SOLUTION

Simulation

The simple solution is to "draw" the walls onto a two-dimensional array and use a 0-1 BFS algorithm (starting from any outer cell) to calculate the flooding time for each cell. A wall is left standing after the flood if the times needed to reach its two sides are equal.

The 0-1 BFS algorithm is used to find the shortest path in graphs in which edge has weight exactly 0 or 1 (using a double-ended queue and inserting elements at the front when expanding through a 0-edge). In our model, each pair of two adjacent cells is connected by an edge; weight of the edge is 1 if there is a wall between the cells, and 0 otherwise.

Since the size of the graph depends on the coordinates of the points, the number of cells we need to consider is too big in general. This solution was awarded approximately 40 points.

Simulation with coordinate compression

To improve the above algorithm, we can observe that the exact coordinates of points are irrelevant – only the relative order of points matters for the purpose of counting the walls standing after the flood. Therefore, if the x-coordinates take A different values and the y-coordinates take B different values, we can map them to sets $\{1, 2, ..., A\}$ and $\{1, 2, ..., B\}$, respectively. The number of cells we need to consider while searching is now $O(A \cdot B)$.

As both A and B are bounded by N (the number of points), this approach has total time complexity $O(N^2)$. This solution was awarded approximately 55 points.

Model solution (dual-graph)

We represent each region (a maximal set of connected cells) as a node in a graph and we add edges between pairs of nodes if the corresponding regions share a wall. We also add the outer region as a node in the graph and connect it with all regions immediately exposed to water.

Notice that we do not need to find the exact flooding time for each region – we are only interested if two regions sharing a wall are equally distant from the outer region or not. If one connected set of regions is completely contained inside another region we can consider it separately while generating the above graph and connect it directly to the outer region.

Running a simple BFS algorithm on the above graph starting from outer region gives us enough data to solve the task. The hardest part of the solution is generating the graph.

One possible strategy is to imagine each wall as two one-way roads (going in opposite directions). Now, we pick any road and drive along, turning right at each intersection – if we can turn right we do so, else if we can go forward we do so, else if we can turn left we do so, else we turn backwards. For each starting road we will make one lap around one region and end up where we started. By successively choosing new roads and traversing them, we discover all possible regions. Exactly one traversal will give the an outer region. If for a particular wall, the two roads yield different regions, we connect those regions by an edge.

This algorithm can be implemented with time complexity $O(N+W)$.

Alternative model solution (Outer wall traversal)

Another, somewhat different, approach to the problem is to continuously walk around the components and figure out which of the outer walls are going to collapse. We start by selecting a leftmost outer wall. Now, imagine we put a right hand on the wall and start walking around the walls keeping the hand on the wall until we return to the starting position. We observe that, of all the walls we touched, only those that we touched on both sides will survive the flood. Next, we remove all the walls we touched, and repeat the process until all walls are removed. This algorithm can be implemented in O(N log N + W) and also yields full points.