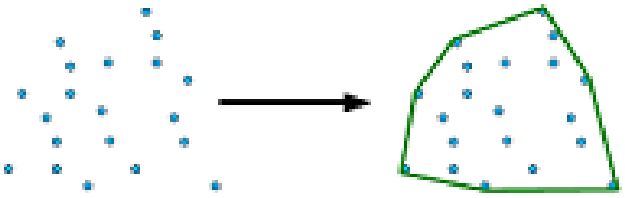
Convex Hull | Set 1 (Jarvis's Algorithm or Wrapping)

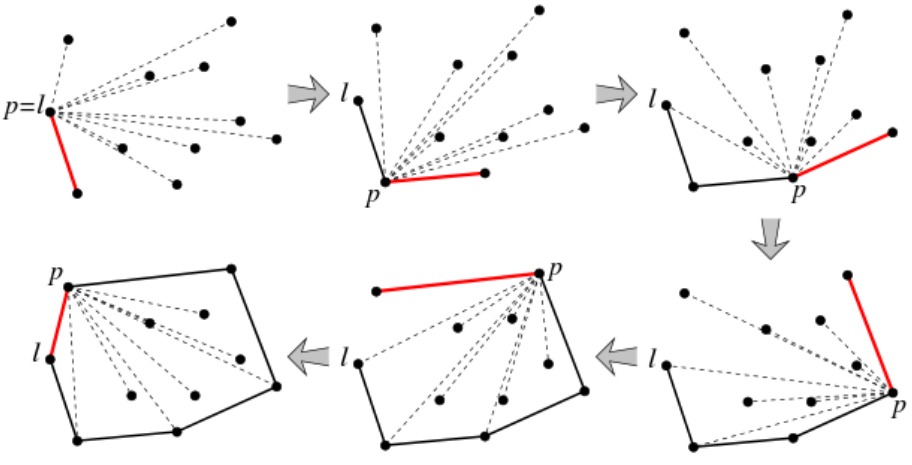
Given a set of points in the plane. the convex hull of the set is the smallest convex polygon that contains all the points of it.



We strongly recommend to see the following post first.
How to check if two given line segments intersect?

The idea of Jarvis's Algorithm is simple, we start from the leftmost point (or point with minimum x coordinate value) and we keep wrapping points in counterclockwise direction. The big question is, given a point p as current point, how to find the next point in output? The idea is to use orientation() here. Next point is selected as the point that beats all other points at counterclockwise orientation, i.e., next point is q if for any other point r, we have "orientation(p, r, q) = counterclockwise". Following is the detailed algorithm.

1) Initialize p as leftmost point.
2) Do following while we don't come back to the first (or leftmost) point.
…..**a)** The next point q is the point such that the triplet (p, q, r) is counterclockwise for any other point r.
…..**b)** next[p] = q (Store q as next of p in the output convex hull).
…..**c)** p = q (Set p as q for next iteration).



The execution of Jarvis's March.

Below is C++ implementation of above algorithm.

```
// A C++ program to find convex hull of a set of points. Refer
// http://www.geeksforgeeks.org/orientation-3-ordered-points/
// for explanation of orientation()
#include <bits/stdc++.h>
using namespace std;

struct Point
{
    int x, y;
};

// To find orientation of ordered triplet (p, q, r).
// The function returns following values
// 0 --> p, q and r are colinear
// 1 --> Clockwise
// 2 --> Counterclockwise
int orientation(Point p, Point q, Point r)
{
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;  // colinear
    return (val > 0)? 1: 2; // clock or counterclock wise
}

// Prints convex hull of a set of n points.
void convexHull(Point points[], int n)
{
    // There must be at least 3 points
    if (n < 3) return;

    // Initialize Result
    vector<Point> hull;

    // Find the leftmost point
    int l = 0;
    for (int i = 1; i < n; i++)
        if (points[i].x < points[l].x)
            l = i;

    // Start from leftmost point, keep moving counterclockwise
    // until reach the start point again.  This loop runs O(h)
    // times where h is number of points in result or output.
    int p = l, q;
    do
    {
        // Add current point to result
        hull.push_back(points[p]);

        // Search for a point 'q' such that orientation(p, x,
        // q) is counterclockwise for all points 'x'. The idea
        // is to keep track of last visited most counterclock-
        // wise point in q. If any point 'i' is more counterclock-
        // wise than q, then update q.
        q = (p+1)%n;
        for (int i = 0; i < n; i++)
        {
           // If i is more counterclockwise than current q, then
           // update q
           if (orientation(points[p], points[i], points[q]) == 2)
               q = i;
        }

        // Now q is the most counterclockwise with respect to p
        // Set p as q for next iteration, so that q is added to
        // result 'hull'
        p = q;
```

```
    } while (p != l);  // While we don't come to first point

    // Print Result
    for (int i = 0; i < hull.size(); i++)
        cout << "(" << hull[i].x << ", "
             << hull[i].y << ")\n";
}

// Driver program to test above functions
int main()
{
    Point points[] = {{0, 3}, {2, 2}, {1, 1}, {2, 1},
                      {3, 0}, {0, 0}, {3, 3}};
    int n = sizeof(points)/sizeof(points[0]);
    convexHull(points, n);
    return 0;
}
```

Run on IDE

Output: The output is points of the convex hull.

```
(0, 3)
(0, 0)
(3, 0)
(3, 3)
```

Time Complexity: For every point on the hull we examine all the other points to determine the next point. Time complexity is Θ(m * n) where n is number of input points and m is number of output or hull points (m <= n). In worst case, time complexity is $O(n^2)$. The worst case occurs when all the points are on the hull (m = n)

We will soon be discussing other algorithms for finding convex hulls.

