Closest Pair of Points | O(nlogn) Implementation

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.

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

We have discussed a divide and conquer solution for this problem. The time complexity of the implementation provided in the previous post is O(n (Logn)^2). In this post, we discuss an implementation with time complexity as O(nLogn).

Following is a recap of the algorithm discussed in the previous post.

1) We sort all points according to x 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 calculated in above step 6.

The great thing about the above approach is, if the array strip[] is sorted according to y coordinate, then we can find the smallest distance in strip[] in O(n) time. In the implementation discussed in previous post, strip[] was explicitly sorted in every recursive call that made the time complexity O(n (Logn)^2), assuming that the sorting step takes O(nLogn) time.
In this post, we discuss an implementation where the time complexity is O(nLogn). The idea is to presort all points according to y coordinates. Let the sorted array be Py[]. When we make recursive calls, we need to divide points of Py[] also according to the vertical line. We can do that by simply processing every point and comparing its x coordinate with x coordinate of middle line.

Following is C++ implementation of O(nLogn) approach.

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

#include <iostream>
#include <float.h>
#include <stdlib.h>
#include <math.h>
using namespace std;

// 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

    // 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 Px contains
// all points sorted according to x coordinates and Py contains all points
// sorted according to y coordinates
float closestUtil(Point Px[], Point Py[], int n)
{
    // If there are 2 or 3 points, then use brute force
    if (n <= 3)
        return bruteForce(Px, n);

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


    // Divide points in y sorted array around the vertical line.
    // Assumption: All x coordinates are distinct.
    Point Pyl[mid+1];   // y sorted points on left of vertical line
    Point Pyr[n-mid-1];  // y sorted points on right of vertical line
    int li = 0, ri = 0;  // indexes of left and right subarrays
    for (int i = 0; i < n; i++)
    {
      if (Py[i].x <= midPoint.x)
         Pyl[li++] = Py[i];
      else
         Pyr[ri++] = Py[i];
    }

    // 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(Px, Pyl, mid);
    float dr = closestUtil(Px + mid, Pyr, 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(Py[i].x - midPoint.x) < d)
            strip[j] = Py[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)
{
    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);

    // Use recursive function closestUtil() to find the smallest distance
    return closestUtil(Px, Py, 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]);
    cout << "The smallest distance is " << closest(P, n);
    return 0;
}
```

Output:

