Divide and Conquer | Set 2 (Closest Pair of Points)

We are given an array of n points in the plane, and the problem is to find out the closest pair of points in the array. This problem arises in a number of applications. For example, in air-traffic control, you may want to monitor planes that come too close together, since this may indicate a possible collision. Recall the following formula for distance between two points p and q.

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

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(nLogn) time using Divide and Conquer strategy. In this post, a O(n x (Logn)^2) approach is discussed. We will be discussing a O(nLogn) approach in a separate post.

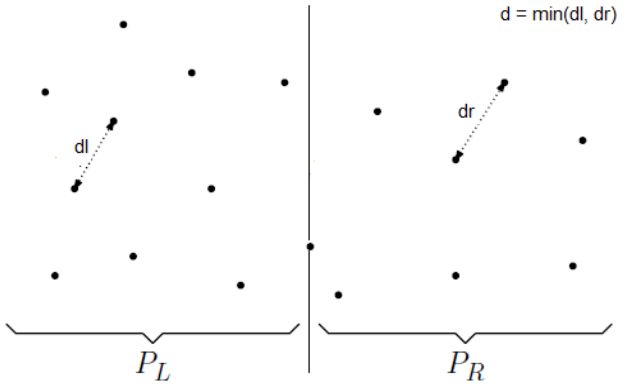
Algorithm
Following are the detailed steps of a O(n (Logn)^2) algortihm.
Input: An array of n points *P[]*
Output: The smallest distance between two points in the given array.

As a pre-processing step, input array is sorted according to x coordinates.

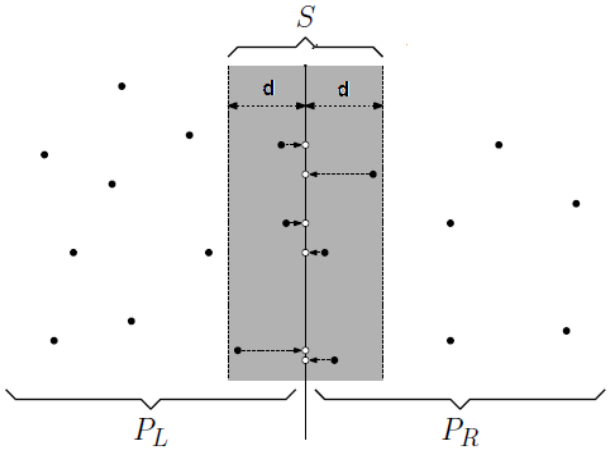
1) Find the middle point in the sorted array, we can take *P[n/2]* as middle point.

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 dl and dr. Find the minimum of dl and dr. Let the minimum be d.



4) From 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 left half and other is from right half. Consider the vertical line passing through 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(nLogn). It can be optimized to O(n) by recursively sorting and merging.

6) Find the smallest distance in strip[]. This is tricky. From 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 strip, we only need to check at most 7 points after it (note that strip is sorted according to Y coordinate). See this for more analysis.

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

Implementation
Following is C/C++ implementation of the above algorithm.

```
// A divide and conquer program in C/C++ to find the smallest distance from a
// given set of points.

#include <stdio.h>
#include <float.h>
#include <stdlib.h>
#include <math.h>

// A structure to represent a Point in 2D plane
struct Point
{
    int x, y;
};

/* Following two functions are needed for library function qsort().
   Refer: http://www.cplusplus.com/reference/clibrary/cstdlib/qsort/ */

// Needed to sort array of points according to X coordinate
int compareX(const void* a, const void* b)
{
    Point *p1 = (Point *)a,  *p2 = (Point *)b;
    return (p1->x - p2->x);
}
// Needed to sort array of points according to Y coordinate
int compareY(const void* a, const void* b)
{
    Point *p1 = (Point *)a,   *p2 = (Point *)b;
    return (p1->y - p2->y);
}

// A utility function to find the distance between two points
float dist(Point p1, Point p2)
{
    return sqrt( (p1.x - p2.x)*(p1.x - p2.x) +
                 (p1.y - p2.y)*(p1.y - p2.y)
               );
}

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

```
}

// A utility function to find minimum of two float values
float min(float x, float y)
{
    return (x < y)? x : y;
}


// A utility function to find the distance beween the closest points of
// strip of given size. All points in strip[] are sorted accordint to
// y coordinate. They all have an upper bound on minimum distance as d.
// Note that this method seems to be a O(n^2) method, but it's a O(n)
// method as the inner loop runs at most 6 times
float stripClosest(Point strip[], int size, float d)
{
    float min = d;  // Initialize the minimum distance as d

    qsort(strip, size, sizeof(Point), compareY);

    // Pick all points one by one and try the next points till the difference
    // between y coordinates is smaller than d.
    // This is a proven fact that this loop runs at most 6 times
    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;
}
```

```
// A recursive function to find the smallest distance. The array P contains
// all points sorted according to x coordinate
float closestUtil(Point P[], int n)
{
    // If there are 2 or 3 points, then use brute force
    if (n <= 3)
        return bruteForce(P, n);

