MAC0331 - Lista 1

Matheus T. de Laurentys, 9793714

May 5, 2020

Q 2:

```
1  function "point.<"(this, other):
2          if (this.y == other.y): return this.x < other.x
3          return this.y < other.y
4
5  function binary_search (arr, min, max, pt):
6          if min > max: return false
7          mid = min/2 + max/2
8          if arr[mid] < pt: return binary_search(arr, mid + 1, last, pt)
9          else if pt < arr[mid]: return binary_search(arr, min, mid − 1, pt)
10         else: return true
11
12 function belong_polygon (n, P, k, e, d, q):
13         return binary_search (e, 0, k−1, q) or binary_search (d, 0, n−k+1, q)
```

Q 7:

a): At the moment of line five's execution, it is known that there is one extreme point above the point X and one below or at the same "height". What is left to know is if this segment is to the right of the point. Instead of checking if the segment will intersect with a possible ray crossing from X towards $+\inf$, one could simply check if X is to the left of the segment. It happens that we can use the function at_left created early on in the class to decide that.

b): The problem that version one of the algorithm fails to take into account is if the point is colinear with any of the segments. While the form version two manages this problem is nice, it would be possible to add an extra check to line 6. That is, instead of just checking for proper intersection, check for colinearity too.