200. Number of Islands

847 33

Description (/problems/number-of-islands/description/) Hints (/problems/number-of-islands/hints/) Submissions (/problems/number-of-islar

Quick Navigation View in Article (/articles/number-of-islands/)

Notes

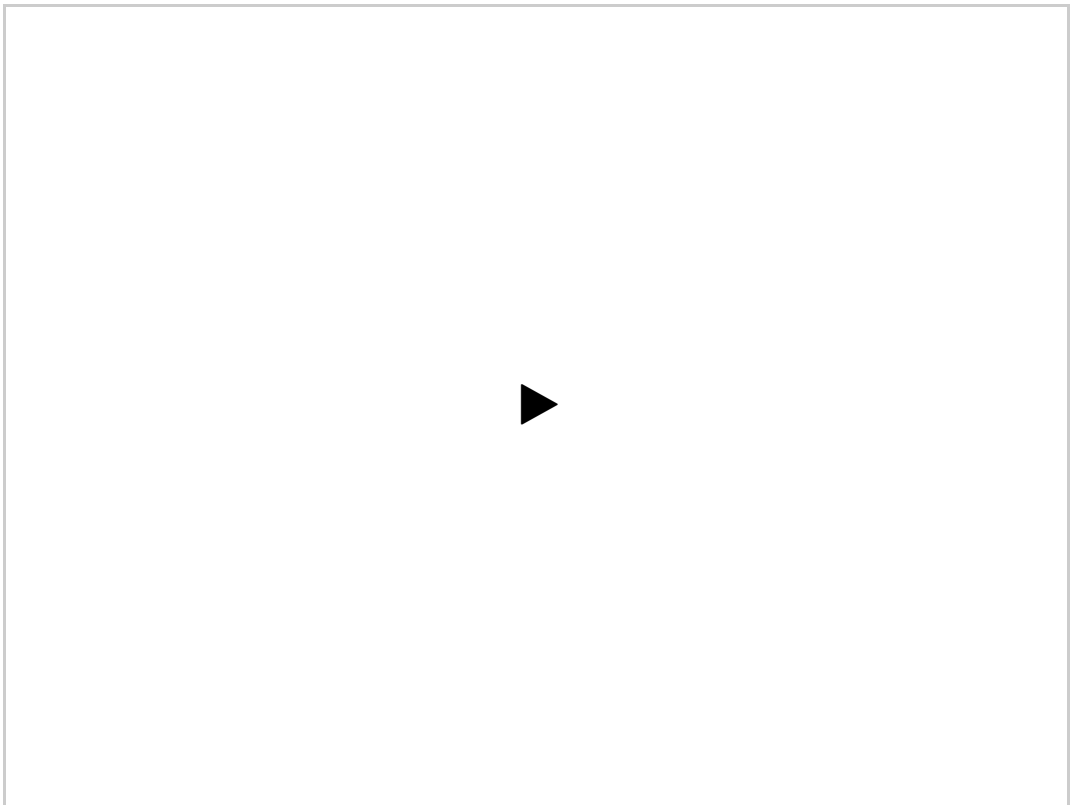
Approach #1 DFS [Accepted]

Intuition

Treat the 2d grid map as an undirected graph and there is an edge between two horizontally or vertically adjacent nodes of value '1'.

Algorithm

Linear scan the 2d grid map, if a node contains a '1', then it is a root node that triggers a Depth First Search. During DFS, every visited node should be set as '0' to mark as visited node. Count the number of root nodes that trigger DFS, this number would be the number of islands since each DFS starting at some root identifies an island.



1 / 8

C++ Java Copy

Complexity Analysis

- Time complexity : $O(M \times N)$ where M is the number of rows and N is the number of columns.
- Space complexity : worst case $O(M \times N)$ in case that the grid map is filled with lands where DFS goes by $M \times N$ deep.

Approach #2: BFS [Accepted]

Algorithm

Linear scan the 2d grid map, if a node contains a '1', then it is a root node that triggers a Breadth First Search. Put it into a queue and set its value as '0' to mark as visited node. Iteratively search the neighbors of enqueued nodes until the queue becomes empty.

