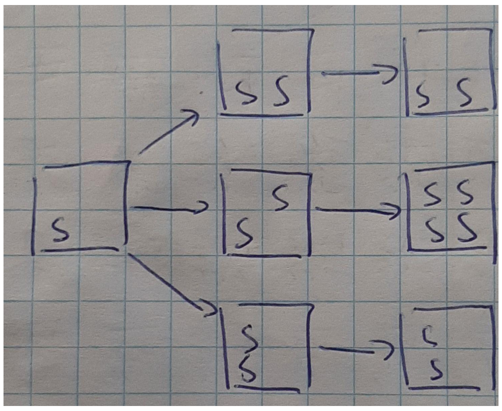
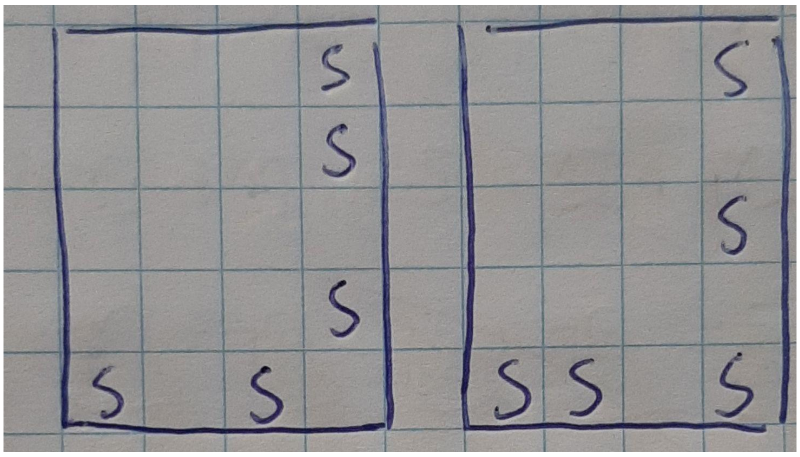
Leo Venn   StudentID: 3430125

To solve how many sick people, we need to fill a board we first look at cases with no immunes. A square board can be filled by a minimum of the dimension of the square amount of people. For example, a 5 by 5 grid can be filled with 5 sick people. This is easy to see by example. Given a single S on the board and one S to place, if we place the S either on top or next to the other S, we only fill that side of the S. But if we place it on the diagonal space next to it, we get a 2 by 2 grid.



We can easily see that from this 2 by 2, if we were given another S, we can either expand one of the two sides to create a 2 by 3 or 3 by 2, but if we place it in the corner, we get a 3 by 3 grid. Therefore, to fill an n-by-n square we need n sick people. Another way to think about it is to take two adjacent sides of the square, add them up, then divide by 2. E.g for a 5 by 5 grid we have 5 + 5 = 10 and 10/2 = 5. So, we need 5 sick people. This method generalizes to rectangles nicer. We apply the same method but instead round up when we have a .5 e.g for a 4 by 5 rectangle we have 4+5/2 = 4.5, round up to 5. So, we can fill the rectangle in 5 and we do so by placing them like so.



So, we have two configurations, one with an S in the corner and one without but both will fill the rectangle.

FORMULA: Given an m by n board, we know we will need at least ceil(m+n/2) sick people to fill the board.

To figure out the minimum amount of sick people required to fill any board with any amount of immunes, we do an exhaustive search. Given an m by n board, we know for a fact we will need at least ceil(m+n/2) sick people. We first start off with that amount and we go through every possible different board state containing this amount of sick people to see if the board fills. If we don't find a solution, we add another sick person in and redo the search until it's solved and whatever amount of sick people there are at the point of solving is therefore the minimum. *Obviously this computation becomes massive, for example a 5 by 5 board has 25*24*23*22*21 or 6375600 possible starting states given only 5 starting sick people. However, this calculation assumes that there is a distinction from each individual sick person. Such as if we have a board with a certain configuration of sick people, if we swap two of them so the location of the sick people are the same, this swap results in a new state. But this is not the case as we do not make distinctions between sick individuals because their effect on infecting the board remains the same regardless. Therefore instead of a permutation problem we have a combination problem. Meaning the actual amount of unique states is (25*24*23*22*21)/5! = 53130 for five sick people on a 5 by 5 board*. The amount of possible starting board states given any m by n board, S amounts of sick people to place, and N immunes is described by (n*m-N)!/((n*m-N-S)!*S!).