```
The smallest distance is 1.41421
```

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. Also, it takes O(n) time to divide the Py array around the mid vertical line. 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(n) + O(n)
T(n) = 2T(n/2) + O(n)
T(n) = T(nLogn)

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

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

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: Divide and Conquer, geometric algorithms

Tweet 1

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

25 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

coder • a month ago

what will happen when input 6 points all x coordinate same then during construction of pyl[] and pyr[] all points go to pyl[] even it dosent hav that much memory and nothing into pyr[] ;

• Reply • Share ›

The Internet • 5 months ago

I echo the sentiment below: too much implementation!

http://ideone.com/t1wiLn

The above should be C++11 compatible - at least I tried. It runs in ideone and on my machine at least.

Here are the main points of what I have changed:

- Dealing with all the dynamically allocated arrays is horrible (which you have to do if you want dynamically sized arrays and not use GCC extensions as above). I use local vectors and never have to worry about cleanup.
- Calculating the minimum distance in the base case can be a special case of the one that does so for a strip: the strip simply extends over all points passed in.
- Numerous changes for consistency. Seriously, if you cannot decide if your arguments should be lower or upper case, it looks like you don't really care when writing code.
- Now using C++ vectors so I don't have to pass sizes and index bounds around everywhere.
- Your algorithm is a bit inefficient: you only need to put the points from the right side into the strip and not all points from the set sorted by y.
- I got rid of most of the index-based for loops and I'm now just using iterators over the vectors. The code looks a lot cleaner that way.
- Overall code reduction by about 50%. Retested and seems functionally equivalent. Let me

know if you find bugs!

Can we please clean up this article?

• Reply • Share ›

ExamToday The Internet • 3 months ago

You are not using Points py in your logic. You pass it as a parameter but you don't use it..

In here you should be looping through:

for (auto p : px) {

if (p.x <= px[mid].x) pyl.push_back(p);

else pyr.push_back(p);

}

py not px, because you have to populate pyl and pyr with values from py.

So the code above should be:
for (auto p : py) {

if (p.x <= px[mid].x) pyl.push_back(p);

see more

• Reply • Share ›

abc The Internet • 5 months ago

why show off?

1 • Reply • Share ›

Gokhan • 5 months ago

Can anybody tell me what "Px + mid" does in C++? I'm not familiar with C++. Is it taking upper half of the array?

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

• Reply • Share ›

lkjfaldjdalskfjl Gokhan • 5 months ago

That's pointer arithmetic. The expression yields a pointer that is moved forward by "mid" elements from the current start Px. It is the same as writing "&Px[mid]". The ampersand gets the address of the element "mid" in "Px".

Ugh, I really don't like how this source is written :(

Reply • Share ›

Bango • 6 months ago

I appreciate what you've done. But kind of misleading. This doesn't give the closest pair of points in a set, but rather gives the shortest distance between any 2 points in a set. With the 2 closest points, it would be easy to calculate the distance between them, but not the other way around.

Reply • Share ›

arjomanD • a year ago

Too much Implementation !!
c++
==
http://paste.ubuntu.com/745054...

Reply • Share ›

The Internet → arjomanD • 5 months ago

Thank you! That code looks a lot better.

Reply • Share ›

ColacX • a year ago

Point Px[n];

is this valid C++ code? doesn't compile for me.
where is the memory located, on the stack or heap?

Reply • Share ›

The Internet → ColacX • 5 months ago

The non-const array definitions are not standard C++. The author was probably using GCC when writing this code. G4G: when you test your code (if you do), can you please use the flag -pedantic or some of that sort? I see code like this on here a lot. If you don't feel like writing standard C++, at least say so.

Also, using int for array indexing is not good. There are 64 bit platforms where int is 32 bit, while arrays can have more elements. Use size_t or ptrdiff_t.

Also, some of the code on G4G is seriously upsetting my OCD: why do you write "strip[j] = Py[i], j++;" if you just used "Pyr[ri++] = Py[i];" a few lines up? Why not "strip[j++] = Py[i]"? Why not simply two lines? Are you going to have all blocks of expressions as a long string of comma-separated values until you fill up the line?

Please, please, please G4G: read over the code you post a little more, make sure it's consistent, pay attention to grammar and spelling, use C++ headers if you claim to write C++, set your compile flags properly, etc. It's the same thing on almost every article on here.

Reply • Share ›

The Internet → The Internet • 5 months ago

Ah, also: there is std::min and std::max. You don't need to re-define those methods for every article on this site ...

Reply • Share ›

adf → ColacX • 6 months ago

You'd better allocate the memory for it

Reply • Share ›

Sekhar • a year ago

Does this algorithmn take care of mid && px+mid points for min distance calculation ? ex i have 6 pair of points, does this take care of 3rd and 4th pair in calculation for min distance

Reply • Share ›

Serif • a year ago

A simple n log n trailing edge algorithm:

<script src="http://ideone.com/e.js/SzAd4N" type="text/javascript"></script>

Reply • Share ›

rohan • a year ago

GeeksforGeeks

you are using one variable extra in both pyl n pyr as below

Point Pyl[mid+1]; // y sorted points on left of vertical line

if mid==3 for 6 elements then we need array of 3 elements (mid elements instead of mid+1)

Point Pyr[n-mid-1]; // y sorted points on right of vertical line//same reason

kindly update it as following as it creates a lot of confussion

Point Pyl[mid]; // y sorted points on left of vertical line

Point Pyr[n-mid]; // y sorted points on right of vertical line

Reply • Share ›

Guest → rohan • a year ago

I don't think so. I am implementing this code right now and when I follow your suggestion, I get an out_of_range error. When I follow G4G's code, I do not.

Reply • Share ›

Ofer · a year ago

An O(n) algorithm using some assumptions:

If you know the maximum distance you're looking for, i.e points with distance greater than say 'r' don't matter, and you can properly discretize your input into integer coordinates, and assuming the number of points closer than r is O(logn), and assuming the radius r is constant, then you can design an algorithm with O(n) runtime.

Here's an outline of the algorithm:

- For each point, you add the point and its neighbor points within the radius to a hashmap. If there's already a different point in the hashmap in that spot, then we know the two points are neighbors, i.e within radius r. We add their distance to a min heap.

- We take the minimum of the min heap as the minimum distance.

Since the heap has only O(logn) points, and so O(log^2(n)) pairs, adding all pairs of points to the heap is O(log^2(n)*log(logn)) throughout the algorithm.

As we went over the points once, the runtime is amortized O(n). If we want it O(n) worstcase, we can use a huge array instead of the hashmap.

· Reply · Share ›

viki · a year ago

This line will cause Pyl to go out of bound-

Pyl[i] = Py[i];

5 · Reply · Share ›

GeeksforGeeks Mod viki · a year ago

viki, thanks for pointing this out. We have updated the code.

1 · Reply · Share ›

Guest · a year ago

I've made this program in C

```
#include<stdio.h>

#include<conio.h>

void abs(int*);//Finds absolute Value

int diff(int,int);//Finds difference

void enter_ar(int *);//To enter Array
```

```
struct inf{

          int a,b,c;

          }z[50],f;
```

see more

Reply • Share ›

Guest • a year ago

I've tried to make this program in C, its output is correct

```
#include<stdio.h>
#include<conio.h>



void abs(int*);//Finds absolute Value
int diff(int,int);//Finds difference
void enter_ar(int *);//To enter Array

struct inf{
   int a,b,c;
   }z[50],f;

struct inf sm(struct inf*,int);//To find smallest in "struct inf" type array

void main()
{
```

see more

1 • Reply • Share ›

Guest • a year ago

I tried to make this program in C, its output is correct

```
#include<stdio.h>


#include<conio.h>


void abs(int*);//Finds absolute Value


int diff(int,int);//Finds difference
```

```
void enter_ar(int *);//To enter Array

struct inf{

          int a,b,c;

          }z[50],f;
```

see more

• Reply • Share ›

Vivek VV • a year ago

Can someone post the link to part one of this post.
Thanks

• Reply • Share ›

rahul Vivek VV • a year ago

http://www.geeksforgeeks.org/c...

• Reply • Share ›

-
-

-
-
    - 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

-