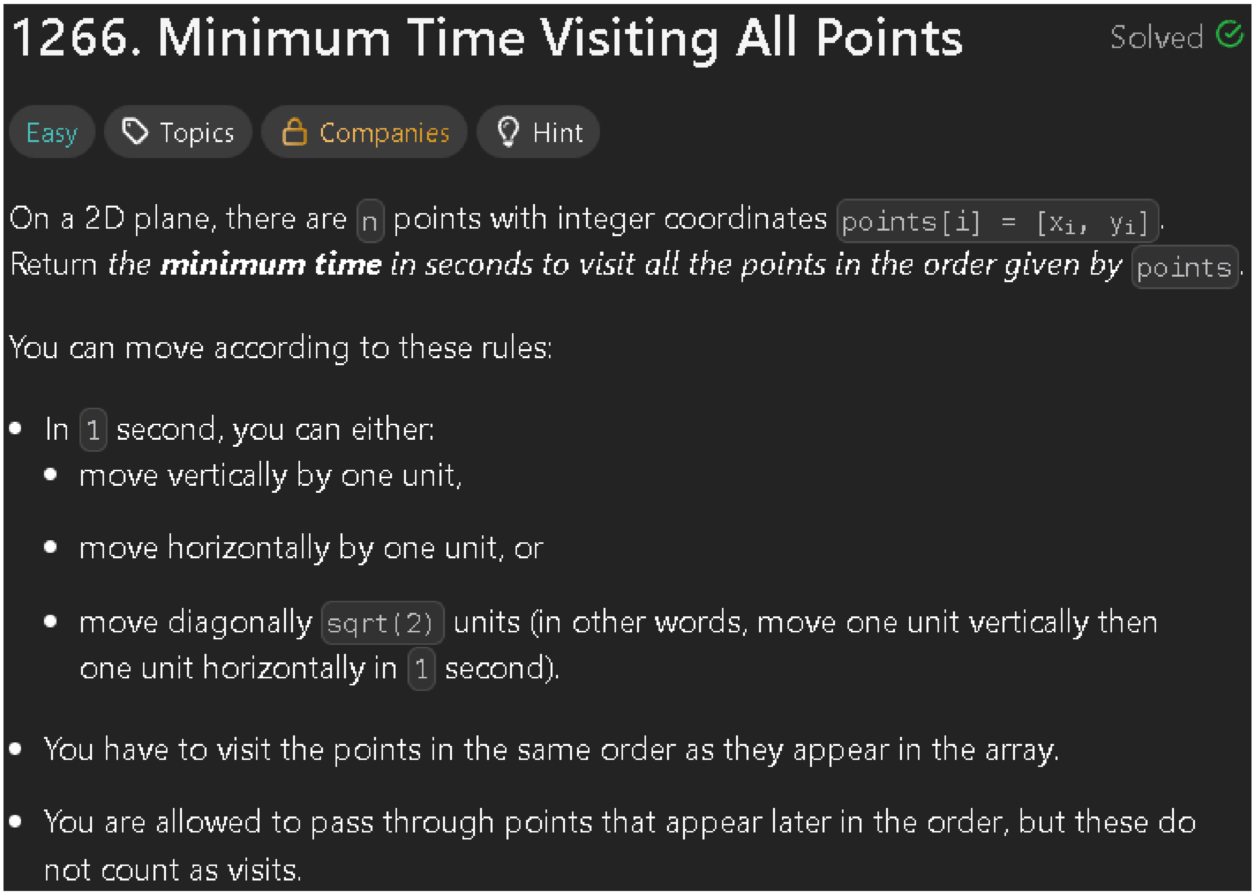


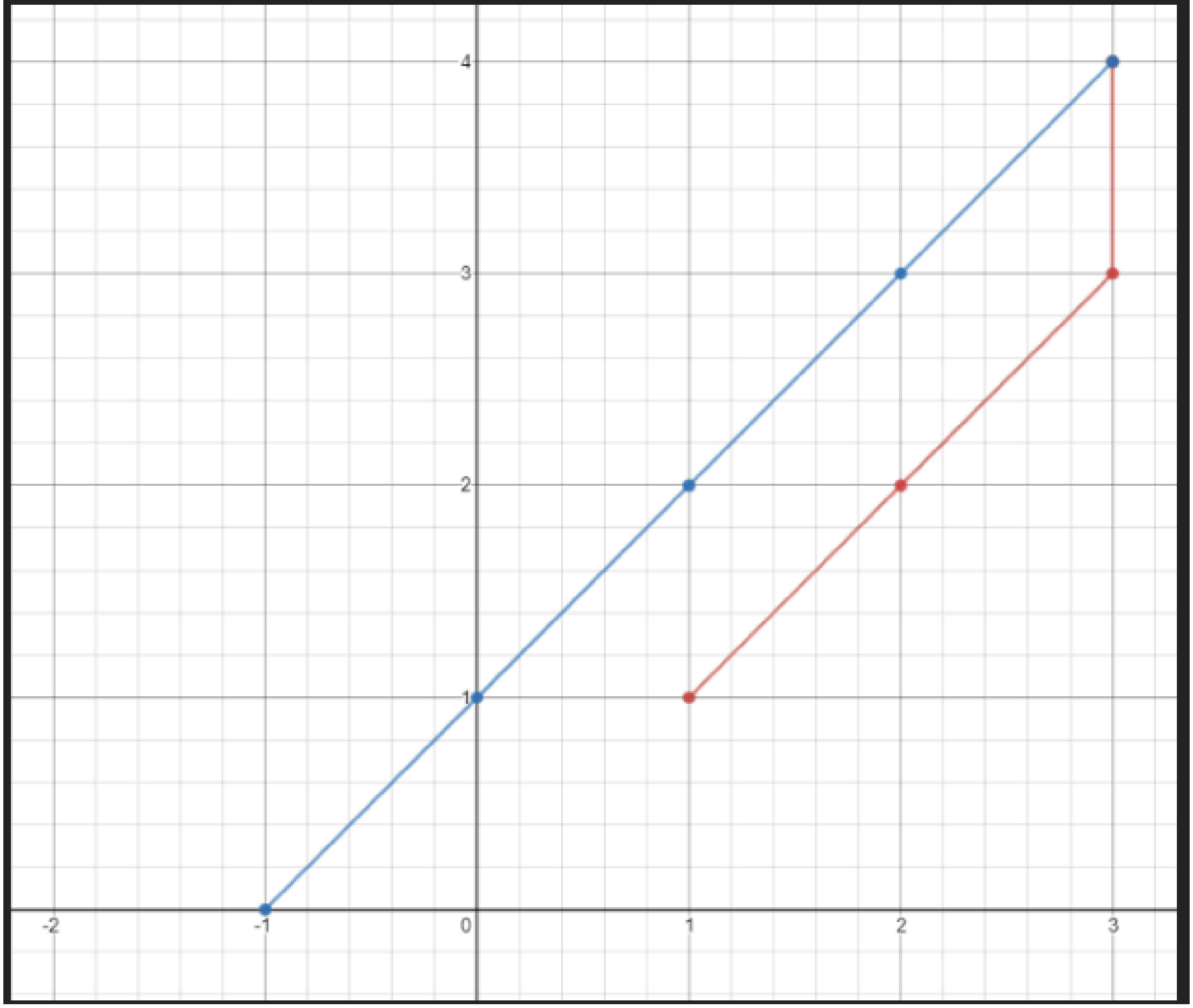
1266. Minimum Time Visiting All Points

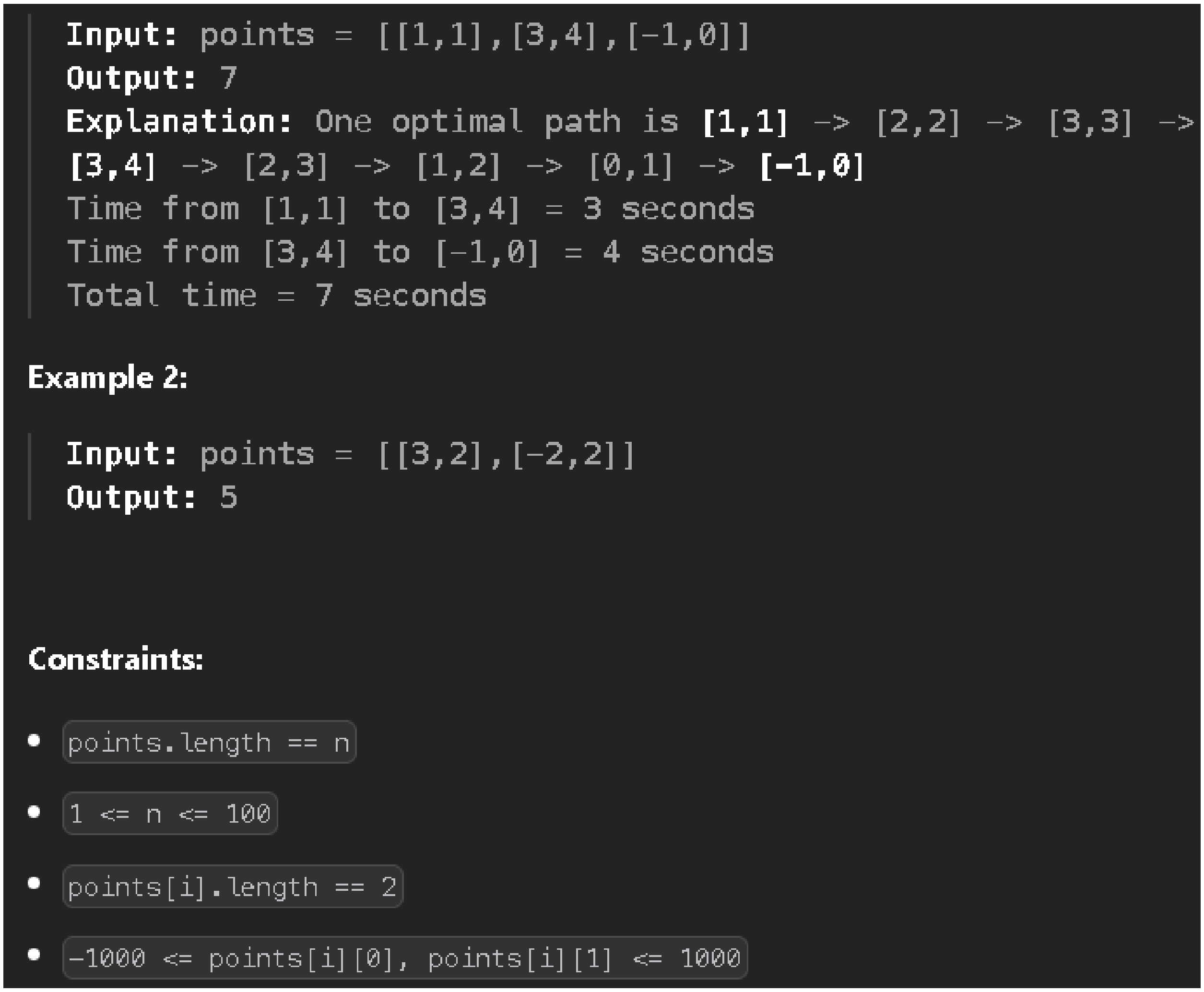
On a 2D plane, there are `n` points with integer coordinates `points[i]` = [x_i, y_i]. Return *the **minimum time** in seconds to visit all the points in the order given by* `points`.

You can move according to these rules:

- In `1` second, you can either:
  - move vertically by one unit,
  - move horizontally by one unit, or
  - move diagonally `sqrt(2)` units (in other words, move one unit vertically then one unit horizontally in `1` second).
- You have to visit the points in the same order as they appear in the array.
- You are allowed to pass through points that appear later in the order, but these do not count as visits.

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] -> [-1,0]
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`

Python:

```
class Solution:
    def minTimeToVisitAllPoints(self, p: List[List[int]]) -> int:
        Ans = 0
        for i in range(1, len(p)):
            Ans += max(
                abs(p[i][0] - p[i - 1][0]),
                abs(p[i][1] - p[i - 1][1])
            )
        return Ans
```

JavaScript:

```
var minTimeToVisitAllPoints = function(p) {
    let ans = 0;
    for (let i = 1; i < p.length; i++) {
        ans += Math.max(
            Math.abs(p[i][0] - p[i - 1][0]),
            Math.abs(p[i][1] - p[i - 1][1])
        );
    }
    return ans;
};
```

Java:

```
class Solution {
    public int minTimeToVisitAllPoints(int[][] p) {
        int ans = 0;
        for (int i = 1; i < p.length; i++) {
            ans += Math.max(
                Math.abs(p[i][0] - p[i - 1][0]),
                Math.abs(p[i][1] - p[i - 1][1])
            );
        }
        return ans;
    }
}
```