

In terms of space complexity for this program, what we are really thinking about is the storing of the 2D array. What a 2D array is really storing is just a bunch of values that we put in that array. Thus, as the size of the array increases, the number of values we store increases, and the more memory the computer must use to store these values. If we assume that the 2D array is always a "square," then the number of values that we store will exponentially increase with the size of the array. This is because the number of values that we store will be dependent on the dimensions of the grid. For example, If we have a 1x1 grid, 1 value will be stored, a 2x2 grid will store 4 values, a 3x3 grid will store 9 values and so on. If we assume that the "side lengths" of our grid are the same, therefore, our grid is always a square then with each individual increase of "n" (the side length) the amount of memory needed will be  $O(n^2)$  and thus will increase exponentially. However, if we don't assume that our grid is always a square and that there can be different "side" lengths of our grid, the space complexity may be a variation of on the order of linear growth. It might look like  $O(kn)$  instead of  $O(n)$ .