Bob the Builder

Problem Solving Technique: Why is this hard?
Explore straightforward "just do it" approaches or thought processes, even if they seem wrong. Articulate precisely *what's stopping you*, i.e. what is the heart of what makes the problem hard—this gives you something concrete to try to attack.

Standard Toolbox: Complete Search
Can't figure out the right value? Just try everything.

- **Pros:** Always works
- **Cons:** Might be too slow (do the analysis)

Can't I just brute force it?

- For each empty cell, try placing the fountain there.
  - There can be up to $O(rc)$ cells to try.
- Then, for each one, re-compute the score of the board.
  - Computing the score of the entire board is $O(rc)$.

The answer is the maximum score attained across all possible placements.

This gives a running time of $O((rc)^2)$, which is actually enough to get 67 points. How do we improve this?

Standard Approach: What to cut?
Suppose your running time is quadratic or worse because it is the product of many linear factors (usually from nested loops or such).
If you want to improve it, try to think: Which of these linear terms requires the least amount of smartness to optimize away?

There *must* be a faster way to re-compute the score of the entire board.

Problem Solving Technique: What does it do?

Have a strange operation? Try rephrasing it in more concrete terms:

- What is its effect on the other things in the scenario?
- What does it enable you to do, in terms of these other things?

It may also be helpful to describe what it *doesn't* do.

Placing a fountain on the board does these things:

- Activates the fountains next to it (+2 for each).
- Is an active fountain itself if placed next to a fountain (+2).

Also, *it has no effect on any fountains other than the ones next to it.*

So, placing a fountain on some cell leaves the old score mostly the same. We only need to check a small area around that cell (it and its neighbors) to look for new activated fountains.

We can spend $O(rc)$ to compute the score the first time *once*, save that value, and then "the score if I placed a new fountain here" is just

$$\text{old score} + 2 \cdot (\text{no. of new activated fountains})$$

which can be computed in $O(1)$. So, even if we brute-force test all possible cells to place the fountain in, the total runtime is still only $O(rc)$, which gets 100 points.

Remark

This solution can be more succinctly stated as

$$\text{old score} + 2 \cdot (1 + \text{max no. of unactivated fountains next to any cell}).$$

This is essentially the same solution as the one described above. Can you see why?
Also, it's wrong! Can you spot the edge case?