    // Find the middle point
    int mid = n/2;
    Point midPoint = P[mid];

    // Consider the vertical line passing through the middle point
    // calculate the smallest distance dl on left of middle point and
    // dr on right side
    float dl = closestUtil(P, mid);
    float dr = closestUtil(P + mid, n-mid);

    // Find the smaller of two distances
    float d = min(dl, dr);

    // Build an array strip[] that contains points close (closer than d)
```

```
    // to the line passing through the middle point
    Point strip[n];
    int j = 0;
    for (int i = 0; i < n; i++)
        if (abs(P[i].x - midPoint.x) < d)
            strip[j] = P[i], j++;

    // Find the closest points in strip.  Return the minimum of d and closest
    // distance is strip[]
    return min(d, stripClosest(strip, j, d) );
}

// The main functin that finds the smallest distance
// This method mainly uses closestUtil()
float closest(Point P[], int n)
{
    qsort(P, n, sizeof(Point), compareX);

    // Use recursive function closestUtil() to find the smallest distance
    return closestUtil(P, n);
}

// Driver program to test above functions
int main()
{
    Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};
    int n = sizeof(P) / sizeof(P[0]);
    printf("The smallest distance is %f ", closest(P, n));
    return 0;
}
```

Output:

```
The smallest distance is 1.414214
```

Time Complexity Let Time complexity of above algorithm be T(n). Let us assume that we use a O(nLogn) sorting algorithm. The above algorithm divides all points in two sets and recursively calls for two sets. After dividing, it finds the strip in O(n) time, sorts the strip in O(nLogn) time and finally finds the closest points in strip in O(n) time. So T(n) can expressed as follows
T(n) = 2T(n/2) + O(n) + O(nLogn) + O(n)
T(n) = 2T(n/2) + O(nLogn)
T(n) = T(n x Logn x Logn)

Notes
1) Time complexity can be improved to O(nLogn) by optimizing step 5 of the above algorithm. We will soon be discussing the optimized solution in a separate post.
2) The code finds smallest distance. It can be easily modified to find the points with smallest distance.
3) The code uses quick sort which can be O(n^2) in worst case. To have the upper bound as O(n (Logn)^2), a O(nLogn) sorting algorithm like merge sort or heap sort can be used

References:
http://www.cs.umd.edu/class/fall2013/cmsc451/Lects/lect10.pdf

http://www.youtube.com/watch?v=vS4Zn1a9KUc
http://www.youtube.com/watch?v=T3T7T8Ym20M
http://en.wikipedia.org/wiki/Closest_pair_of_points_problem

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Closest Pair of Points, Divide and Conquer

Writing code in comment? Please use **ideone.com** and share the link here.

nemo • 2 months ago
You are sorting inside StripClosest, we should not do that! sorting is only done at the very beginning

Guest • 3 months ago
Why can't we double loop for 4 points in splitting condition instead of 7 since we know that 4 points are towards one half and the other

Guest • 6 months ago
In code it is written that This loop will run at most six time
But suppose if we have 10 point (0,1),(10^6-1,1.1), (10^6-2,1.2),.....(10^6-9,1.9) in strip array and distance of strip is 10^6 on either side.
Than in first loop of strip closest function it will compare 1st point to all other points.
But always(9 times) difference between y coordinate comes out to be less than min as first time difference is 0.1 and min is 10^6 ,in second time
difference is 0.2 and min is 999999.00001 now again value of min will decrease. Hence in this

difference is 0.2 and min is 999999.00001 now again value of min will decrease. Hence in this way more than six time loop will run .
Am i wrong or i didnt understand meaning of running six time.

^ | v • Reply • Share ›

Salazar ➦ Guest • 4 months ago

The code would first sort in terms of the x coordinate, so your input would be rearranged to be:

(0,1), (10^6-9,1.9), (10^6-8,1.8), ... , (10^6-2,1.2), (10^6-1,1.1)

On the first recursion it should find the strip to be at the midX = (10^6-5, 1.5) (location P[10/2] = P[5], or the 6th item).

And it would split into the left and right subsections from there and find the min_dist for each recursively. Now lets say it found min_dist and we are now searching down the strip. The min_dist would be d = sqrt(1 + .1^2) = sqrt(1+.01) ~= 1.005.

We strip coordinates which aren't at least a distance min_dist from the strip axes (or middle x). The middle is at x = 10^6-6, and we try to reach out 1.005 in the +x and -x direction. The only points which would be 1.005 within 10^6-6 are 10^6-5 and 10^6-7. No other point is within that distance.

Now we get to the loop which you are referring to, and lets by induction the recursive step returns the correct result, so all we have to analyze is the final step. The y-

see more

^ | v • Reply • Share ›

Anuj • 7 months ago

The O(nlog n) solution is posted at http://www.geeksforgeeks.org/c.... Can you please add it to this thread.

2 ^ | v • Reply • Share ›

Serif • a year ago

Is the theme of this article intended to be divide and conquer algorithms in general of just an O(n log n) solution to the closest pair problem? If the latter, then wouldn't a trailing edge algorithm provide a much simpler solution?

