1.二分查找

```
    public static void main(String[] args) {
        System.out.println(binarySerach(new Integer[]{0}, null));
    }

    public static int binarySerach(int[] array, int key){
        if(array == null || array.length == 0){
            return -1;
        }
        int left = 0;
        int right = array.length - 1;
        while(left <= right){
            int mid = (left + right) >> 1;
            if(key < array[mid]){
                right= mid - 1;
            }else if(key > array[mid]){
                left = mid + 1;
            }else{
                return mid;
            }
        }
        return -1;
    }
```

2.快速排序

当初始的序列整体或局部有序，快速排序的性能就会下降，此时，快速排序将退化为冒泡排序。

```
//排序
    public static void quickSort(int[] array, int low, int high){
        if(array == null || array.length == 0) return;
        if(low < high){
            int j = partition(array, low, high);
            quickSort(array, low, j-1);
```

```
            quickSort(array, j+1, high);
        }
    }

    //切分
    public static int partition(int[] array, int low, int high){
        int tmp = array[low];
        int i = low;
        int j = high;
        while(i < j){
            while(i < j && array[j] >= tmp){
                j--;
            }
            if(i < j){
                array[i++] = array[j];
            }
            while(i < j && array[i] < tmp){
                i++;
            }
            if(i < j){
                array[j--] = array[i];
            }
        }
        array[i] = tmp;
        return i;
    }


    public static void main(String[] args) {
        int[] array = {38, 65, 97, 76, 13, 27, 49, 1, -40, -90};
        quickSort(array, 0, array.length-1);
        System.out.println(Arrays.toString(array));
    }
```

3.归并排序

```
public void merge(int[] array, int p, int q, int r){
    int[] left = new int[q-p+1];
    int[] right = new int[r-q];
    for(int i = 0; i < left.length; i++){
        left[i] = array[i+p];
    }
    for(int i = 0; i < right.length; i++){
        right[i] = array[i+q+1];
    }
    int a = 0, b = 0, c = p;
    while(a < left.length && b < right.length){
        if(left[a] <= right[b]){
            array[c++] = left[a++];
        }else{
            array[c++] = right[b++];
        }
    }
    while(a < left.length){
        array[c++] = left[a++];
    }
    while(b < right.length){
        array[c++] = right[b++];
    }
}

public void mergeSort(int[] array, int p, int r){
    if(p < r){
        int q = (p + r) >> 1;
        mergeSort(array, p, q);
        mergeSort(array, q+1, r);
        merge(array, p, q, r);
    }
}
```

```
public void sort(int[] array){
        if(array == null || array.length == 0)return;
        mergeSort(array, 0, array.length-1);
}
```