C++ Java Copy

Complexity Analysis

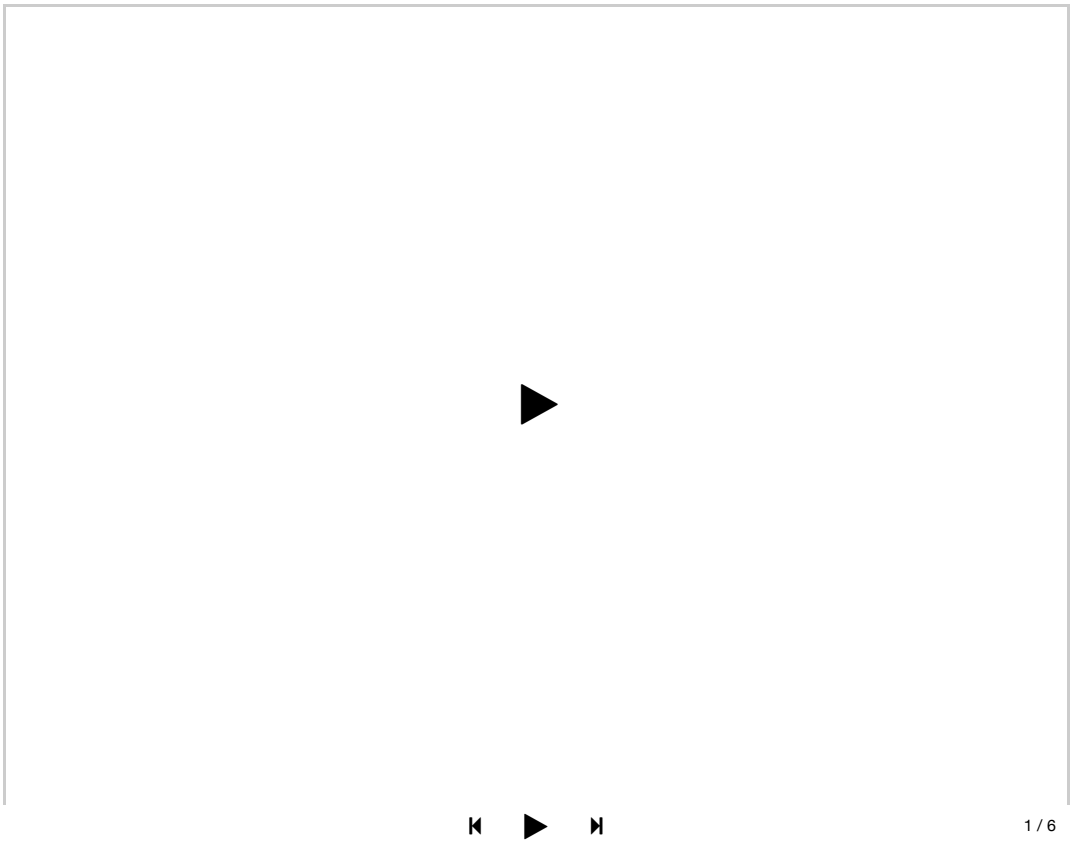
- Time complexity : $O(M \times N)$ where M is the number of rows and N is the number of columns.
- Space complexity : $O(min(M, N))$ because in worst case where the grid is filled with lands, the size of queue can grow up to min(M, N).

Approach #3: Union Find (aka Disjoint Set) [Accepted]

Algorithm

Traverse the 2d grid map and union adjacent lands horizontally or vertically, at the end, return the number of connected components maintained in the UnionFind data structure.

For details regarding to Union Find, you can refer to this article (https://leetcode.com/articles/redundant-connection/).



C++ Java Copy

Complexity Analysis

- Time complexity : $O(M \times N)$ where M is the number of rows and N is the number of columns. Note that Union operation takes essentially constant time[1] when UnionFind is implemented with both path compression and union by rank.
- Space complexity : $O(M \times N)$ as required by UnionFind data structure.

Analysis written by: @imsure (https://leetcode.com/imsure).

Thanks to @williamfu4leetcode (https://leetcode.com/williamfu4leetcode/) for correcting the space complexity analysis of BFS approach.

Footnotes

1. https://en.wikipedia.org/wiki/Disjoint-set_data_structure (https://en.wikipedia.org/wiki/Disjoint-set_data_structure) ↩

Join the conversation

Signed in as **williamfu4leetcode**.

Post a Reply

I (https://discuss.leetcode.com/user/imsure)

imsure commented 2 months ago

@jocelynayoga (https://discuss.leetcode.com/uid/15328) Thanks for the comments! Think about the worse case: the grid if filled with '1'. During the first iteration, it takes MN to do BFS to identify the giant island. After that, it takes another MN to scan the grid, even though all the cells become '0' by then. So the total is 2MN which is still O(M*N).