

Homework #1

Lucio Franco, worked with Sam Windham.

10/17/2017

1)

```
PROCEDURE LeftTangent(Polygon p, Point q):
    // first point is left tangent
    if isLeft(q, p[n-1], p[0]) AND !isLeft(q, p[0], p[1]):
        return (q, p[0])

    let i = 0
    let offset = 1
    loop:
        // direction of current edge
        let edgeDir = isLeft(q, p[i], p[i+1])

        // current edge is the left tangent point
        if isLeft(q, p[i-1], p[i]) AND !edgeDir:
            return (q, p[i])

        // check directions and continue search
        let prevEdgeDir = isLeft(q, p[i - offset], p[i - offset + 1])

        // current edge is left
        if edgeDir:
            // previous edge is left AND current point left of previous point
            if prevEdgeDir AND !isLeft(q, p[i-offset], p[i]):
                i -= offset
                offset = 1
            // prev edge is right OR current edge left of previous edge
            else:
                offset *= 2
        else:
            // previous edge is left
            if prevEdgeDir:
                i -= offset
                offset = 1
```

```

else:
    offset *= 2

return i = min(i + offset, n - 1)

```

The algorithm will run an exponential search to find the tangent. It will return the line segment that forms the left tangent on P. The `isLeft(a, b, c)` will check if the three vectors create a left turn. Exponential search runs in $O(\log n)$.

3)

We know that there are $\binom{n}{2}$ lines induced from a set of n points in general position. If we transform every point to the dual, the lines produced will intersect forming the point that represents the line between those two points that are now lines in the dual. We know that if the point formed from the intersection of the two lines is on the positive side of the dual, aka it has a x value greater than zero, it's line segment in the primal has a positive slope.

From this we can run a plane sweep starting at the origin and sweeping along the x axis of the dual. Each time we encounter an intersection of two lines we can count this as a positive slope. There are also four possible combinations of the lines, when at the origin one can tell if both lines have positive slope, the line above has positive slope and the line below has negative slope, the line above has negative slope and the line below has positive slope, or they both have negative slopes.

From this we can eliminate possible events because for a fact we know that those two lines can not intersect as we move to the right when the line above has a positive slope and the line below has a negative slope. For part (a) we would continue the line sweep until we've gone through all of the intersections but for part (b) instead of starting at the origin, we would start at m' and continue until we have hit m'' . Once an event has been checked, the lines can be swapped and checked against their neighbors.

Since as we move to the right the slope of the line produced from the intersection in the dual will have a larger slope. Every intersection found within this bound will be counted as a positive slope. A brute force algorithm would take $\theta(\binom{n}{2})$ but with the plane sweep and the elimination of events we can reduce this. We know that for n points, there can be k induced lines with positive slope, with a max of $\binom{n}{2}$. Each event we face takes $O(\log n)$ using a balanced binary tree. To initially build the status it takes $O(n \log n)$. Therefore the runtime will be $O(n \log n) + O(k \log n) = O((n + k) \log n)$.

4)

Let $R(\alpha)$ be the ray from p with the angle α . Now we can create a schedule with each endpoint sorted on α . Now a min-heap is created to store the visible elements, first we add the first event to the min-heap. Now on the other events either a segment is removed or is added. The heap is sorted on the minimum distance from p . Since we know that the segments are disjoint, the order in the heap will not change.

Therefore when process and event, there are two possibilities, the segment that belongs to the event is in the heap or its not. If it is in the heap we remove it, if it is not in the heap we can add that segment. At the end of each heap operation, the heap is peaked in $O(1)$ and that segment is reported as visible.

There are a possible $2n$ events because each segment has two endpoints. At each event there is either a heap insertion or a deletion which can be performed in $O(\log n)$, the minimum of the heap is also peaked in $O(1)$. Therefore the runtime is $O(n \log n) + O(n \log n) + O(n \log n) = O(n \log n)$. This is because we need to initially sort the endpoints, we need to process each in the schedule and to sort and remove duplicate visible segments, therefore the runtime would be $O(n \log n)$.

5)