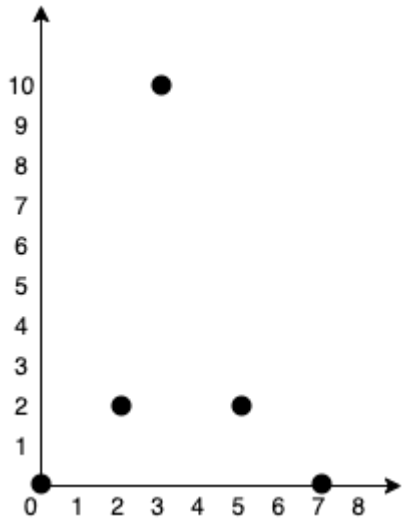
1584. Min Cost to Connect All Points

You are given an array `points` representing integer coordinates of some points on a 2D-plane, where `points[i] = [x<sub>i</sub>, y<sub>i</sub>]`.

The cost of connecting two points `[x<sub>i</sub>, y<sub>i</sub>]` and `[x<sub>j</sub>, y<sub>j</sub>]` is the **manhattan distance** between them: `|x<sub>i</sub> - x<sub>j</sub>| + |y<sub>i</sub> - y<sub>j</sub>|`, 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:
![](_resources/325e5a4fefd14139a7ee615ec97eb4f6.png)
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
```

Constraints:

- `1 <= points.length <= 1000`
- `-10<sup>6</sup> <= x<sub>i</sub>, y<sub>i</sub> <= 10<sup>6</sup>`

- All pairs `(x<sub>i</sub>, y<sub>i</sub>)` are distinct.

```
from collections import defaultdict
import heapq
class Solution:
    def minCostConnectPoints(self, points: List[List[int]]) -> int:
        graph = defaultdict(list)

        for i in range(len(points)):
            for j in range(len(points)):
                if i!=j:
                    dis = abs(points[i][0]-points[j][0])+abs(points[i][1]-
points[j][1])
                    graph[i].append((j,dis))
        heap = [[0,0]]
        amount = 0
        visited = [False]*len(points)
        while len(heap)>0:
            cost,point = heapq.heappop(heap)
            if visited[point] == True:
                continue
            visited[point] = True
            amount=cost+amount
            for nbr in graph[point]:
                pt,wght = nbr
                if visited[pt] == False:
                    heapq.heappush(heap,(wght,pt))
        return amount
```