CS 133 Assignment 2

Sungho Ahn

862026328

Spring 2019

04/23/2019

1. Given a list of points that come in either clockwise or counter-clockwise, it is important to check their direction in order to test whether these points form a convex hull or not. For example, if first two points have a CCW orientation then it means they have left direction. Now, in order to fully form a convex hull, the given list of points must maintain this direction until the end of list is encountered. To illustrate some code for this algorithm:

// Create a Point class with 2 point attributes x and y

Class Point {

Public: double x, y;

};

// Helper function that returns sorted direction (Col, CW, CCW)

Int Direction(Point p1, Point p2, Point p3) {  
 int dir = (p2.y - p1.y) \* (p3.x - p2.x) – (p2.x - p1.x) \* (p3.y - p2.y);

If (dir == 0) return 0; // It is collinear

else if (dir > 0) return 1; // It is clockwise

else return 2; // It is counter-clockwise

}

int main() {

Point Points[] = {{0, 0}, {1, 0}, {2, 2}} // Set with points in Examples

int num\_points = sizeof(Points) / sizeof(Points[0]); // Number of points

int temp = Direction(Points[0], Points[1], Points[2]); // Dir of first 3 points

bool result = true; // bool checker

for (int i = 2; I < num\_points, i++) {

int temp2 = Direction(Points[i-2], Points[i-1], Points[i]);

if (temp != temp2) {

result = false;

break;

}

}

return result;

}

This will output true if convex hull is formed with given list of points, and false if it does not form a convex hull, meaning the direction is changed somewhere during the iteration.

2. **Given:** A set of points P and a straight line p1p2

**Claim:** Point pi in P that is farthest away from the line p1p2 is part of convex hull of P

**Claim 2:** Point pi in P is part of convex hull of P if there exists a line through it for which all points in P are on the same side of the line.

**Proof:** Assume that a point with minimum y-coordinate is strictly a part of convex hull. Then consider a point pi that is farthest away from the line p1p2. Try rotate the coordinate axis to make the point pi has minimum y-coordinate. Then consider a line through this point pi that is parallel to x-axis. From Claim 2, we see that all other points in P are on the same side of the line that goes through pi. Hence, pi is part of convex hull of P and we have proved the Claim.

3. **Given:** A set of points P and the farthest pair of points p1 and p2

**Claim:** The farthest pair of points p1 and p2 in P are both on the convex hull of P

**Claim 2:** Point pi in P is part of convex hull of P if there exists a line through it for which all points in P are on the same side of the line.

**Proof:** Assume that a point with minimum y-coordinate is strictly a part of convex hull. Then consider two points p1 and p2 that has the largest distance among the points in P. Then for each of these two points, try rotate the coordinate axis to make the point has minimum y-coordinate. Then consider a line through each of these points that is parallel to x-axis. From Claim 2, we see that all other points in P are on the same side of the line that goes through this point. Thus, the proof holds for the lines perpendicular to the points p1 and p2, through points p1 and p2.Hence, we have proved that the farthest pair of points p1 and p2 are both indeed on the convex hull of P.

4. A worst-case scenario of the Quick Hull algorithm takes O(n2) time complexity. First of all, the quickhull algorithm is based on the divide-and-conquer strategy and thus, the speed of the algorithm is highly dependent on the size of sub-problems. It needs to divide the problem into 2 smaller sub-problems of similar size. If they are not balanced and hence the number of elements divided into each sub-problem differs greatly, it will case recursion of N times. To illustrate, if that is the case, the problem size will be reduced by a constant instead of fraction of the main problem size. In conclusion, in order to craft a worst-case of quickhull algorithm, the input points need to be balanced in distribution.