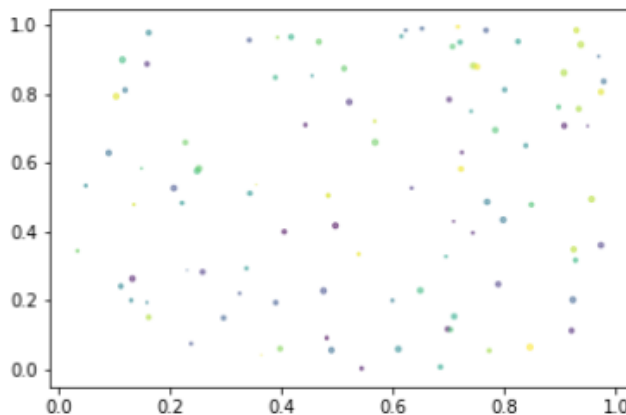
The Closest Pair of Points Problem

Problem Statement

Find the closest pair of points:

Generate a random pair of 100 values. Apply the closest pair algorithm to find the closest pair (distance).

A C++ Implementation from scratch implementation with neat documentation is expected.



Random pairs on a plane Example

The time complexity of the algorithm should be $O(N \log N)$

Aim

To find the pair of points, whose distance is minimum in a set of n points are given on the 2D plane.

First, I have divided the problem 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 and smallest distance is found from the strip array.

Two lists are created with data points:

- To hold data points which are sorted on x values.
- To hold data points which are sorted on y values.

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

Pseudocode

I based my program code on the following pseudocode. I should note that this pseudocode is high-level and does not fully express the minutiae of implementing the algorithm.

findMinDist (P,n)

Construct P1 and P2 //P1 is all the points stored by x-coordinate

//P2 is all the points stored by y-coordinate

return findMinDist (P1,P2)

findClosest (P1,P2,n)

//find the distance between closest pair of points in stripe array

Construct xSorted and ySorted

// xSorted-y sorted points in the left side

// ySorted- ypoints in the right side

leftDist = findClosest(xSorted, ySortedLeft, mid);

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

dist = min(leftDist, rightDist)

Find the distance between the closest pair of points in the given array

//although it may seem like O(n),it is actually O(nlogn) because of the

//mathematical property that any two points must be within fifteen positions of each other

findClosest(xSorted, ySorted, n)

Implementation

We are given a pair of n points in the plane, and the problem is to find out the closest pair of points in graph.

The formula for distance between two points p and q.

$$\|pq\| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Structure and variable declarations

I have introduced a data structure (point) to store a point (with coordinates x and y) and compare the operators required for two types of sorting. I defined the data type as float for the coordinates since my limit to find the shortest distance is from 0.0 to 1.0



```
1 #include <iostream>
2 #include <cmath>
3 #include <algorithm>
4 using namespace std;
5
6 struct point {
7     float x, y;
8 };
9
10 int cmpX(point p1, point p2) {    //to sort according to x value
11     return (p1.x < p2.x);
12 }
13
14 int cmpY(point p1, point p2) {    //to sort according to y value
15     return (p1.y < p2.y);
16 }
17
18 float dist(point p1, point p2) {    //find distance between p1 and p2
19     return sqrt((p1.x - p2.x)*(p1.x - p2.x) + (p1.y - p2.y)*(p1.y - p2.y));
20 }
21
22 float findMinDist(point pts[], int n) {    //find minimum distance between two points in a set
```

findMinDist(pointsList, n)

