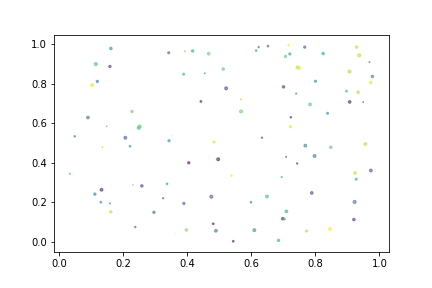
**Problem Statement 1:**

**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)

**SOLUTION:**

**FORMULA FOR DISTANCE BETWEEN TWO POINTS:**

**ALGORITHM:**

We use Divide and Conquer algorithm to find the closest pair

**1)** We sort all points according to x and y coordinates.

**2)**Divide all points in two halves.

**3)** Recursively find the smallest distances in both subarrays.

**4)**Take the minimum of two smallest distances. Let the minimum be d.

**5)** Create an array strip[] that stores all points which are at most d distance away from the middle line dividing the two sets.

**6)**Find the smallest distance in strip[].

**7)** Return the minimum of d and the smallest distance is calculated

**CODE:**

#include <iostream>

#include <float.h>

#include <stdlib.h>

#include <math.h>

using namespace std;

struct Point // A structure to represent a Point in 2D plan

{

int x, y;

};

int compareX(const void\* a, const void\* b) //To sort array of points in X coordinate

{

Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;

return (p1->x - p2->x);

}

int compareY(const void\* a, const void\* b) //To sort array of points in Y coordinate

{

Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;

return (p1->y - p2->y);

}

float dist(Point p1, Point p2) // A function to find the distance between two points

{

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

(p1.y - p2.y)\*(p1.y - p2.y)

);

}

float bruteForce(Point P[], int n)// A Brute Force method to return the smallest distance between two points

{

float min = FLT\_MAX;

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

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

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

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

return min;

}

float min(float x, float y) // A function to find a minimum of two float values

{

return (x < y)? x : y;

}

float stripClosest(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 closestUtil(Point Px[], Point Py[], int n)// A recursive function to find the smallest distance

{

if (n <= 3)

return bruteForce(Px, n);

int mid = n/2;

Point midPoint = Px[mid];

Point Pyl[mid];

Point Pyr[n-mid];

int li = 0, ri = 0;

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

{

if (Py[i].x <= midPoint.x && li<mid)

Pyl[li++] = Py[i];

else

Pyr[ri++] = Py[i];

}

float dl = closestUtil(Px, Pyl, mid);

float dr = closestUtil(Px + mid, Pyr, n-mid);

float d = min(dl, dr);

Point strip[n];

int j = 0;

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

if (abs(Py[i].x - midPoint.x) < d)

strip[j] = Py[i], j++;

return stripClosest(strip, j, d);

}

float closest(Point P[], int n)

{

Point Px[n];

Point Py[n];

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

{

Px[i] = P[i];

Py[i] = P[i];

}

qsort(Px, n, sizeof(Point), compareX);

qsort(Py, n, sizeof(Point), compareY);

return closestUtil(Px, Py, n);

