Computational Geometry

Spring 2019-2020

Programming Project 1

Participants

The current project was written by the following:

- Lekkas Nikolaos - 1115201600089
- Lykos Emmanouil - 1115201600096

Manual

In order to run the exercises you should run in sage the .sage files or the .py files that were automatically produced running the .sage files. The code of .sage files is similar to the .python files.

In order to run the exercises just write on the terminal the command

```
sage exercise_{exercise_num}.sage
```

with **exercise_num={1,2,3,4}**.

Furthermore, while running the last two exercises, two image files will be produced. The first one will produce a scatter plot of the input points, while the second one will produce the original scatter plot but there will drawn lines which represent the edges of the convex hull in 2 dimensions.

Exercise 1

Implement an algorithm that takes as input three points in the plane. checks that they form a triangle and whether the interior of the triangle contains the origin (0, 0) or not.

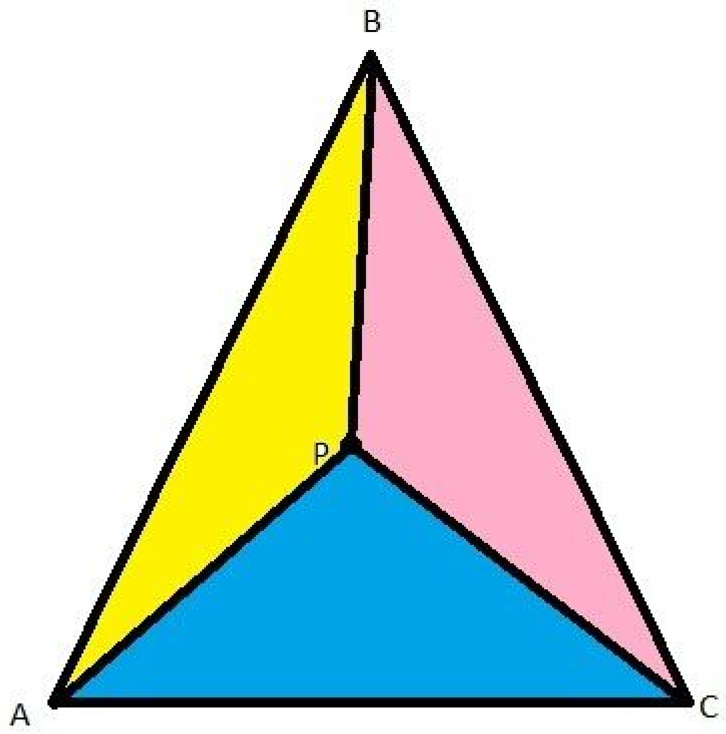
In this algorithm we give 3 points(A, B, C) and these points depict our triangle(ABC).

Initially, we check if the given points are forming a triangle by applying the CCW predicate to the three points given. The CCW predicate just takes the cross product between vector AB and vector AC and checks the third coordinate of the cross product in order to find the rotation of the cross product, consequently the rotation of vector AB relative to vector AC. In fact, if the third coordinate of the cross product is 0 then the 3 given points are collinear, therefore they do not form a triangle, else they form the triangle.

We give another point(P) and we check if this point is inside our triangle. To find that out we calculate the area of our first triangle(ABC). To do that we use the triangle area formula which is:

$$\left| \frac{A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)}{2} \right|$$

Now if the point is inside the triangle then all the areas of the internal triangles, created by our P point, summed will output the area of the first triangle(ABC)
In other words the areas of ABP + BCP + ACP will be equal to the area of ABC.



Algorithm: return (area(ABC) == area(PAB)+area(PBC)+area(PAC))
A second way to determine if the given point (P_x, P_y) lies inside the triangle is to check if that given point is a convex combination of the three triangle points given. Therefore, the following system should be solved:

$$\begin{cases} l_1(A_x, A_y) + l_2(B_x, B_y) + l_3(C_x, C_y) = (P_x, P_y) \\ l_1 + l_2 + l_3 = 1 \\ l_1 \geq 0,\ l_2 \geq 0,\ l3 \geq 0 \end{cases} \Leftrightarrow \begin{cases} l_1 A_x + l_2 B_x + l_3 C_x = P_x \\ l_1 A_y + l_2 B_y + l_3 C_y = P_y \\ l_1 + l_2 + l_3 = 1 \\ l_1 \geq 0,\ l_2 \geq 0,\ l3 \geq 0 \end{cases}$$

In order to solve this system we pass this system into the *solve* function of sage and if there is a solution of the aforementioned system, then the given point(in this case, the (0,0)) lies inside the given triangle.

Exercise 2

Given a circle of radius r in the plane with (0, 0) as center, implement an algorithm that finds the total lattice points on the circumference. Lattice Points are points with integer coordinates.

