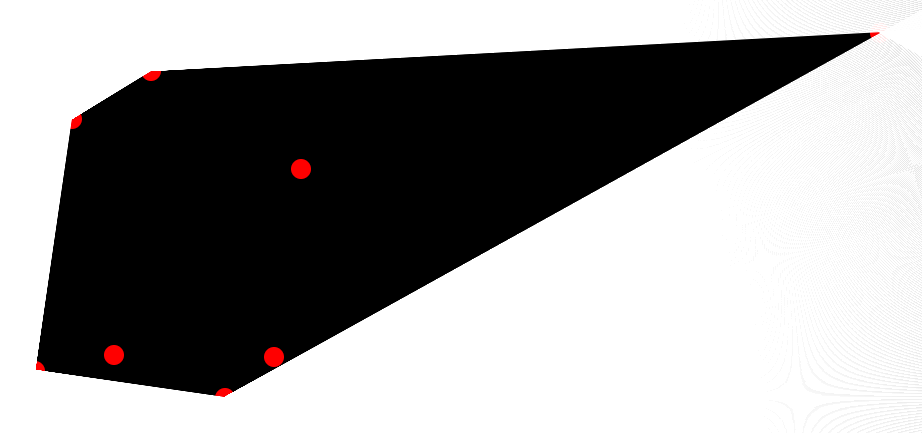
**ALGORITHMS ASSIGNMENT**



Convex Hull

R.P. Nihal - 106112067

Ashutosh N. Agarwal - 106112015

Cse 2nd Year

Nitt

**CONVEX HULL**

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.

convexHull

**USING GRAHAM SCAN ALGORITHM TO FORM THE CONVEX HULL**

Let points[0..n-1] be the input array.

1) Find the bottom-most point by comparing y coordinate of all points. If there are two points with same y value, then the point with smaller x coordinate value is considered. Put the bottom-most point at first position.

2) Consider the remaining n-1 points and sort them by polar angle in counterclockwise order around points[0]. If polar angle of two points is same, then put the nearest point first.

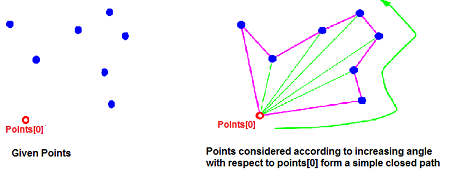
3) Create an empty stack ‘S’ and push points[0], points[1] and points[2] to S.

4) Process remaining n-3 points one by one. Do following for every point ‘points[i]‘  
 4.1) Keep removing points from stack while orientation of following 3 points is not counterclockwise or they don’t make a left turn.  
 a) Point next to top in stack  
 b) Point at the top of stack  
 c) points[i]

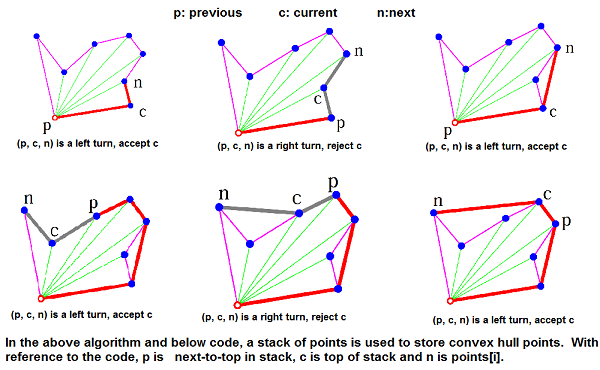
4.2) Push points[i] to S

5) Print contents of S.

