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

Google™ Custom Search

GeeksforGeeks

53,525 people like GeeksforGeeks.

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

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

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 f
// 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)
```


HP Chromebook 11

google.com/chromebook

Everything you need in one laptop.
Made with Google. Learn more.

Popular Posts

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
{
    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 po
// 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 o
// 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 dif
    // 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;
            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 co
// all points sorted according to x coordinates and Py contains all po
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
// 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
    // 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
    // distance is strip[]
    return min(d, stripClosest(strip, j, d) );
}
```

705 Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 9 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 49 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 52 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

AdChoices

▶ Source Code C++

▶ C++ Vector

▶ Math Geeks

AdChoices

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
// 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 dista
    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)

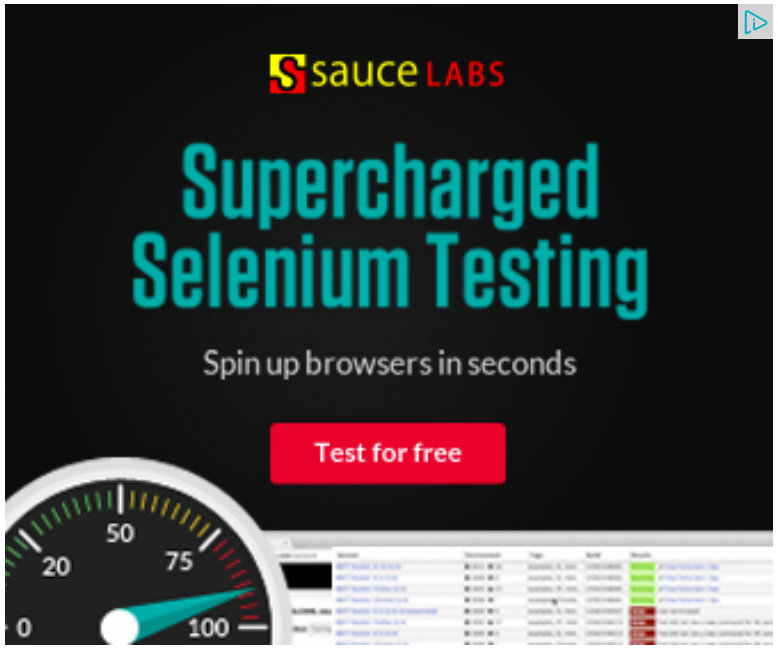
References:
http://www.cs.umd.edu/class/fall2013/cmsc451/Lects/lect10.pdf

▶ Points and Lines
▶ Points Of
▶ C++ Array
AdChoices
▶ C++ Array
▶ Programming C++
▶ Int C++

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

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 Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Find the maximum distance covered using n bikes

14 Tweet 3 1

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

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

14 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

arjomanD • 15 hours ago

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

ColacX • 19 days ago

Point Px[n];

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

Sekhar • a month ago

Does this algorithmn take care of mid && px+mid points for min distance calc
this take care of 3rd and 4th pair in calculation for min distance

Serif • 2 months ago

A simple n log n trailing edge algorithm:

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

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

rohan • 5 months 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 inst

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

Guest ➚ rohan • 23 days ago

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

Ofer • 5 months ago

An O(n) algorithm using some assumptions:

If you know the maximum distance you're looking for, i.e points with distance g
you can properly discretize your input into integer coordinates, and assuming t
O(logn), and assuming the radius r is constant, then you can design an algorit

Here's an outline of the algorithm:

- For each point, you add the point and its neighbor points within the radius to a
point in the hashmap in that spot, then we know the two points are neighbors,

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

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 pa
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 (
array instead of the hashmap.

viki • 5 months ago

This line will cause Pyl to go out of bound-

Pyl[i] = Py[i];

5 

GeeksforGeeks Mod viki • 5 months ago

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

1 

Guest • 5 months 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
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
struct inf{

          int a,b,c;

          }z[50],f;
```

see more

Guest • 5 months 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

void main()
{
```

see more

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

1 ^ | v •

Guest • 5 months 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

^ | v •

Vivek VV • 6 months ago

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

^ | v •

rahul ➚ Vivek VV • 6 months ago

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

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Subscribe Add Disqus to your site

@geeksforgeeks, **Some rights reserved** **Contact Us!** Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com