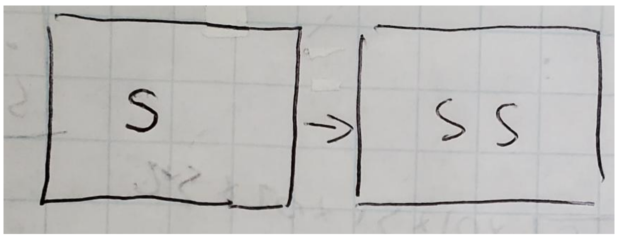
*However, the total number of starting states that a board can take given that we can vary the amount of sick people however we want, is simply described by the product of the number of states that a cell that we can place a sick person into can take, over all such cells. In our case it's either sick or vulnerable, giving two states. So therefore we have:

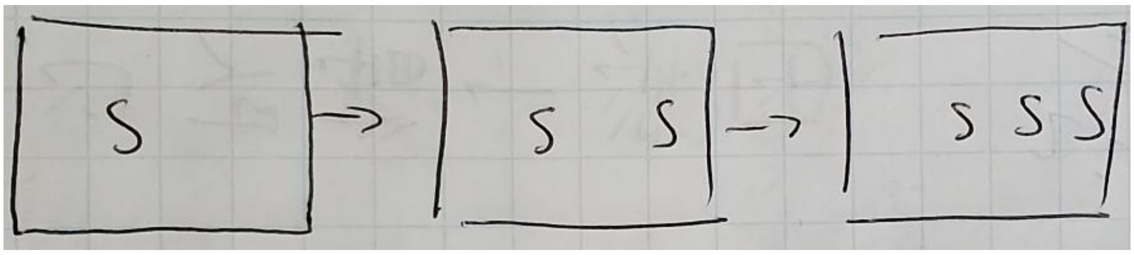
2^(cells we can place sick people in)

E.g. For a 5 by 5 board with 5 immunes, we have 20 spaces we can put a sick person in. So therefore the total amount of starting states this board can take is 2^20. This is obviously massive, but we can trim this space by always starting with ceil(m+n/2) sick people (in this case 5). In most cases however, we rarely will need to test all these starting states to see if the board solves as sick people can propagate over many cells so a solution is often closer than it seems.

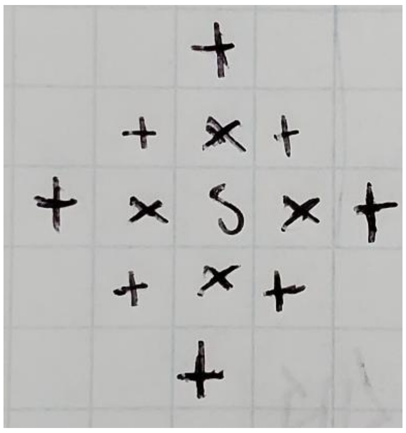
A simple method for testing all these states is to just start by filling out the first S sick spaces reading from left to right, top to bottom, then iteratively shifting the end sick person over all the valid cells (first moving right, then down and to the left most valid cell when at the end of the valid cells on that row) until the final cell. If no solution is found we then return this sick person to their initial cell. We then shift the last 2 sick people by one then once again shift the last sick person over all cells. We then return the last sick person back to sitting next to the second last sick person and shift both of them again by one. We do this until both sick persons reach the end of the board where we then return both of them to their initial starting cells and shift 3 forward one space. We then repeat this over all the sick people until the board is solved, or you move all sick people to the final S amount of valid cells where you add a sick person then restart the process. It's easy to see this process will cover all possible board states as in order for all sick people to traverse the board from the first valid S sick spaces to the last valid S sick spaces they must go over all possible states in between before getting there. We never skip a cell and only move the sick “train” once all possible combinations of the board have been tested for the sub train in front of it. In terms of repeated boards, this process does not repeat. To see this we think of the sick people as being unique such as having been labeled s1 to sN. We label the left-top most piece s1 and then label them chronologically from left to right up to sN. Because this method restricts it such that pieces ahead of other pieces can never exist in a cell before it, the ordering of the sick people from left to right, top to bottom, must always be of ascending order from 1 to N. We know that states are repeated when we take a board configuration and swap around sick people, but if they are labeled then this swap breaks the ascending order, and so therefore this cannot happen and all boards reached by this method are unique (up until a symmetry as discussed below)

There are optimizations we can do. For example a 5 by 5 board could be symmetric when mirrored and rotated giving 8 possible symmetries so the amount of unique states is then 796950. So depending on the symmetries we can reduce the space. Another optimization we can do is that we observe that there is no point placing sick people adjacent to each other if there is a space over in the direction of the adjacent space that's free. This is as placing it one space over fills the space that we were going to try to fill before with the bonus of the extra filled space we just placed into E.g:





So if we place a sick person on the board who has sufficient space around them, we can at most remove 5 possible cells with a single sick person placed.



(X represents places we do not place sick people. The +'s next to X's represent spaces where we can place a sick person that fill out that X anyway)

The amount of cells we can remove from the valid cells for our next sick person placement thus depends on the spaces around the sick person we place. E.g. depending on how close the sick person is to a wall or how many immunes surround it. In the best case where there is enough space in all four compass directions, we remove 5. In the worst case, such as small boards or when the sick person is surrounded by immune cells, we have only one space removed. Thus we can reduce our search by first always maximising the amount of spaces we cancel for when we place a sick person, then if that does not solve it, we allow for placements that cancel out fewer spaces. This optimizes the space further by first trying to solve easier boards. Note when a sick person cancels out a cell that's already cancelled by another sick person, that cell does not contribute to the amount of cells that sick person removes. Given a board can be designed any way, there's no easy way of determining the average amount of cells removed per sick person placement so we know this optimization works, just its effectiveness depends on the board.