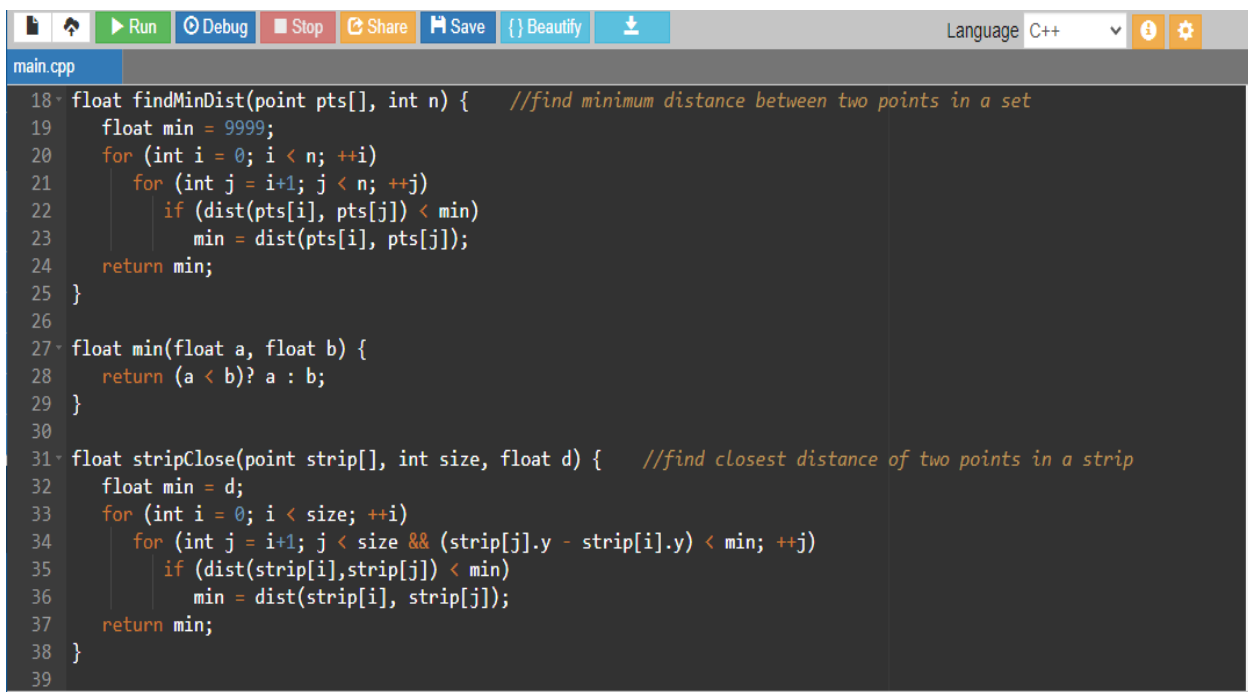
Input: Given point list and number of points in the list.

Output: Finds the minimum distance from two points.

stripClose(strips, size, dist)

Input: Different points in the strip, number of points, distance from the midline.

Output: Closest distance from two points in a strip.

