CSE355

Zhiming Fan

October 2018

1.

We can prove this question by contradiction. First of all, we find a point that has the smallest X-coordinates in the set of S and call it point G.We just suppose point G is not on convex hull, therefore it must be inside the convex hull boundary. However, the smallest X-coordinate of point G will be contradictory to our prerequisite because there will be another point that has smaller X-coordiantes than point G. We can realize this contradiction.

By the way, we also need to mention the equition to compute distance between two points is $\sqrt{(x-x_0)^2+(y-y_0)^2}$ in a two dimensional coordinate system. We can easily realize that, to obtain the longest distance between points in a set, we must make sure the difference of $x-x_0$ and difference of $y-y_0$ as big as they can. Thus we consider what makes the difference longer. In the other words, the points that have extremely minimal and maximum coordinates will be considered to find the longest distance. Those extreme points are all on the convex hull as well(We proved it by contradiction in paragraph 1).

Here we assume that the two vertices are not on the convex hull, we can rotate the circle around the midpoint 360 degree. We can find some points are not covered inside the circle due to the short diameter. To cover more points, we have to extend the diameter so the final outcome will be the two vertices are on the convex hull.This contradicts to our initial assumption.

So we can conclude the diameter of a set of points S is connected between two vertices on the convex hull.

2.

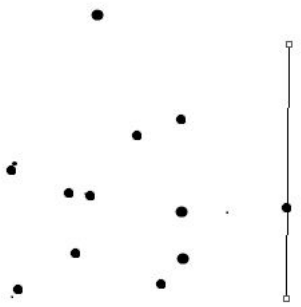
a).

For incremental construction, we are going to compare the length of distance to each collinear point. If the new added point donates a larger covered area than previous collinear point, we just delete all the connections to previous point and build new polygon for new added point. Once we have such collinear problem, we just compare to previous point and use this method to handle this. Since previous comparision has deleted the point that that donates smaller area, we only need to compare next collinear point by comparing one time. We also don't care the order of appearing collinear points because we deleted the point that we don't want.

For example,if the new added point donate area and it completely cover the previous area:

```
while(remainPoints!=0){
        if(previousPoint.isCollinear(currentPoint)&&currentPoint.donatedArea>previousPoint.donatedArea){
                delete(previousPoint);
                add(currentPoint);
        }
        remainPorints-=1;
}
```

However, if the intersection area is not completely covered. We just add all three points to the convex hull.
b).
For Graham scan, we must observe the orientation of current vertice.Before encounter the collinear three points, we set up tangent points to determine the orientation of tangent line. If the line between current and previous line are concave, we may delete previous and current line and try the next point.
3.
a).
.
Suppose the two vertices are not on the convex hull.The line of two vetices will split the set of points into two parts. Apparently, it contradicts with initial algorithm.
b)



Firstly, we can find those extreme points with minimal and maximum X-coordinate. Pick one of vertices of them and draw a vertical through it, and let the line rotate clockwisely. It will stop once it firstly touches another vertice. Following the algorithm we can get the required line L, which satisfies the two requisites: has all the points of a given set to one side; miniminzes the max of the perpendicular distances of the points of L.
4.
a).
Since we already have the polygon, we can triangulate this polygon very fast in time complexity O(n). Then we start the point with highest Y-coordinates. Check if the line connecting current vertice and next vertice,and the the line connecting current vertice and previous vertice forms a acute angle or right angle, then we can confirm it is on convex hull. Otherwise, we indicate it as useless point and skip this point to jump next vertice. We finally traverse the polygon

with time complexity O(n) and generate its convex hull.