

```
1 //Classical Binary Search
2 public class Solution {
3     public int binarySearch(int[] array, int target) {
4         if (array==null||array.length==0){
5             return -1;
6         }
7         int left = 0;
8         int right = array.length -1;
9
10        while( left <= right){ //注意边界条件, 这边是 left小于等于right
11            int mid = left + (right - left)/2;
12            if (array[mid]==target){
13                return mid;
14            } else if (array[mid]<target){
15                left = mid +1;
16            } else {
17                right = mid -1;
18            }
19        }
20        return -1;
21    }
22
23 }
```

```
1  public class Solution {
2      public int firstOccur(int[] array, int target) {
3          if(array==null||array.length==0){
4              return -1;
5          }
6          int left = 0;
7          int right = array.length-1;
8          while(left < right -1){
9              int mid = left + (right -left)/2;
10             if (array[mid]<=target){
11                 left = mid + 1;
12             } else {
13                 right = mid;
14             }
15         }
16         if (array[left]==target){
17             return left;
18         } else if (array[right]== target){
19             return right;
20         }
21         return -1;
22     }
23 }
24
```

```
1 public class Solution {
2     public int lastOccur(int[] array, int target) {
3         if (array==null||array.length==0){
4             return -1;
5         }
6         int left = 0;
7         int right = array.length -1;
8         while (left <right -1){
9             int mid = left +(right - left )/2;
10            if (array[mid]<= target){
11                left = mid;
12            } else {
13                right = mid;
14            }
15        }
16        if (array[right]== target){ //
17            return right;
18        } else if (array[left]== target){
19            return left;
20        }
21        return -1;
22    }
23 }
24
```

```
1 public class Solution {
2     public int closest(int[] array, int target) {
3         if(array==null||array.length==0){
4             return -1;
5         }
6         int left = 0;
7         int right = array.length-1;
8         while(left <right -1){
9             int mid = left + (right - left)/2;
10            if(array[mid]==target){
11                return mid;
12            } else if (array[mid] <target){
13                left = mid;
14            } else {
15                right = mid;
16            }
17        }
18        if(Math.abs(array[left] - target) <= Math.abs(array[right]- target)){
19            return left;
20        }
21        return right;
22    }
23 }
24
```

```
1  // convert the 2D array to 1D array and do binary search
2  public class Solution {
3      public int[] search(int[][] matrix, int target) {
4          if(matrix.length==0||matrix[0].length==0){
5              return new int[] {-1,-1};
6          }
7          int rows=matrix.length;
8          int cols= matrix[0].length;
9          int left = 0;
10         //convert the 2D array to 1D array with rows*cols elements
11         int right = rows*cols-1;
12         while(left <= right){
13             int mid = left + (right - left)/2;
14             // convert the postion in 1d array back to row and col in 2D array.
15             int row = mid /cols;
16             int col = mid % cols;
17             if(matrix[row][col]==target){
18                 return new int[]{row,col};
19             } else if (matrix[row][col] < target){
20                 left = mid +1;
21             } else {
22                 right = mid -1;
23             }
24         }
25         return new int[] {-1,-1};
26     }
27 }
```

```
1  // 561. Find the Kth Element in The Matrix
2  ```
3  Given a matrix, find the Kth index element.
4
5
6
7
8
9  example:
10
11  matrix:
12
13  1 3 4
14
15  5 6 7
16
17  8 9 10
18
19
20
21  k = 4 → return: 6
22  ```
23  public class Solution {
24      public int findElement(int[][] matrix, int k) {
25          int i=k/matrix[0].length; //row
26          int j=k%matrix[0].length;//columns
27          return matrix[i][j];
28      }
29  }
30
```