Sources:

http://www.cs.uiuc.edu/~jeffe/teaching/373/notes/x05-convexhull.pdf

http://www.dcs.gla.ac.uk/~pat/52233/slides/Hull1x1.pdf

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

22 Comments Category: Geometric Tags: geometric algorithms, MathematicalAlgo

Related Posts:

- Number of Integral Points between Two Points
- Count Integral points inside a Triangle
- Orientation of 3 ordered points
- Find Simple Closed Path for a given set of points
- Minimum Cost Polygon Triangulation
- Find if two rectangles overlap
- Closest Pair of Points | O(nlogn) Implementation
- Given n line segments, find if any two segments intersect

(Login to Rate and Mark)

4 Average Difficulty : **4/5.0**
Based on **2** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

congratulations man, your code is running very well!

Very good.

• Reply • Share ›

Александр Плосков • 5 months ago

Point points[] = {{0, 0}, {0, 1}, {0,2}, {0, 3}, {0,5}, {0,4}, {1, 0}, {2, 0},

{3, 0}, {5, 0}, {4, 0}};
wrong answer
must be 3 points in convex hull

• Reply • Share ›

Kartik ↗ Александр Плосков • 3 months ago

Number of points don't matter. The idea is to minimize the area of polygon. So if the above program produces more points, then it doesn't mean it is incorrect. Please correct me if I am wrong.

• Reply • Share ›

buptwugh • 5 months ago

The print result should be changed fellow code to print point counterclockwise.

// Print Result

p = l;

do {

cout << "(" << points[p].x << ", " << points[p].y << ")\n";

p = next[p];

} while (p != l);

• Reply • Share ›

Abhay Doke • 7 months ago

orientation(p,q,r) should be counterclockwise.. You have written orientation(p,r,q) in the instructions. Please take a look and correct it.

• Reply • Share ›

Deepesh Maheshwari • a year ago

q = (p+1)%n;

Why this is used to get next point, didn't get the logic behind it

• Reply • Share ›

FallingStar ↗ Deepesh Maheshwari • 3 months ago

% is the modulo operator. (p+1)%n means (p+1) mod n. With p < n-1 it gives p+1, whereas p = n-1 it gives 0.

• Reply • Share ›

Ram ➚ Deepesh Maheshwari • a year ago

if the left most x point, i.e. 'p' in the example is the last element of the array, to get the next point, i.e. the 0th element, we need to do (p+1)%n which will return 0.

• Reply • Share ›

ggm8 • a year ago

This doesn't output them in counter-clockwise order... It just finds them in a random order.

3 • Reply • Share ›

Kartik ➚ ggm8 • 3 months ago

We have updated the program so that all points are printed in order.

• Reply • Share ›

poname • a year ago

not works well for
9 points
x y
0 0
0 1
1 0
0 2
2 0
0 3
3 0
1 2
2 1

output must be 9 but gives 6

2 • Reply • Share ›

Anubhav Joshi ➚ poname • 6 months ago

if u join the points given in the output, you'll get the convex hull.
i think the algo is correct as it gives the points which on joining will give the hull.

I think this small addition http://ideone.com/sdP1QR resolves the issue.

• Reply • Share ›

mayank • 2 years ago

in the line "... point is q if for any other point r, we have “orientation(p, r, q) = counterclockwise. Following is the detailed algorithm....." it should be..."orientation(p, q, r) = counterclockwise"...

@admin please check this

1 • Reply • Share ›

Deepesh Maheshwari mayank • a year ago

It is p,r,q - I also confused with it.
just run the program in debug mode, you will get the explanation

• Reply • Share ›

michael_skynet • 2 years ago

Thanks, helped me a lot :)

• Reply • Share ›

Tyler Johnsson • 2 years ago

Fantastic article, thank you very much!
I just wanted to share that I spent a good few hours tearing my hair over why it only worked "sometimes". Then I realized that it had to be p = l (lima) and not p = 1 (one) as I originally thought, haha. Thank you again!

1 • Reply • Share ›

venkat • 2 years ago

for points

• Reply • Share ›

Manish Kumar • 3 years ago

There is one more error. The for loop in both cases should start from 0 and not from 1.

• Reply • Share ›

GeeksforGeeks Manish Kumar • 3 years ago

@Manish Kumar: Thanks for your inputs. We have updated the second loop. The first loop looks fine though, it's a typical way to find min value in an array.

1 • Reply • Share ›

vsethuooo • 3 years ago

Inside the convexHull(..) function,
inside the do while loop,
why do you use, [script]for(i=1;i<n;i++)[script]="" instead="" of="" [script]for(i="0;i<n;i++)[/script]" ??="" pls="" explain="">

• Reply • Share ›

sumit1294 • 3 years ago

Time complexity of the given algo. must be O(m x n) not O(m x m) as mentioned above.

1 • Reply • Share ›

GeeksforGeeks sumit1294 • 3 years ago

Thanks for pointing this out. This was a typo. We have corrected it. Keep it