- You have a 2D plane w/ n points represented as:

  points: list<x, y>

  To visit the next point, it takes 1 unit to either:

  1.) Move vertically one unit
  2.) Move horizontally one unit
  3.) Move horizontally and vertically one unit.

Input: (1,1), (3,4), (-1,0)

$\underline{(1,1) \to (2,2) \to (3,3)} \to \underline{(3,4)} \to \underline{(2,3) \to (1,2) \to (0,1) \to (-1,0)}$

Total steps: 7

✱ Subdivide: How to get $(1,1) \to (3,4)$?

Compute distances: $|<1,1> - <3,4>| = <2,3>$

Maximum # of steps is the largest number of result. $\underline{3}$

✱ How to get $(3,4) \to (-1,0)$?

$|<3,4> - <-1,0>| = <4,4>$. Max is $\underline{4}$

Answer would be 3 + 4 = 7

```
f ( points:  list< List< int>> ) → int
    start = list[0].itr
    while  start < list.size() - 1
        destination = *(start + 1)
        [ $x_1$ , $y_1$ ] = start
        [ $x_2$ , $y_2$ ] = destination
        result += max( abs( $x_1 - x_2$ ), abs( $y_1 - y_2$ ))
    return result
```

Assume result initialized to 0