Most Frequent Value: given an array of size n, return the value which occurs the most.

Problem size: n, the number of rows or colums.

To make analysis easier, we can divide the algorithm into three parts, thus making $T(n) = T_1(n) + T_2(n) + T_3(n)$

$T_1(n)$: Initializing variable count

when we initialize an array it will always require n operations. In Java an array will be initialized with the value zero, but in languages like C/C++ we would need another loop to initialize element values.

$\therefore \quad T_1(n) = O(n)$

$T_1(n) = \Omega(n)$

$T_2(n)$

The second section is our nested for loops. When analyzing algorithms we want to focus on instructions that are important for the algorithm. This algorithm we'll focus on

```
int val = arr[row][col];   1
counts++;                  2
```

We can decompose 1 into 2 operations arr[row][col] and assigning a value to val. arr[row][col] counts as 1 operation since we access the address in memory by multiplying the size of int by the offset (row x col). We currently have 3n operations. We can add an additional n operations by incrementing col in each inner loop.

Adding our declaration of col brings our inner loop to $4n+1$, which the outer loop will always do n times.

$$n(4n+1) = 4n^2 + n$$

$\therefore \quad T_2(n) = O(n^2)$

$\quad\quad T_2(n) = \Omega(n^2)$

For our last section we need to look at two scenarios.

I. Worst Case: when count is in ascending order.

First we initialize max $\quad O(n) = 1$

Next, our for loop preforms a linear search across our array. The body of the for loop will preforms 4 operation each loop (when its the wost case scenario)

1. counts[val] (we'll call this α)
2. $\alpha > max$
3. counts[val] (we'll call this β)
4. $max = \beta$

Since these will fire each loop in the worst case scenario

$$T(n) = O(4n+1) = O(n)$$

II. Best case scenario

The zeroth element is the largest

This would chang our last formula to $T(n) = \Omega(n+5)$ since the if statement is only fired once.

In both cases $T_3 = O(n)$

$$T_3 = \Omega(n)$$

Now we can sum these equations together

$$T(n) = T_1(n) + T_2(n) + T_3(n)$$

$$= O(n) + O(n^2] + O(n)$$

$$= \max(O(n), O(n^2)) + O(n)$$

$$= O(n^2) + O(n)$$

$$= \max(O(n^2), O(n))$$

$$= O(n^2)$$

$$T(n) = \Omega(n) + \Omega(n^2) + \Omega(n)$$

$$= \max(\Omega(n), \Omega(n^2)) + \Omega(n)$$

$$= \Omega(n^2) + \Omega(n)$$

$$= \max(\Omega(n^2), \Omega(n))$$

$$= \Omega(n^2)$$

Since $O(n^2) = \Omega(n^2)$ then we can say $\Theta(n^2)$

Here we were able to create the equations for our algorithm. We simplified steps in order to determine $O(\cdot)$ and $\Omega(\cdot)$, but we could have easily found constants c_1, c_2 and n_0