We create a circle giving its radius. The radius should be ≥ 0.
We search for all the points inside the circle where it's coordinates are integers. To check that we search for every integer from 0 to r(radius) (x_{new}), we find it's equivalent y coordinate in the circle with the circles formula: $x^2 + y^2 = r^2$. This y coordinate should be an integer and that's why we take the integer value of square root of y^2 (y_{new}).
Then we check if y_{new} and x_{new} is inside the circle with $x_{new}^2 + y_{new}^2 = r^2$ and if that is true then we add 4 to the result because of all the possible coordinates in the circle.($-x_{new}$, $-y_{new}$), ($-x_{new}$, y_{new}), (x_{new} , $-y_{new}$), (x_{new} , y_{new})

```
Algorithm:
result <- 0
for x in (0 - r)
    y <- Find the integer value of y from $\sqrt{y^2} = r^2 - x^2$
    if $x^2 + y^2 = r^2$ then
        result =+ 4
return result
```

Exercise 3

Implement the incremental 2D algorithm for computing the convex hull of a finite set of points in the plane.

As we know an incremental convex hull algorithm has the following form:

1. Begin from a simple convex hull(e.g. a triangle),
2. Incrementally add a new point to convex hull at each iteration.

Therefore, following this form we wrote the **2D Beyond-Beneath** algorithm because this was the one of the two algorithms that were taught in this class(the other one is the gift-wrapping algorithm), so we thought that it was asked to implement this algorithm.

Note that, that we assumed that the first three points that form the initial convex hull are not collinear(so, they form a triangle) and there are no points with the same x-coordinate, in order to be easier for us to implement our algorithm and did not become chaotic by the many corner cases.

The algorithm is the following:

```
1) Sort the given points by decreasing order on the x-coordinate. Let
   the sorted points be the $p_1, p_2, \ldots, p_n$.
2) Insert in the convex hull ($CH$) the edges that are formed from the
   first three points. $CH \leftarrow CH \cup \{(p_0,p_1),(p_1,p_2),(p_3,p_0)\}$
3) $previousPoint \leftarrow p_2$
4) For i in 4...n:
   a) It is obvious that the current point is outside the convex hull,
      so we will take the edges that are tangent to the previous
      point.
   b) We will find one **red** edge $RE$ tangent to the previous point. (I
      will explain the coloring later).
   c) $EdgesToExamine = [(None, RE)]$
   d) while $EdgesToExamine \neq [\ ]$
      i)   $prevRedEdge,\ currEdge = EdgesToExamine.head$ (deletes the
           first element of the list)
      ii)  Set $InternalCHPoint$ a point that is inside the convex hull
           and it is not on the current edge.
      iii) $currEdgeColor = findColor(currEdge,\ p_i,\ InternalCHPoint)$
      iv)  if $currEdgeColor = red$ then
           (1) $CH = CH - \{currEdge\}$
           (2) Find all the neighbors $n_k$ of the convex hull
           (3) $EdgesToExamine = EdgesToExamine + [(currEdge, n_k), \forall k = 1, \ldots, \#\ neighbors]$
      v)   if $currEdgeColor = blue$ then
           (1) $intersection = findIntersection(currEdge, prevRedEdge)$
           (2) $CH = CH \cup \{(currEdge, intersection)\}$
      vi)  $return\ CH$
```

.

The coloring of the edges happens as follows. Firstly, we search for a point $\left(Q_1, Q_2\right)$ that is inside the current convex hull and not on the current edge (A, B) where $A=\left(A_x, A_y\right)$, $B=\left(B_x, B_y\right)$ and we check if the convex hull point and the new point $P=\left(P_1, P_2\right)$ are on the same half space relative to the current edge. If they're on the same halfspace the current edge is **blue,** else it is **red.** Mathematically, we check if the sign of the following determinants* is the same. If they have the same sign, then the two points belong to the same halfspace, thus edge is blue, else they belong to different halfspace, therefore the edge is red.

*The arrays that we take theirs determinant sign are the following:

$$\begin{bmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ P_x & P_y & 1 \end{bmatrix}, \begin{bmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ Q_x & Q_y & 1 \end{bmatrix}$$

Exercise 4

Implement the gift wrap algorithm for computing the convex hull of a finite set of points in the plane .

The algorithm of this exercise is known and straightforward. The pseudocode is the following

```
1. Find the leftmost point $p$ in set of points $S$.
2. Initialize convex hull $CH=[p]$.
3. $currVertex=p$
4. $nextVertex=p$.
5. For each point $p_{curr}$ in $S$
    a. if $CW\left(currVertex, nextVertex, p_{curr}\right)$ then
        i. $nextVertex = p_{curr}$
```

   b. else if the points are collinear then $nextVertex$ takes the value of t$nextVertex = p$ he most far vertex from $currPoint$.

6. if then terminate the algorithm and return CH,
   else $S = S - \{nextVertex\}$, $currVertex = nextVertex,\ CH = CH + [nextVertex]$ and proceed to step 5.

Note that the predicate CW is just the orientation predicate we mentioned at **exercise 1**, but in order to be true the 3rd coordinate of the cross product of the two vectors should be negative. That means the orientation should be counter clockwise.