}

int main() //Main function

{

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

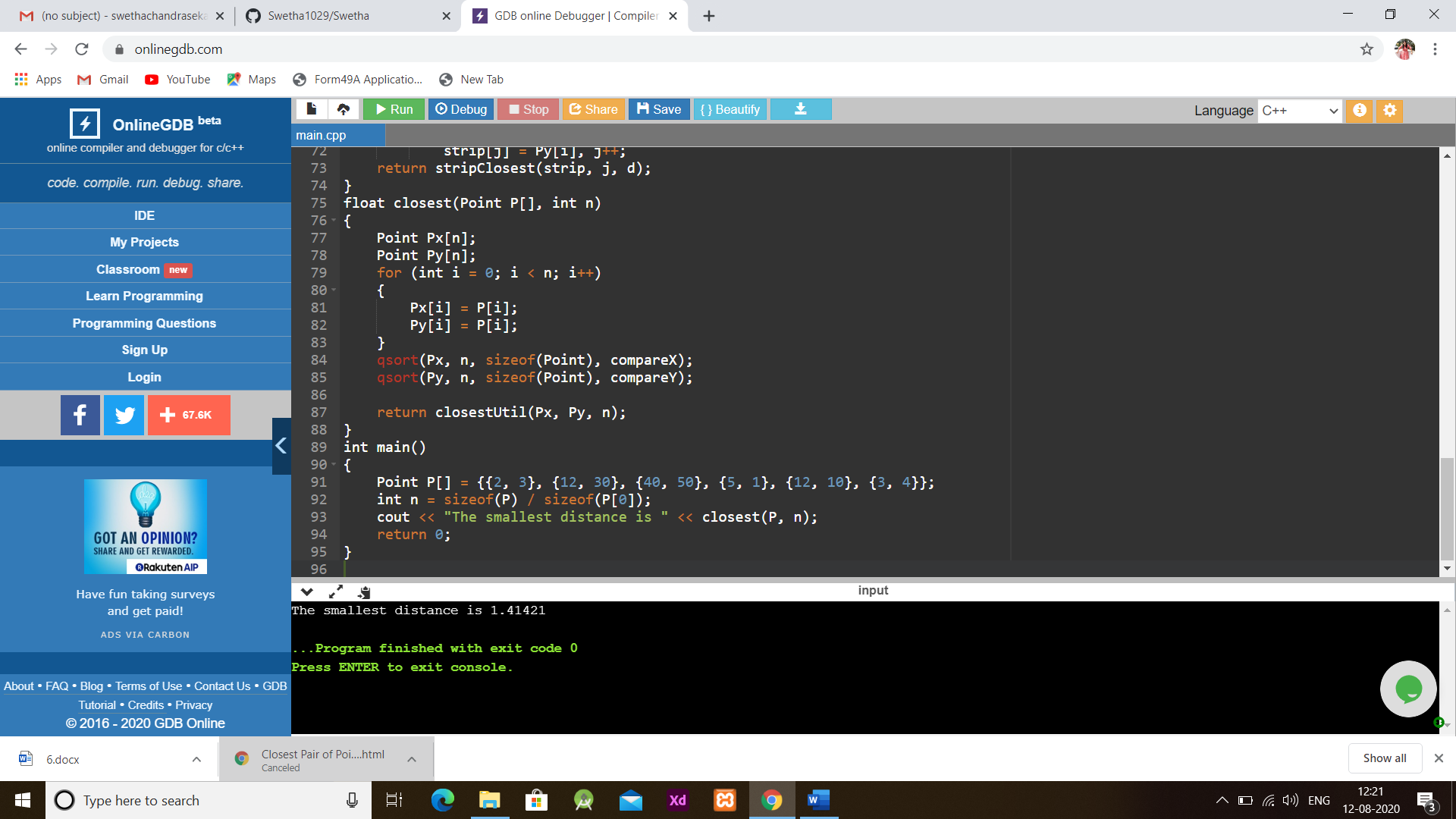
int n = sizeof(P) / sizeof(P[0]);

cout << "The smallest distance is " << closest(P, n);

return 0;

}

**OUTPUT:**



**EXPLANATION:**

* Main Function :
  + In main function an structure variable p is initialized.
  + Another variable n is initialized to find the size
  + In the print (cout) statement we call the function closest with two parameters.
* Closest ():
  + In Closest(),two arrays are declared and assigned using a for loop
  + qsort() is used to compare structure elemensts or of anytpe
  + compareX and compare are used to sort arrays
  + returns a recursive function closestutil().
* closestUtil():
  + It is used to find the smallest distance
  + If there are 2 or more points bruteforce() is implemented
  + Returns minimum of d and closest distance
* Bruteforce():
  + It returns the smallest distance between two points
* ClosestStrip():
  + It is to find the distance between the closest points of the strip of a given size
  + Pick all points one by one and try the next points till the difference between y coordinates is smaller than d.