723. Candy Crush

Medium 677 284 Add to List Share

This question is about implementing a basic elimination algorithm for Candy Crush.

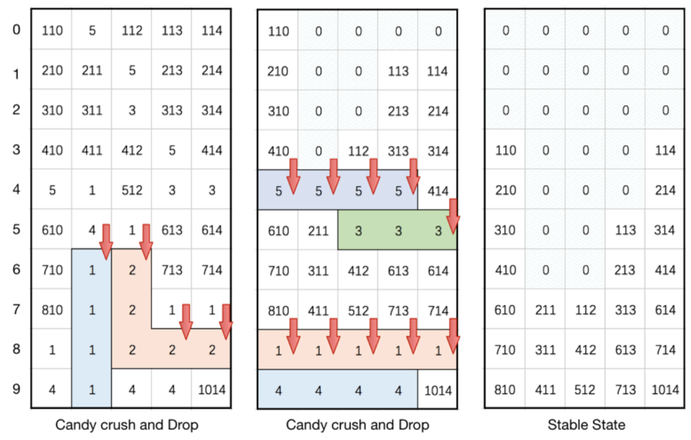
Given an `m x n` integer array `board` representing the grid of candy where `board[i][j]` represents the type of candy. A value of `board[i][j] == 0` represents that the cell is empty.

The given board represents the state of the game following the player's move. Now, you need to restore the board to a stable state by crushing candies according to the following rules:

- If three or more candies of the same type are adjacent vertically or horizontally, crush them all at the same time - these positions become empty.
- After crushing all candies simultaneously, if an empty space on the board has candies on top of itself, then these candies will drop until they hit a candy or bottom at the same time. No new candies will drop outside the top boundary.
- After the above steps, there may exist more candies that can be crushed. If so, you need to repeat the above steps.
- If there does not exist more candies that can be crushed (i.e., the board is stable), then return the current board.

You need to perform the above rules until the board becomes stable, then return *the stable board*.

Example 1:




```
Input: board = [[110,5,112,113,114],[210,211,5,213,214],[310,311,3,313,314],[410,411,412,5,414],
[5,1,512,3,3],[610,4,1,613,614],[710,1,2,713,714],[810,1,2,1,1],[1,1,2,2,2],[4,1,4,4,1014]]
Output: [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[110,0,0,0,114],[210,0,0,0,214],[310,0,0,113,314],
[410,0,0,213,414],[610,211,112,313,614],[710,311,412,613,714],[810,411,512,713,1014]]
```

Example 2:

```
Input: board = [[1,3,5,5,2],[3,4,3,3,1],[3,2,4,5,2],[2,4,4,5,5],[1,4,4,1,1]]
Output: [[1,3,0,0,0],[3,4,0,5,2],[3,2,0,3,1],[2,4,0,5,2],[1,4,3,1,1]]
```

Constraints:

- `m == board.length`
- `n == board[i].length`
- `3 <= m, n <= 50`
- `1 <= board[i][j] <= 2000`

Accepted 43,062 | Submissions 58,013

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Bloomberg | 5 Capital One | 4 Facebook | 2 Pinterest | 2

Related Topics

Array Two Pointers Matrix Simulation

Hide Hint 1

Carefully perform the "crush" and "gravity" steps. In the crush step, flag each candy that should be removed, then go through and crush each flagged candy. In the gravity step, collect the candy in each column and then rewrite the column appropriately. Do these steps repeatedly until there's no work left to do.

Java Autocomplete

```java
class Solution {
    public int[][] candyCrush(int[][] board) {
        int R = board.length, C = board[0].length;
        boolean todo = false;
        for (int r = 0; r < R; ++r) {
            for (int c = 0; c + 2 < C; ++c) {
                int v = Math.abs(board[r][c]);
                if (v != 0 && v == Math.abs(board[r][c+1]) && v == Math.abs(board[r][c+2])) {
                    board[r][c] = board[r][c+1] = board[r][c+2] = -v;
                    todo = true;
                }
            }
        }
        for (int r = 0; r + 2 < R; ++r) {
            for (int c = 0; c < C; ++c) {
                int v = Math.abs(board[r][c]);
                if (v != 0 && v == Math.abs(board[r+1][c]) && v == Math.abs(board[r+2][c])) {
                    board[r][c] = board[r+1][c] = board[r+2][c] = -v;
                    todo = true;
                }
            }
        }

        for (int c = 0; c < C; ++c) {
            int wr = R - 1;
            for (int r = R-1; r >= 0; --r)
                if (board[r][c] > 0)
                    board[wr--][c] = board[r][c];
            while (wr >= 0)
                board[wr--][c] = 0;
        }

        return todo ? candyCrush(board) : board;
    }
}
```