**Algorithm HW6**

**Presumption**

The input will be a 2-D array. Each 1-D array in this array will be in an increasing order.

**Solution**

**Approach 1:**

1. Create index pointers for each 1-D array and a complete 1-D array to store all elements from input. And create a count initialized to -1. After finding the minimum value in each pair of comparison, increment the `count`. Create a variable `curMin` to store minimum value in each comparison.
2. Check if the number of elements is even or odd. This will determine how to get the median.
3. Compare each element of 1-D array at their current index pointer with each other and find the minimum value. Then, put the minimum value into the first empty block in the complete array.
4. If `count` equals to `mid`, then get the median number in the array. If the number of array is odd, the median is just the mid-th element of the complete array. If the number of array is even, the median is the average value of (mid – 1)-th element and the mid-th element of the complete array.

**Approach 2:**

1. Create index pointers for each 1-D array and a complete 1-D array to store all elements from input. And create a count initialized to -1. After finding the minimum value in each pair of comparison, increment the `count`. Create a variable `curMin` to store minimum value in each comparison and a variable `prevMin` to store the previous minimum value.
2. Check if the number of elements is even or odd. This will determine how to get the median.
3. Compare each element of 1-D array at their current index pointer with each other and find the minimum value. Then, set `curMin` to the minimum value.
4. If `count` equals to `mid`, then get the median number in the array. If the number of array is odd, the median is the value of `curMin`. If the number of array is even, the median is the average value of `curMin` and `prevMin`.
5. Before finishing this round of loop, set `prevMin` to `curMin`.