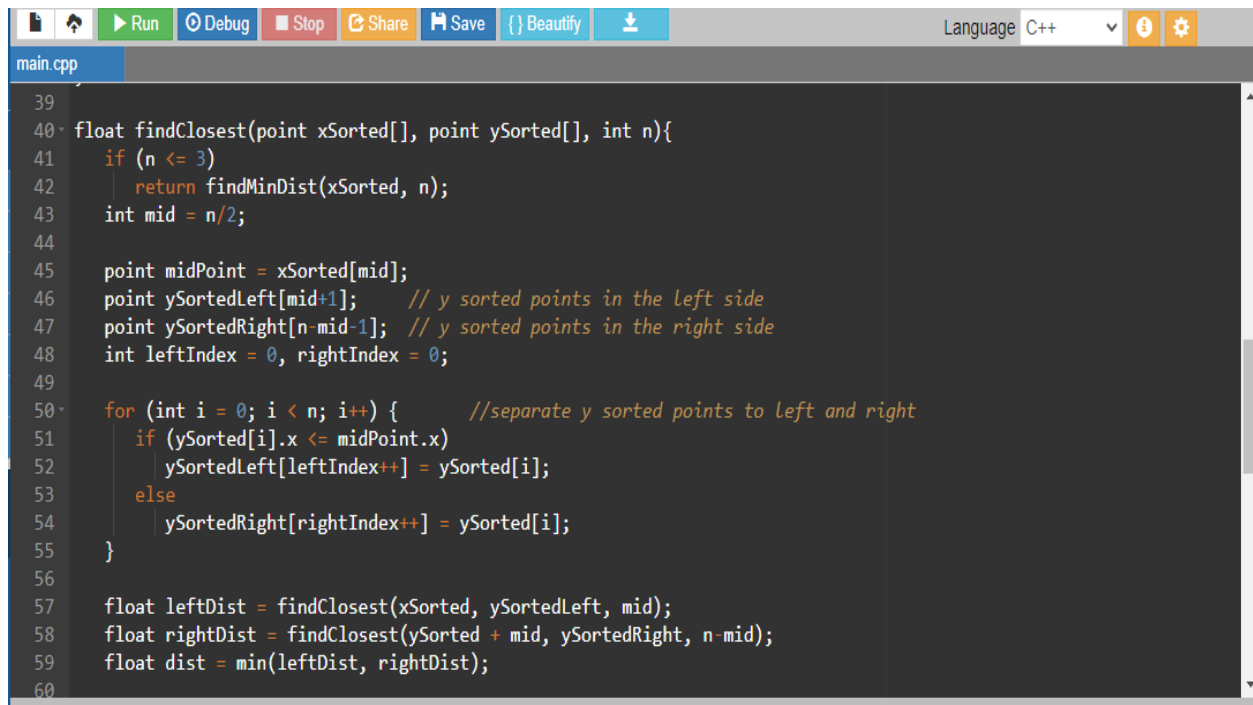
A screenshot of a C++ code editor interface. The editor has a toolbar at the top with icons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to C++. The code is in a file named 'main.cpp'. It contains three functions: findMinDist, min, and stripClose. The findMinDist function finds the minimum distance between two points in a set. The min function is a helper to find the minimum of two floats. The stripClose function finds the closest distance of two points in a strip.

```
18 float findMinDist(point pts[], int n) {    //find minimum distance between two points in a set
19     float min = 9999;
20     for (int i = 0; i < n; ++i)
21         for (int j = i+1; j < n; ++j)
22             if (dist(pts[i], pts[j]) < min)
23                 min = dist(pts[i], pts[j]);
24     return min;
25 }
26
27 float min(float a, float b) {
28     return (a < b)? a : b;
29 }
30
31 float stripClose(point strip[], int size, float d) {    //find closest distance of two points in a strip
32     float min = d;
33     for (int i = 0; i < size; ++i)
34         for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)
35             if (dist(strip[i], strip[j]) < min)
36                 min = dist(strip[i], strip[j]);
37     return min;
38 }
39
```

findClosest(xSorted, ySorted, n)

Input – Points sorted on x values, and points sorted on y values, number of points.

Output – Find the minimum distance from the total set of points.



```
39
40 float findClosest(point xSorted[], point ySorted[], int n){
41     if (n <= 3)
42         return findMinDist(xSorted, n);
43     int mid = n/2;
44
45     point midPoint = xSorted[mid];
46     point ySortedLeft[mid+1]; // y sorted points in the left side
47     point ySortedRight[n-mid-1]; // y sorted points in the right side
48     int leftIndex = 0, rightIndex = 0;
49
50     for (int i = 0; i < n; i++) { //separate y sorted points to left and right
51         if (ySorted[i].x <= midPoint.x)
52             ySortedLeft[leftIndex++] = ySorted[i];
53         else
54             ySortedRight[rightIndex++] = ySorted[i];
55     }
56
57     float leftDist = findClosest(xSorted, ySortedLeft, mid);
58     float rightDist = findClosest(ySorted + mid, ySortedRight, n-mid);
59     float dist = min(leftDist, rightDist);
60 }
```

Strip array

Using distances from the middle line, the points are separated into some strips and smallest distance is found from the strip array. It holds the points closer to the vertical line and finds the minimum using pair in strip.

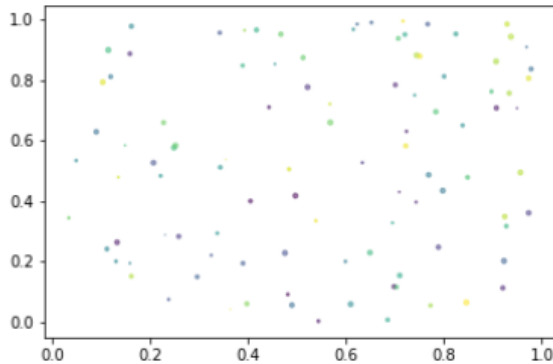
```
main.cpp
60
61 point strip[n];    //hold points closer to the vertical line
62 int j = 0;
63
64 for (int i = 0; i < n; i++)
65     if (abs(ySorted[i].x - midPoint.x) < dist) {
66         strip[j] = ySorted[i];
67         j++;
68     }
69 return min(dist, stripClose(strip, j, dist));    //find minimum using dist and closest pair in strip
70 }
71
72 float closestPair(point pts[], int n) {    //find distance of closest pair in a set of points
73     point xSorted[n];
74     point ySorted[n];
75
76     for (int i = 0; i < n; i++) {
77         xSorted[i] = pts[i];
78         ySorted[i] = pts[i];
79     }
80
81     sort(xSorted, xSorted+n, cmpX);
```

