This is a basic union-find problem. Given a graph with points being added, we can at least solve:

1. How many islands in total?
2. Which island is pointA belonging to?
3. Are pointA and pointB connected?

The idea is simple. To represent a list of islands, we use **trees**. i.e., a list of roots. This helps us find the identifier of an island faster. If roots[c] = p means the parent of node c is p, we can climb up the parent chain to find out the identifier of an island, i.e., which island this point belongs to:

Do root[root[roots[c]]]... until root[c] == c;

To transform the two dimension problem into the classic UF, perform a linear mapping:

int id = n \* x + y;

Initially assume every cell are in non-island set {-1}. When point A is added, we create a new root, i.e., a new island. Then, check if any of its 4 neighbors belong to the same island. If not, union the neighbor by setting the root to be the same. Remember to skip non-island cells.

**UNION** operation is only changing the root parent so the running time is O(1).

**FIND** operation is proportional to the depth of the tree. If N is the number of points added, the average running time is O(logN), and a sequence of 4N operations take O(NlogN). If there is no balancing, the worse case could be O(N^2).

Remember that one island could have different roots[node] value for each node. Because roots[node] is the parent of the node, not the highest root of the island. To find the actually root, we have to climb up the tree by calling **findIsland** function.

Here I've attached my solution. There can be at least two improvements: union by rank & path compression. However I suggest first finish the basis, then discuss the improvements.