Algorithm Analysis and Design

Ghazale Kharadpoor 9513012

November, 2018

We assume that the sorted array (as mentioned in the question) has gotten to us before. So the algorithm with the time complexity of $O(m+n)$ in worst case would be like below.
First we start to traverse the matrix from right top of the matrix. Three cases will be happen:
1. If $matrix[i][j] = k$, then the result will be the location of the cell.
2. If $matrix[i][j] > k$, then we have to move one cell to left.
3. If $matrix[i][j] < k$, then we have to move one cell to below.

This is my algorithm:

```
FINDING-IN-SORTED-MATRIX($M, k$)
  $row = 0$
  $column = n - 1$
  while $true$
      if $k == M[row][column]$
          return $row, column$
      elseif $k > M[row][column]$
          if $column > 0$
              $column = column - 1$
          else return FALSE
      elseif $k < M[row][column]$
          if $row < m - 1$
              $row = row + 1$
          else return FALSE
```

• The time complexity of this algorithm is $O(n+m)$ because we start to traverse the matrix from the right top of the matrix and the longest traverse that we would have is $m+n$. In the worst case, we have to move from the right top of the matrix to the left below of the matrix. $(M[0][n-1] \rightarrow M[m-1][0])$ If we find the k sooner that the worst case, so the time cost will be less.

• At the i-th iteration of the while loop, we are traversing the i-th cell of out path on the matrix, maintaining that the value of the current cell becomes closer to the value of k.

Initialization: In this step we have to show the loop invariant is true prior to the first iteration of the while loop. After the first while loop, four cases will happen. At the first case, we reach the location of the k and return it. At the second or third case, we move to adjacent cell if k was not equal to the previous cell's amount. At the fourth case, $m = 1$ and $n = 1$, so if the only cell of the matrix was not equal to the k, we have to return $false$.

Maintenance: In this step, we have to show the loop invariant is true before and after one iteration of the while loop. Before the i-th iteration of the while loop, we are currently in a cell that is not equal to k. After the i-th loop, if k reaches the wanted cell, the cell will be returned. In other cases, we move to left or below, in order to become closer to the wanted cell (if exists).

Termination: When the loop terminates, we get the location of the wanted cell or we get the $false$ result. If we get the $false$ result, it means that we are moving in to a cell out of the matrix bound. In othe words, the equal cell to the k have not been found. The loop will be terminate certainly because $m + n$ is finite.