**Step 1:-**



**Step 2:-**



*CODE*

*1*

#include<iostream>

using namespace std;

#include<stdlib.h>

#include<stack>

#include<time.h>

struct Point

{

int x;

int y;

};

Point p0;

void swap(Point &p1, Point &p2)

{

Point temp=p1;

p1=p2;

p2=temp;

}

//Returns the square of the distance between the two points

int dist(Point p1, Point p2)

{

return (p1.x - p2.x)\*(p1.x - p2.x) + (p1.y - p2.y)\*(p1.y - p2.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;

return (val>0) ? 1 : 2;

}

//Sort an array of points with respect to the first point

int compare(const void \*vp1,const void \*vp2)

{

Point \*p1 = (Point \*)vp1;

Point \*p2 = (Point \*)vp2;

int o = orientation(p0, \*p1, \*p2);

if (o == 0)

return (dist(p0, \*p2) >= dist(p0, \*p1))? -1 :1;

return (o == 2)? -1: 1;

}

Point nextToTop(stack<Point> &S)

{

Point p = S.top();

S.pop();

Point res = S.top();

S.push(p);

return res;

}

void convexHull(Point points[],int n)

{

int ymin=points[0].y, min=0;

for(int i=0;i<n;i++)

{

int y=points[i].y;

if(y<ymin || (y==ymin && points[i].x<points[min].x))

ymin=points[i].y, min=i;

}

swap(points[0], points[min]);

p0=points[0];

/\*

void qsort (void\* base, size\_t num, size\_t size, int (\*compar)(const void\*,const void\*));

-->Sorts the num elements of the array pointed by base, each element size bytes long, using the compar function to determine the order.

\*/

qsort(&points[1], n-1, sizeof(Point), compare);

stack<Point> S;

S.push(points[0]);

S.push(points[1]);

S.push(points[2]);

for(int i=3;i<n;i++)

{

while(orientation(nextToTop(S),S.top(),points[i])!=2)

S.pop();

S.push(points[i]);

}

while(!S.empty())

{

Point p=S.top();

cout<< "(" << p.x << ", " << p.y << ")" << endl;

S.pop();

}

}

int main()

{

int n;

cout<<"Enter the number of points you want to generate:";

cin>>n;

Point points[n];

srand( (unsigned)time(NULL) );

for(int i=0;i<n;i++)

{

points[i].x = rand() % 100 + 1;

points[i].y = rand() % 100 + 1;

cout<<"("<<points[i].x<<","<<points[i].y<<")";

if(i==(n-1))

cout<<endl;

}

int m = sizeof(points)/sizeof(points[0]);

convexHull(points, m);

return 0;

}

*CODE*

*2*

<html>

<head>

<!--Convex Hull-->

<!--30 April 2014...18:15-->

</head>

<style>

\* {

margin: 0px;

padding: 0px;

}

#canvas {

border: 2px solid black;

border-radius: 10px;

background-color: #000000;

}

body {

background-color: #222222;

}

#left {

position: absolute;

}

#right{

position: absolute;

}

</style>

<body>

<font color="white">

<div id="left"></div>

<div id="right"></div>

</font>

<canvas id="canvas"></canvas>

<script>

var canvas = document.getElementById('canvas'),

ctx = canvas.getContext('2d');

canvas.width = 0.8 \* window.innerWidth;

canvas.height = 0.8 \* window.innerHeight;

canvas.style.position = "absolute";

canvas.style.left = (window.innerWidth-canvas.width)/2;

canvas.style.top = (window.innerHeight-canvas.height)/2;

document.getElementById("left").style.left = 0;

document.getElementById("right").style.left = window.innerWidth - 55;

var num\_points = 8;

var points = [];

var rpoints = [];

for(var i=0;i<num\_points;i++) {

var ranx = Math.floor(Math.random()\*(canvas.width-200)+100);

var rany = Math.floor(Math.random()\*(canvas.height-100)+50);

points.push(new point(ranx,rany,10));

rpoints.push(new point(ranx,rany,10));

//alert(ranx+" "+rany);

}

/\*

points.push(new point(0,3,2));

points.push(new point(1,1,2));

points.push(new point(2,2,2));

points.push(new point(4,4,2));

points.push(new point(0,0,2));

points.push(new point(1,2,2));

points.push(new point(3,1,2));

points.push(new point(1,3,2));

rpoints.push(new point(0,3,2));

rpoints.push(new point(1,1,2));

rpoints.push(new point(2,2,2));

rpoints.push(new point(4,4,2));

rpoints.push(new point(0,0,2));

rpoints.push(new point(1,2,2));

rpoints.push(new point(3,1,2));

rpoints.push(new point(1,3,2));

