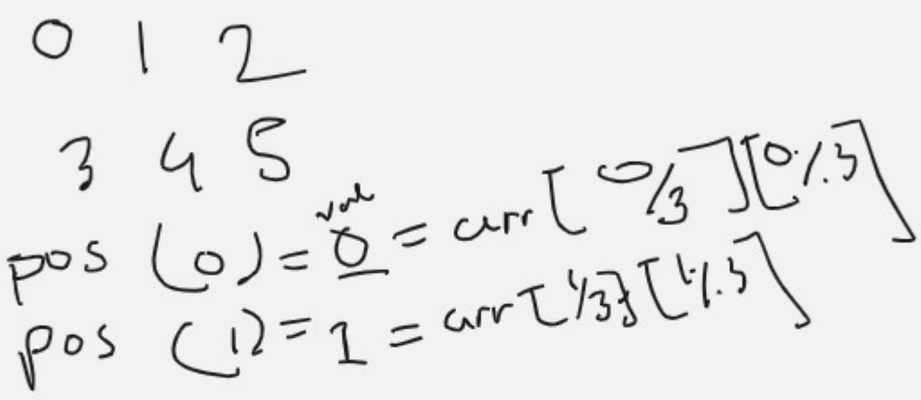
[74. Search a 2D Matrix](#)

Given a sorted 2D Matrix, we need to find a target number already given in the question. So I have a matrix say, [[1,2] [3,4]], I will be given a target say = 3, I need to find 3 in log(m*n) time. Need to find if it exists or not. Given each row is sorted and the next starting is greater than the ending of the first row.

We solve this by using only one binary search. We take low as pos 0, high as pos (m*n) - 1. The trick is to convert position into array index.



position[2] = arr[2/3][2%3] = arr[0][2]
position[3] = arr[3/3][3%3] = arr[1][0]
position[x] = arr[x/n][x%n] - Here n is the column number.

This can also be solved by using two binary searches. First we search for the nearest number from each row first column to the target. Then from that target we apply another binary search. Here the time complexity is log(m)+log(n). It is equal to the one algorithm mentioned earlier (log(m*n)).

The algorithm is:

```
search2DMatrix(Arr[m][n],target) :
   low = 0
   high = m*n - 1
   while low <= high :
      mid = low + (high - low)/2
      if(Arr[mid/n][mid%n] == target)
            return true;
      if(Arr[mid/n][mid%n] < target)
            high = mid+1
      if(Arr[mid/n][mid%n] > target)
            low = lmid - 1
   return false;
```

```
class Solution {
public:
   bool searchMatrix(vector<vector<int>>& matrix, int target) {
      int low = 0;
      int row = matrix.size();
      int column = matrix[0].size();
      int high = (row*column) - 1;
      while(low <= high){
         int mid = low + (high - low)/2;
         //cout << matrix[mid/column][mid%column] << endl;
         if(matrix[mid/column][mid%column]==target) return true;
         else if(matrix[mid/column][mid%column] < target) low = mid+1;
         else if(matrix[mid/column][mid%column] > target) high = mid-1;
         //high--;
      }
      return false;
   }
};
```