

# Homework 1

Elliott Pryor

24 Jan 2021

**Problem 1**

---

Some sort of binary search.

Jointly search  $P$  and  $P'$  Pick  $a \in P$ ,  $b \in P'$  Want line with highest  $z$  intercept

Idea:  $O(n)$  run Graham's Scan on it. Already sorted so takes  $O(n)$  time.

Idea: Start at  $a$  compute point  $b \in P'$  tangent to  $a$  ( $O(\log n)$ ). Then reverse, find  $a$  tangent to  $b$ . Then forward, find  $b$  tangent to  $a$ . Then done????

---

**Algorithm 1** Tangent Function

---

```

1: function TANGENT( $a, P$ )
2:   Binary search to find point of tangency
3:    $low \leftarrow 0$ 
4:    $high \leftarrow |P|$ 
5:   while !found do
6:      $m \leftarrow \lceil (low + high)/2 \rceil$ 
7:      $line \leftarrow \overrightarrow{a, m}$ 
8:     if  $line([m+1]_x) > [m+1]_y$  and  $line([m-1]_x) > [m-1]_y$  then
9:       found it (is supporting line)
10:      return  $m$ 
11:     else if  $line([m+1]_x) > [m+1]_y$  then
12:       line intersects some point before  $m$  (tangent point to left)
13:        $high \leftarrow m - 1$ 
14:     else if  $line([m-1]_x) > [m-1]_y$  then
15:       line intersects some point after  $m$  (tangent point to right)
16:        $low \leftarrow m + 1$ 
17:     end if
18:   end while
19: end function

1: function UPPERTANGENT( $P, P'$ )
2:   Run Tangent 3 times to find upper tangent.
3:    $a \leftarrow |P|/2$  // Random point in  $P$ 
4:    $b \leftarrow Tangent(a, P')$  // So we can 'see'  $p_i$  from  $b$ 
5:    $a \leftarrow Tangent(b, P)$  // Finds  $p_i$ 
6:    $b \leftarrow Tangent(a, P')$  // Finds  $p_j$ 
7:   return  $\overrightarrow{a, b}$ 
8: end function

```

---

---

**Problem 2**


---

1. The points on the  $Pareto(P)$  fall on a series of horizontal and vertical line segments. It follows a line in 'Manhattan' distance. In order to make the analogous assertion, we define corner  $C(p)$   $p \in P$  as the region  $(x, y) \in \mathbb{R}^2$   $x \leq p_x, y \leq p_y$ .

Then a point  $p$  is on the Pareto of a set  $P$  if and only if there is no other corner  $C(p')$ ,  $p \neq p'$ , containing  $p$

2. This is almost identical to Graham's Scan. We replace the Orient() function with a different comparison. We simply compare the  $y$  values of adjacent points. Since the points are in sorted,  $x$ , order if a point  $p_i$  has a larger  $y$ -coordinate than  $p_{i-1}$  the  $p_{i-1}$  would be in  $C(p_i)$  so is not on the Pareto. Since this comparison only needs the first point in the stack, we also adjust initialization of  $S$  to only push  $p_1$  onto the stack, and to not terminate the while loop unless there are 0 points in the stack.

This has the same running time as Graham's Scan. Since after sorting, the algorithm takes a linear pass through all of the points. It also only pops a point at most once from the stack. The only difference with this algorithm from Graham's Scan is the Orient. Since comparing  $y$ -coordinates is also a constant time lookup, our running time is  $O(n \log(n))$  due to the sorting in line 2.

---

**Algorithm 2** Pareto Scan
 

---

```

1: function PARETOSCAN( $P$ )
2:   sort  $P$  by increasing  $x$  value
3:   push  $p_1$  onto stack  $S$ 
4:   for  $i \leftarrow 3, \dots, n$  do
5:     while  $|S| \geq 1$  and  $p_{i,y} \geq S[top]_y$  do
6:       pop  $S$ 
7:     end while
8:     push  $p_i$  onto  $S$ 
9:   end for
10: end function

```

---

3. Our modified Jarvis march algorithm would search for the point with the greatest  $y$  value. Then it would loop and search for the point with the next greatest  $y$ -coordinate to the right of the previous point ( $p_{i,x} > p_{i-1,x}$ ). This loop is repeated  $h - 1$  times (the first point was found in initialization step of finding point with largest  $y$ -coordinate).

---

**Algorithm 3** Jarvis Stairs

---

```
1: function JARVISSTAIRS( $P$ )
2:   find point  $p_1$  with largest y-coordinate
3:    $S \leftarrow p_1$ 
4:   for  $i \leftarrow 2, \dots, h$  do
5:     find point  $p_i$  with largest y-coordinate such that  $p_{i,x} > p_{i-1,x}$ 
6:   end for
7:   return  $S$ 
8: end function
```

---