

Find Closest Pair of Points

The random pair of 100 points is generated in the plane, and the problem is to find out the closest pair of points in the array.

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

The Brute force solution is $O(n^2)$, compute the distance between each pair and return the smallest. We can calculate the smallest distance in $O(n \log n)$ time using Divide and Conquer strategy. In this post, a $O(n \times (\log n)^2)$ approach is discussed. We will be discussing a $O(n \log n)$ approach in a separate post.

Algorithm

Following are the detailed steps of a $O(n (\log n)^2)$ algorithm.

Input: An array of n points $P[]$

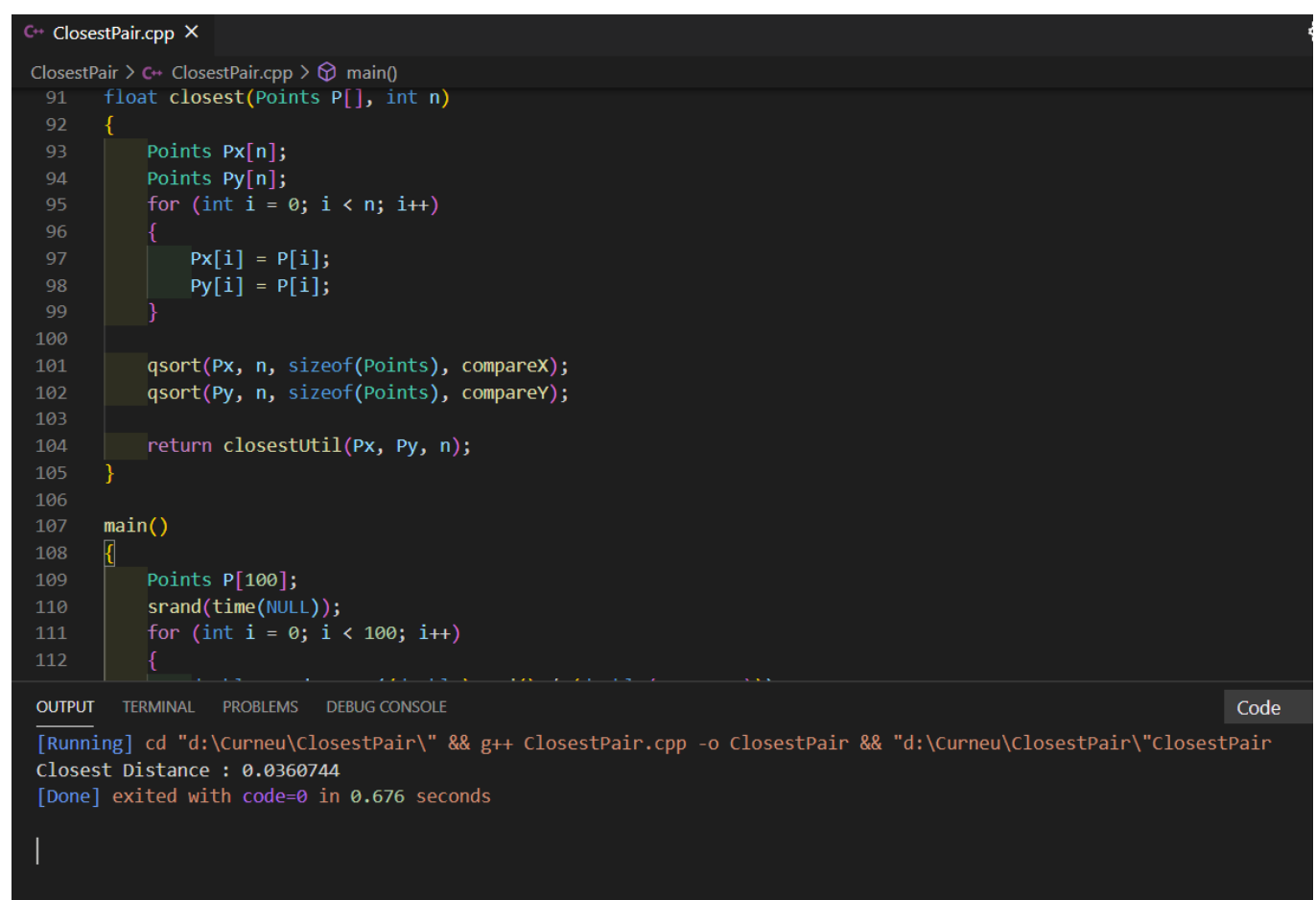
Output: The smallest distance between two points in the given array.

- 1) Find the middle point in the sorted array, we can take $P[n/2]$ as middle point.
- 2) Divide the given array in two halves. The first subarray contains points from $P[0]$ to $P[n/2]$. The second subarray contains points from $P[n/2+1]$ to $P[n-1]$.
- 3) Recursively find the smallest distances in both subarrays. Let the distances be d_l and d_r . Find the minimum of d_l and d_r . Let the minimum be d .
- 4) From the above 3 steps, we have an upper bound d of minimum distance. Now we need to consider the pairs such that one point in pair is from the left half and the other is from the right half. Consider the vertical line passing through $P[n/2]$ and find all points whose x coordinate is closer than d to the middle vertical line. Build an array $strip[]$ of all such points.
- 5) Sort the array $strip[]$ according to y coordinates. This step is $O(n \log n)$. It can be optimized to $O(n)$ by recursively sorting and merging.
- 6) Find the smallest distance in $strip[]$. This is tricky. From the first look, it seems to be a $O(n^2)$ step, but it is actually $O(n)$.

It can be proved geometrically that for every point in the strip, we only need to check at most 7 points after it (note that strip is sorted according to Y coordinate).

7) Finally return the minimum of d and distance calculated in the above step (step 6)

OUTPUT:



```
C++ ClosestPair.cpp X
ClosestPair > C++ ClosestPair.cpp > main()
91 float closest(Point P[], int n)
92 {
93     Point Px[n];
94     Point Py[n];
95     for (int i = 0; i < n; i++)
96     {
97         Px[i] = P[i];
98         Py[i] = P[i];
99     }
100
101     qsort(Px, n, sizeof(Point), compareX);
102     qsort(Py, n, sizeof(Point), compareY);
103
104     return closestUtil(Px, Py, n);
105 }
106
107 main()
108 {
109     Point P[100];
110     srand(time(NULL));
111     for (int i = 0; i < 100; i++)
112     {
113         P[i].x = rand() % 1000;
114         P[i].y = rand() % 1000;
115     }
116     float d = closest(P, 100);
117     cout << "Closest Distance : " << d << endl;
118 }
```

OUTPUT TERMINAL PROBLEMS DEBUG CONSOLE Code

```
[Running] cd "d:\Curneu\ClosestPair\" && g++ ClosestPair.cpp -o ClosestPair && "d:\Curneu\ClosestPair\ClosestPair
Closest Distance : 0.0360744
[Done] exited with code=0 in 0.676 seconds
```