```java
public class Solution {
  public int[] solve(int[] array) {
    if(array==null||array.length==0){
      return array;
    }
    for(int i =0;i<array.length-1;i++){//why array.length-1
      int min = i;
      for(int j = i+1;j<array.length;j++){
        if(array[min]>array[j]){
          min=j;
        }
      }
      swap(array,i,min);

    }
    return array;
  }
  public void swap(int[] array, int left ,int right){
    int temp = array[left];
    array[left]=array[right];
    array[right]=temp;
  }
}

```

```java
public class Solution {
  public int[] mergeSort(int[] array) {
    if(array==null||array.length==0){
      return array;
    }
    //新建一个helper function
    int[] helper = new int [array.length];
    mergeSort(array,helper,0,array.length-1);
    return array;
  }

  //split
  private void mergeSort(int[] array, int[] helper , int left , int right){
    if(left >= right){
      return;
    }
    int mid = left +(right-left)/2;
    mergeSort(array, helper,left,mid);
    mergeSort(array,helper,mid+1,right);
    merge(array,helper,left,mid,right);
  }
  //merge
  private void merge(int[] array, int[] helper, int left , int mid , int right){
    for (int i = left ; i<= right; i++){
      helper[i] = array[i];
    }
    int leftIndex = left ;
    int rightIndex = mid +1;
    while(leftIndex <= mid&&rightIndex <= right){
      if(helper[leftIndex] <= helper[rightIndex]){
        array[left++] = helper[leftIndex++];
      } else{
        array[left++]=helper[rightIndex++];
      }
    }
    while(leftIndex <= mid){
      array[left++] = helper[leftIndex++];
    }
  }
}

```

```java
public class Solution {
  public int[] quickSort(int[] array) {
    if (array == null || array.length <= 1) {
      return array;
    }
    quickSort(array, 0, array.length - 1);
    return array;
  }
  private void quickSort(int[] array, int left, int right) {
    if (left >= right) {
      return;
    }
    int pivotIndex = patition(array, left, right);
    quickSort(array, left, pivotIndex - 1);
    quickSort(array, pivotIndex + 1, right);
  }
  private int patition(int[] array, int left, int right) {
    int randomNum = left + findRandom(right - left);
    int L = left;
    int R = right - 1;
    swap(array, randomNum, right);
    while (L <= R) {
      if(array[L] <= array[right]) {
        L++;
      } else {
        swap(array, L, R--);
      }
    }
    swap(array, L, right);
    return L;
  }
  private void swap(int[] array, int a, int b) {
    int temp = array[a];
    array[a] = array[b];
    array[b] = temp;
  }
  private int findRandom(int range) {
    Random rand = new Random();
    return rand.nextInt(range + 1);
  }
}

```

```java
public class Solution {
  public int[] rainbowSort(int[] array) {
    if (array.length == 0) {
      return array;
    }
    int i = 0;
    int j = 0;
    int k = array.length - 1;
    while (j <= k) {
      if (array[j] == 0) {
        j++;
      } else if (array[j] == 1) {
        swap(array, j, k--);
      } else {
        swap(array, i++, j++);
      }
    }
    return array;
  }
  private void swap(int[] array, int i, int j) {
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
  }
}

```

```java
public class Solution {
  public int minIndex(int[] array, int i) {
    int min = i;
    for(int j =i+1;j<array.length;j++){
      if(array[min]>array[j]){
        min=j;
      }
    }
    return min;
  }
}
```

```java
public class Solution {
  public int[] merge(int[] array1, int[] array2) {
    // 开拓新的空间，储存合并好的数组
    int[] result = new int[array1.length +array2.length];
    int i =0,j=0,k=0;
    while(i<array1.length&&j<array2.length){
      if(array1[i]<array2[j]){
        result[k]=array1[i];
        i++;
      } else{
        result[k]=array2[j];
        j++;
      }
      k++;
    }
    while(i<array1.length){
      result[k]=array1[i];
      i++;
      k++;
    }
    while(j<array2.length){
      result[k]=array2[j];
      j++;
      k++;
    }
    return result;
  }
}

```

```java
class Solution {
    public int random(int a, int b) {
                Random rand = new Random();
                return a + rand.nextInt(b - a + 1);
    }
}
```

```java
class Solution {
    public void partition(int[] array, int pivotIndex) {
    int pivot = array[pivotIndex];
    swap(array, pivotIndex ,array.length-1);
    int leftBound = 0 ;
    int rightBound = array.length-2;
    while(leftBound <= rightBound){
      if(array[leftBound] < pivot){
        leftBound++;
      }else{
        swap(array,leftBound,rightBound);
        rightBound --;
      }
    }
    swap(array,leftBound,array.length-1);
  }
  private void swap(int[] array, int left , int right){
    int temp = array[left];
    array[left] = array[right];
    array[right]=temp;
  }
}
```