# 1266. Minimum Time Visiting All Points

My Submissions (/contest/weekly-contest-164/problems/minimum-time-visiting-all-points/submissions/)

Back to Contest (/contest/weekly-contest-164/)

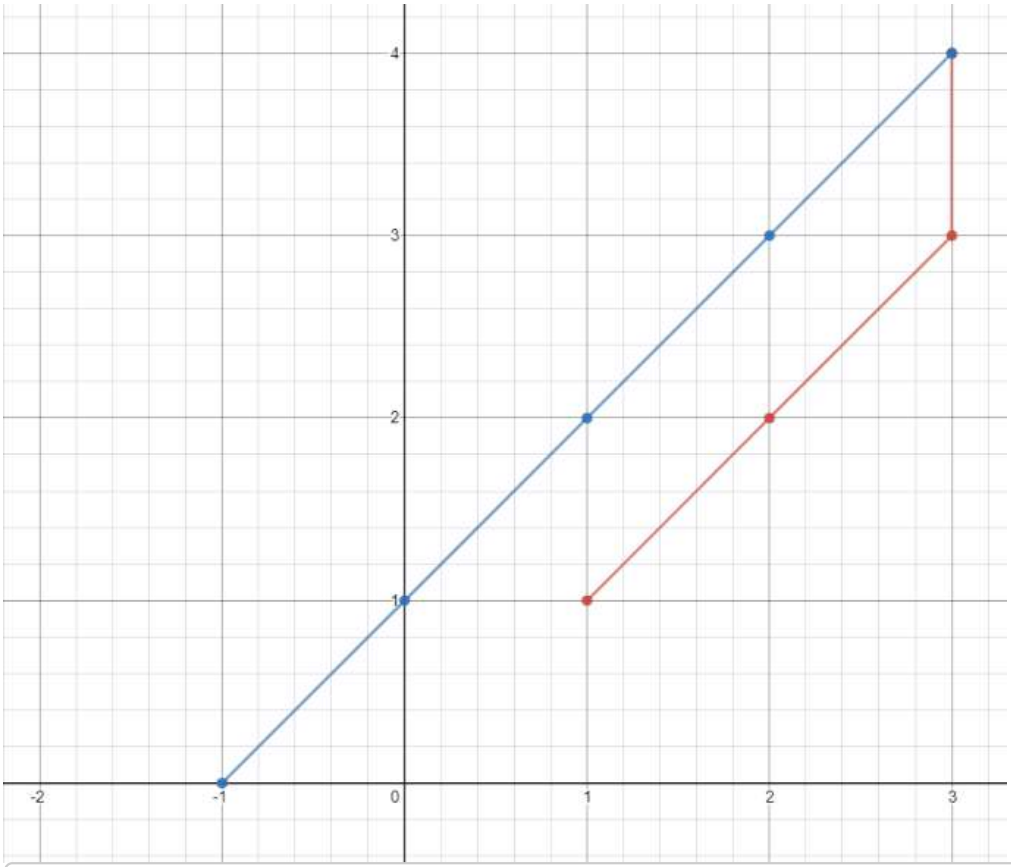
On a plane there are `n` points with integer coordinates `points[i] = [xi, yi]`. Your task is to find the minimum time in seconds to visit all points.

You can move according to the next rules:

- In one second always you can either move vertically, horizontally by one unit or diagonally (it means to move one unit vertically and one unit horizontally in one second).
- You have to visit the points in the same order as they appear in the array.

| | |
|---|---|
| User Accepted: | 2691 |
| User Tried: | 2850 |
| Total Accepted: | 2756 |
| Total Submissions: | 3548 |
| Difficulty: | Easy |

**Example 1:**




```
Input: points = [[1,1],[3,4],[-1,0]]
Output: 7
Explanation: One optimal path is [1,1] -> [2,2] -> [3,3] -> [3,4] -> [2,3] -> [1,2] -> [0,1] ->
Time from [1,1] to [3,4] = 3 seconds
Time from [3,4] to [-1,0] = 4 seconds
Total time = 7 seconds
```

**Example 2:**

```
Input: points = [[3,2],[-2,2]]
Output: 5
```

**Constraints:**

- `points.length == n`
- `1 <= n <= 100`
- `points[i].length == 2`
- `-1000 <= points[i][0], points[i][1] <= 1000`

Discuss (https://leetcode.com/problems/minimum-time-visiting-all-points/discuss)

Java

```java
class Solution {
    public int minTimeToVisitAllPoints(int[][] points) {
        int minTime = 0;
        if (points == null || points.length < 2 || points[0].length < 2) {
            return minTime;
        }
        int m = points.length;
        int n = points[0].length;
        for (int i = 0; i < points.length - 1; i++) {
            minTime += findMinTimeBetweenTwoPoint(points[i], points[i + 1]);
        }
        return minTime;
    }

    private int findMinTimeBetweenTwoPoint(int[] point1, int[] point2) {
        int x1 = point1[0];
        int y1 = point1[1];
        int x2 = point2[0];
        int y2 = point2[1];
        return Math.max(Math.abs(x2 - x1), Math.abs(y2 - y1));
    }
}
```

**Custom Testcase**

Run Submit