2.1

1584. Min Cost to Connect All Points

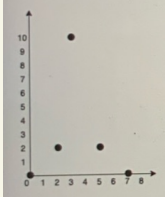
Medium · Topics · Companies · Hint

You are given an array `points` representing integer coordinates of some points on a 2D-plane, where `points[i] = [xi, yi]`.

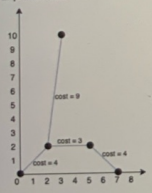
The cost of connecting two points `[xi, yi]` and `[xj, yj]` is the **manhattan distance** between them: `|xi - xj| + |yi - yj|`, where `|val|` denotes the absolute value of `val`.

Return *the minimum cost to make all points connected.* All points are connected if there is **exactly one** simple path between any two points.

Example 1:



```
Input: points = [[0,0],[2,2],[3,10],[5,2],[7,0]]
Output: 20
Explanation:
```



We can connect the points as shown above to get the minimum cost of 20.
Notice that there is a unique path between every pair of points.

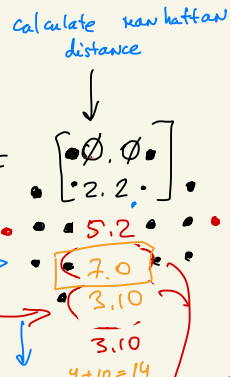
Example 2:

```
Input: points = [[3,12],[-2,5],[-4,1]]
Output: 18
```

calculate manhattan distance

points[][] = [[0,0],[2,2],[3,10],[5,2],[7,0]]

swap

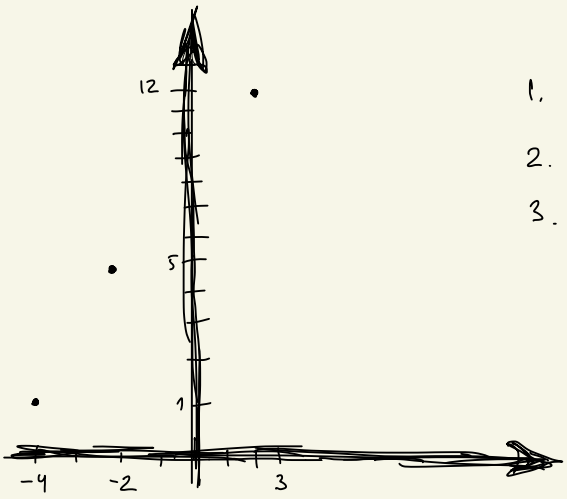


3+10=13
5+2=7
7+0=7

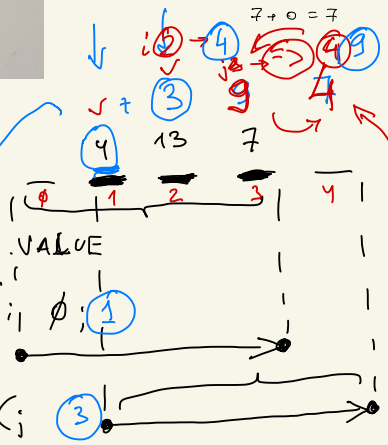
1+8=9
3+0=3
5+2=7

swap in points[j]
tempPoint 7,0

swap in dist[]
tempDist = dist[j]
dist[j] = dist[i+1]



1. dist[]
2. fill() MAX_VALUE
3. for (i; 0; 1)



Math.abs

Min(dist[j], points[i,0] - points[j,0] + points[i,1] - points[j,1])

[-4,1] [-2,5] = 2+4 = 6

[-2,5] [3,12] = 5+7 = 12

// 18

ans += dist[i+1]

ans = 4, 7, 11, 20 ;

```java
class Solution {
    public int minCostConnectPoints(int[][] points) {
        int[] dist = new int[points.length];
        Arrays.fill(dist, Integer.MAX_VALUE);
        int ans = 0;

        for (int i = 0; i < points.length - 1; i++) {
            for (int j = i + 1; j < points.length; j++) {
                dist[j] = Math.min(dist[j], Math.abs(points[i][0] - points[j][0]) +
                        Math.abs(points[i][1] - points[j][1]));
                if (dist[j] < dist[i + 1]) {
                    final int[] tempPoint = points[j];
                    points[j] = points[i + 1];
                    points[i + 1] = tempPoint;
                    final int tempDist = dist[j];
                    dist[j] = dist[i + 1];
                    dist[i + 1] = tempDist;
                }
            }
            ans += dist[i + 1];
        }
        return ans;
    }
```

recheck that one's swap. AGAIN