\*/

function point(x,y,r) {

this.x = x;

this.y = y;

this.r = r;

}

for(var i=0;i<num\_points;i++)

document.getElementById("left").innerHTML = document.getElementById("left").innerHTML + points[i].x + "," + points[i].y+"<BR>";

for(var i=0;i<num\_points-1;i++)

for(var j=0;j<num\_points-1-i;j++)

if(points[j].x > points[j+1].x) {

var k = points[j];

points[j] = points[j+1];

points[j+1] = k;

}

for(var i=0;i<num\_points-1;i++)

for(var j=0;j<num\_points-1-i;j++)

if(rpoints[j].x > rpoints[j+1].x) {

var k = rpoints[j];

rpoints[j] = rpoints[j+1];

rpoints[j+1] = k;

}

window.onload = function disp\_points() {

for(var i=0;i<points.length;i++) {

ctx.beginPath();

ctx.arc(points[i].x,points[i].y,points[i].r,0,2\*Math.PI);

ctx.fillStyle = "rgba(255,0,0,1)";

ctx.fill();

ctx.closePath();

}

find\_initial();

}

var fpoints = [];

var move1 = 180;

var mover1 = 270;

var move2 = 360;

var mover2 = 270;

var hori = 0;

var p1;

var p2;

var num = 0.05;

var end = 0;

function check0(a,x,y) {

for(var i=0;i<rpoints.length;i++) {

var r1 = (-1\*y-(-1)\*rpoints[i].y)/(x-rpoints[i].x);

var r2 = Math.atan(r1)\*180/Math.PI;

if(r2<0) r2 += 360;

else r2 += 180;

//console.log(r2+" "+a);

if(r2 <= a+0.05 && r2 >= a-0.05)

{p1=p2=i; fpoints.push(points[i]); rpoints[i].x=-1; rpoints[i].y=-1; return i;}

}

return (-1);

}

function check1(a,x,y) {

for(var i=0;i<rpoints.length;i++)

if(rpoints[i].x != -1) {

var r1 = (-1\*y-(-1)\*rpoints[i].y)/(x-rpoints[i].x);

var r2 = Math.atan(r1)\*180/Math.PI;

if(r2<0) r2 += 360;

else r2 += 180;

//console.log(r2+" "+a);

if(r2 <= a+0.05 && r2 >= a-0.05)

{p1=i; fpoints.push(points[i]); rpoints[i].x=-1; rpoints[i].y=-1; return i;}

}

var i=p2;

var r1 = (-1\*y-(-1)\*points[i].y)/(x-points[i].x);

var r2 = Math.atan(r1)\*180/Math.PI;

if(r2<0) r2 += 360;

else r2 += 180;

if(r2 <= a+0.05 && r2 >= a-0.05 && end==2)

end = 1;

return (-1);

}

function check2(a,x,y) {

for(var i=0;i<rpoints.length;i++)

if(rpoints[i].x != -1) {

var r1 = (-1\*y-(-1)\*rpoints[i].y)/(x-rpoints[i].x);

var r2 = Math.atan(r1)\*180/Math.PI;

if(r2<0) r2 += 360;

else r2 += 180;

//console.log(r2+" "+a);

if(r2 <= a+0.05 && r2 >= a-0.05)

{p2=i; fpoints.push(points[i]); rpoints[i].x=-1; rpoints[i].y=-1; return i;}

}

var i=p1;

var r1 = (-1\*y-(-1)\*points[i].y)/(x-points[i].x);

var r2 = Math.atan(r1)\*180/Math.PI;

if(r2<0) r2 += 360;

else r2 += 180;

if(r2 <= a+0.05 && r2 >= a-0.05 && end==2)

end = 1;

return (-1);

}

function find\_initial() {

ctx.fillStyle = "white";

ctx.fillRect(hori,0,1,canvas.height);

hori += 1;

var ang = check0(mover1,hori,0);

if(ang == (-1))

setTimeout(function(){find\_initial();},10);

else begin\_find();

