数据结构与算法

2019年6月21日 10:02

一 、排序算法

最近几天回顾了排序算法，总结一下加深印象。

1、按稳定性分类：

稳定的：冒泡、直接插入、归并

不稳定的：希尔、选择、快排、堆排序、 （桶排序，计数排序、基数排序）

2、按照动作类别：

插入排序： 直接插入排序， 希尔排序

选择排序：简单选择排序， 堆排序

交换排序：冒泡， 快速排序

归并排序：

3、冒泡排序： 时间复杂度O(n^2), 常量空间

思想：比较相邻数据，交换，上浮。

```
public static void bubbleSort(int a[]) {
    System.out.println("**********排序结果**********");
    if (a == null || a.length <= 1)
        return;
    int n = a.length;
    for (int i = 0; i < n - 1; i++)    //n-1趟排序
        for (int j = n - 1; j > i; j--) {  // 从末尾开始上浮最小的数
            if (a[j] < a[j - 1]) {
                a[j] = a[j] + a[j - 1];
                a[j - 1] = a[j] - a[j - 1];
                a[j] = a[j] - a[j - 1];
            }

        }
}
```

4、快排。 时间O(n*logn) ~ O(n^2) ,空间复杂度O(logn)

思想:找一个基准值，判断把数组划分成比这个大的。比这个值小的两部分，每次排序这个基准值就在正确的位置了

// 快排

```
public static void quickSort(int a[]){
    System.out.println("********排序结果**********");
    if(a==null || a.length<=1)
        return;
    int n = a.length;
```

```
        quickSort(a,0,n-1);

    }
    public static void quickSort(int a[],int low,int high){
        if(low>=high)
            return;
        int pos = getPos(a,low,high);
        quickSort(a,0,pos-1);
        quickSort(a,pos+1,high);

    }


public static int getPos(int a[],int low,int high){
        if(low>=high)
            return low;
        int pivot = a[low];
        while(low<high){                    // 要high low相遇才行，否则一直比较
            while(high>low && a[high]>=pivot)
                high--;
            a[low] = a[high];
            while(low<high && a[low]<=pivot)
                low++;
            a[high] = a[low];
        }

        a[low] = pivot;
        return low;


}
```

5、归并排序： 时间O(n*logn) ,空间复杂度O(n)

思想： 分割 - 排序 -合并

```
static int t[] ;  //辅助空间O(n)
    public static void twoWaySort(int a[]) {
        System.out.println("*******排序结果********");
        if(a==null || a.length<=1)
            return;
        int n = a.length;
        t = new int[n];
        mergeSort(a,0,n-1);
    }
    //递归
    public static void mergeSort(int a[] ,int low, int high){
```

```
    if(low>=high)
        return;
    int mid = low + (high -low) /2;
    mergeSort(a,low,mid);
    mergeSort(a,mid+1,high);
    merge(a,low,mid,high);
}
// 合并两个有序数组
public static void merge(int a[],int low,int mid,int high){
    int i =low , j= mid+1;
    int k  = low;
    while(i<=mid && j<=high){    // 合并到新的数组中
        while(i<= mid && j<=high && a[i]<=a[j])
            t[k++] = a[i++];
        while(i<=mid && j<=high && a[j]<a[i])
            t[k++] = a[j++];

    }
    // 还存在一个为空的
    while(i<=mid)
        t[k++] =a[i++];
    while(j<=high)
        t[k++] = a[j++];
    // 利用辅助空间存的值返回回去原来的数组
    for(k=low;k<=high;k++)
        a[k] = t[k];
}
```

6.简单选择排序

思想： 每次选择一个最小值，然后交换到最前面，最稳定的算法，时间都为O(n^2)

```
// 简单选择排序
public static void simpleSelect(int a[]){
    if(a==null || a.length<=1)
        return;
    int n = a.length;
    for(int i=0;i<n-1;i++){  //只需要选择n-1次即可
        int min = i;
        for(int j=i+1;j<n;j++){
            if(a[j]<a[min])
                min = j;

        }
        //交换a[i]和a[min] ， 不能用 +-的那种交换方式
       int tmp = a[i];
        a[i] =a[min];
        a[min] = tmp;

    }


}
```