closestPair(pts ,n)

Finally, this function finds the closest pair in set of points.

```
main.cpp
74 }
75
76 float closestPair(point pts[], int n) {    //find distance of closest pair in a set of points
77     point xSorted[n];
78     point ySorted[n];
79
80     for (int i = 0; i < n; i++) {
81         xSorted[i] = pts[i];
82         ySorted[i] = pts[i];
83     }
84
85     sort(xSorted, xSorted+n, cmpX);
86     sort(ySorted, ySorted+n, cmpY);
87     return findClosest(xSorted, ySorted, n);
88 }
89
90 int main() {
91     point P[] ={{0.1, 0.2}, {0.22, 0.33}, {0.40, 0.50}, {0.5, 0.7}, {0.12, 0.10}, {0.3, 0.4}};
92     int n = sizeof(P)/sizeof(P[0]);
93     cout<< "The minimum distance is " <<closestPair(P, n);
94 }
95
```

main() function declaration and Output:



Random pairs on a plane Example

The limit in the above plane which is given as a problem is from 0.0 to 1.0. So, I have used those values to generate the shortest distance as output. Other than this, any number of random pairs of points can be assigned as a input in the pointer P[] and the closest pair with shortest distance can be generated as output.

```
main.cpp
74 }
75
76 float closestPair(point pts[], int n) { //find distance of closest pair in a set of points
77     point xSorted[n];
78     point ySorted[n];
79
80     for (int i = 0; i < n; i++) {
81         xSorted[i] = pts[i];
82         ySorted[i] = pts[i];
83     }
84
85     sort(xSorted, xSorted+n, cmpX);
86     sort(ySorted, ySorted+n, cmpY);
87     return findClosest(xSorted, ySorted, n);
88 }
89
90 int main() {
91     point P[] = {{0.1, 0.2}, {0.22, 0.33}, {0.40, 0.50}, {0.5, 0.7}, {0.12, 0.10}, {0.3, 0.4}};
92     int n = sizeof(P)/sizeof(P[0]);
93     cout<< "The minimum distance is " <<closestPair(P, n);
94 }
95
```

input

```
The minimum distance is 0.10198
...Program finished with exit code 0
Press ENTER to exit console.
```


//Code

```
#include <iostream>
#include<cmath>
#include<algorithm>
using namespace std;
struct point
{
    float x, y;
};

int cmpX(point p1, point p2)
{
    return (p1.x < p2.x);
}
int cmpY(point p1, point p2)
{
    return (p1.y < p2.y);
}

float dist(point p1, point 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)
{
    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)
```

```
{
```

```
    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];
```

```
    point ySortedRight[n-mid-1];
```

```
    int leftIndex = 0, rightIndex = 0;
```

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

```
{
```

```
    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];
```

```
    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));
}

```

```

float closestPair(point pts[], int n)
{
    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[] = {{0.1, 0.2}, {0.22, 0.33}, {0.40, 0.50}, {0.5, 0.7}, {0.12, 0.10}, {0.3, 0.4}};
    int n = sizeof(P)/sizeof(P[0]);
    cout<< "The minimum distance is " <<closestPair(P, n);
}

```

Reference:

http://www.sso.sy/sites/default/files/competitive%20programming%203_1.pdf

https://en.wikipedia.org/wiki/Closest_pair_of_points_problem#:~:text=6%20References-

[,Brute%2Dforce%20algorithm,smallest%20distance%2C%20as%20illustrated%20below.](#)

<https://www.geeksforgeeks.org/closest-pair-of-points-onlogn-implementation/>