<script src="http://ideone.com/e.js/GyXwt0" type="text/javascript"></script>

(Not sure how to embed ideone scripts here.)

^ | v • Reply • Share ›

shiva • a year ago

How come the time complexity for brute force is O(n^2) ...it should be O(n!) right?

• Reply • Share ›

Josh shiva • a year ago

In that method all distances (binary relations) between all points are computed ... ~ C(n, 2) = n(n-1)/2

• Reply • Share ›

Guest • a year ago

• Reply • Share ›

rem45acp • 2 years ago

I really truly wish this could have been implemented in such a way that it calculates what two Points are the closest, not the distance between them. It's is proving extremely difficult to modify it so that it can do this. The title even says Closest Pair of Points.

• Reply • Share ›

annonymoe rem45acp • a year ago

why difficult.. whenever you update the closest distance keep a global variable to track those two points

6 • Reply • Share ›

Yash • 2 years ago

What if I partition the space in such a way that 1 partition has n/4 points and second one has 3n/4 points. Then too the complexity will remain O(nlogn). What if first partition has sqRoot(n) then what.

• Reply • Share ›

rohan • 2 years ago

Do we need to sort y cordinates?...as we are passing it again so i dnt think its will be required to sort it ...plz clarify?

• Reply • Share ›

Salazar rohan • 4 months ago

Yes, we do. We only sorted in terms of the X coordinates, so our points are from smallest to largest in the x position. During the stripClosest routine, we have the following inner-loop:

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

This will not work if strip[j].y - strip[i].y < 0, so strip[i].y <= strip[j], I mean, technically the computer will still find the min coordinate, but we can no longer say that the routine is not of O(n^2) complexity since we can't assure that the inner loop doesn't have a max number of calls that will be made to it. We might as well just say that j < size if the y coordinates in strip[] aren't sorted

coordinates in strip[] aren't sorted.

• Reply • Share ›

sakshi chourasia • 2 years ago

hi,
I have a doubt in algorithm. In line 3 of algo we goes on recursively dividing and finding dl and dr and then d=min(dl,dr) but there is also possibility of existence of dlr so in every recursive call we should also find dl,dr,dlr and then d should be min(dl,dr,dlr).

• Reply • Share ›

Salazar ➚ sakshi chourasia • 4 months ago

We can't find dlr unless we know how to check across without having to go through every point between the left and right sections. So after d = min(dl, dr), we check the strip which is in the middle of the x coordinates, and we check a horizontal distance outwards of d for possible points with a lower d (widening our check will only increase runtime, but not correctness). stripClosest does the job of find that dlr, so min(d, stripClosest(strip, j, d) is just like saying min(dl, dr, dlr) or min(dlr, min(dl, dr)) with dlr = stripClosest(strip, j, d).

• Reply • Share ›

Anupam • 2 years ago

The link for the proof of 6th point is broken.
Please fix it!

• Reply • Share ›

GeeksforGeeks Mod ➚ Anupam • 2 years ago

Thanks for pointing this out. We have changed the link.

• Reply • Share ›

knownAnonymous • 2 years ago

I am not getting he concept of 7 points. Please elaborate.

• Reply • Share ›

Mike • 2 years ago

It's been a while since you wrote this, but I really want to thank you for writing this, it's the ONLY article that actually made me understand this algorithm.

Other sources fail to explain or even mention the O(n * log n * log n) complexity, they just blatantly lie and say it's O(n * log n).

Thank you very very much.

P.S. Not sure where can I find the O(n * log n) post here (?), although I think I know how to do it (pass the y-sorted arrays and merge them instead of sorting again)

2 • Reply • Share ›

LAKHAN SINGH MEENA • 2 years ago

#include
#include
#include
#include
#include

// A structure to represent a Point in 2D plane
struct Point
{
int x, y;
};

/* Following two functions are needed for library function qsort().
Refer: http://www.cplusplus.com/refer... */

// Needed to sort array of points according to X coordinate
int compareX(const void* a, const void* b)
{
Point *p1 = (Point *)a, *p2 = (Point *)b;
return (p1->x - p2->x);

see more

• Reply • Share ›

abcon23 • 2 years ago

```
float dr = closestUtil(P + mid, n-mid);
```

it should be

```
float dr = closestUtil(P + mid, n-mid-1);
```

1 • Reply • Share ›

Subscribe Add Disqus to your site Privacy DISQUS

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Follow @GeeksforGeeks

Recent Comments

- Nikhil kumar

  public class...

  Print missing elements that lie in range 0 – 99 · 5 minutes ago

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 27 minutes ago

- Vlad

  Thanks. Very interesting lectures.

  Expected Number of Trials until Success · 1 hour ago

- cfh

  My implementation which prints the index of the...

  Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 2 hours ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 2 hours ago