}

function begin\_find() {

ctx.save();

ctx.translate(points[p1].x,points[p1].y);

ctx.rotate(move1/180\*Math.PI);

ctx.fillStyle = "white";

ctx.fillRect(0,0,1,2000);

move1 += num;

mover1 -= num;

if(mover1<=180) {

mover1 = 360;

end = 2;

}

check1(mover1,points[p1].x,points[p1].y);

ctx.restore();

ctx.save();

ctx.translate(points[p2].x,points[p2].y);

ctx.rotate(move2/180\*Math.PI);

ctx.fillStyle = "white";

ctx.fillRect(0,0,1,2000);

move2 -= num;

mover2 += num;

if(mover2>=360)

mover2 = 180;

check2(mover2,points[p2].x,points[p2].y);

ctx.restore();

if(end != 1)

setTimeout(function(){begin\_find();},1);

else {

for(var i=0;i<fpoints.length;i++)

document.getElementById("right").innerHTML = document.getElementById("right").innerHTML + fpoints[i].x + "," + fpoints[i].y+"<BR>";

}

}

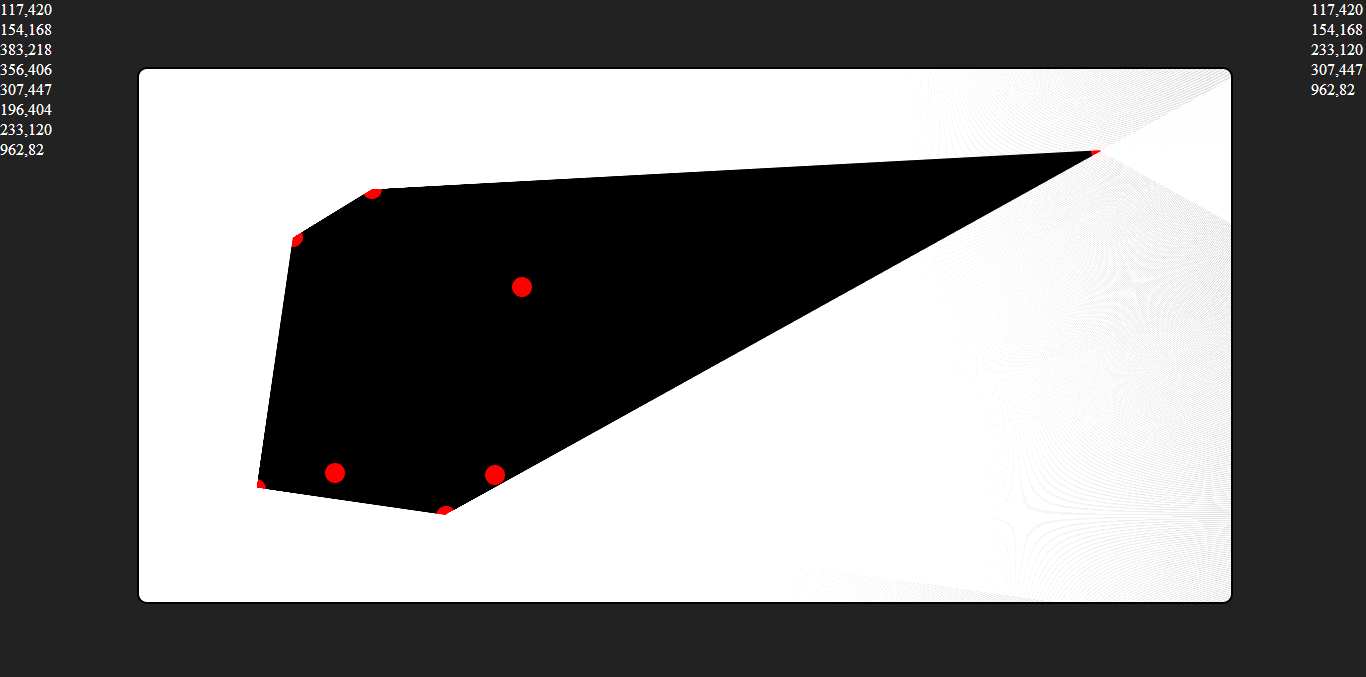
</script>

</body>

</html>

*SNAPSHOTS*



**

*THANK*

*YOU*