数据结构与算法

各种排序算法的比较，时间空间复杂度

参考十大经典排序算法总结（Java语言实现）_wang的博客-CSDN博客排序算法java

排序算法	平均时间复杂度	最好情况	最坏情况	空间复杂度	排序方式	稳定性
冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	In-place	稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	In-place	不稳定
插入排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	In-place	稳定
希尔排序	$O(n \log n)$	$O(n \log^2 n)$	$O(n \log^2 n)$	$O(1)$	In-place	不稳定
归并排序	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$	Out-place	稳定
快速排序	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$	In-place	不稳定
堆排序	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	In-place	不稳定
计数排序	$O(n + k)$	$O(n + k)$	$O(n + k)$	$O(k)$	Out-place	稳定
桶排序	$O(n + k)$	$O(n + k)$	$O(n^2)$	$O(n + k)$	Out-place	稳定
基数排序	$O(n \times k)$	$O(n \times k)$	$O(n \times k)$	$O(n + k)$	Out-place	稳定

1. 冒泡排序（bubble sort）

   过程描述：比较相邻的两个元素，如果第一个比第二个大，那么就交换，依次类推，一直比到最后一对，这样得到的最后一个元素就是最大的；然后重复这个过程，得到倒数第二个元素，依次类推。

   时间复杂度为O(n2)

```
void bubbleSort(int[] array){
    int len=array.length;
    for(int i=0; i<len; i++){
        for(int j=0; j<len-i-1; j++){
            if(array[j]>array[j+1]){
                int temp=array[j];
                array[j]=array[j+1];
                array[j+1]=temp;
            }
        }
    }
}
```

2. 选择排序

   过程描述：首先在未排序序列中找到一个最小的元素，存放到排序序列的起始位置，然后再从剩下未排序元素中继续找到一个最小的元素，放在已排序序列的末尾，直到所有元素均排序完毕。

```
int[] selectionSort(int[] array){
    int len=array.length;
```

```
    for(int i=0; i<len; i++){
        int min=array[i];
        int index=i;
        for(int j=i; j<len; j++){
            if(min>array[j]){
                min=array[j];
                index=j;
            }
        }
        int temp=array[index];
        array[index]=array[i];
        array[i]=temp;
    }
    return array;
}
```

3. 插入排序

过程描述：在要排序的一组数中，假设前n-1个数已经排好序，然后将第n个与前面的n-1个数依次比较，放在合适的位置，使得这n个数也是排好顺序的。依次循环，直到全部排好顺序。

```
int[] insertionSort(int[] array){
    int len=array.length;
    for(int i=1; i<len; i++){
        int j=i;
        while(j-1>=0 && array[j]<array[j-1]){
            int temp=array[j];
            array[j]=array[j-1];
            array[j-1]=temp;
            j--;
        }
    }
    return array;
}
```

4. 希尔排序
5. 归并排序

过程描述：把已经有序的子序列合并，得到完全有序的序列。归并排序一般指的是二路归并，具体流程为：把一个长度为n的输入序列分成两个长度为n/2的子序列，对这两个子序列分别进行归并排序，然后把已经排好序的两个子序列进行合并。

```
void mergeSort(int[] array, int left, int right){
    if(left>=right) return;
    int mid=(left+right)/2; //mid 用作前一个序列的最后一个值
    mergeSort(array, left, mid);
    mergeSort(array, mid+1, right);
    merge(array, left, mid, right);

}
void merge(int[] array, int left, int mid, int right){
    int[] nums=new int[right-left+1];
    int i=0;
    int l=left, r=mid+1;
    while(l<=mid && r<=right){
        if(array[l]<=array[r]){
            nums[i]=array[l];
```

```
            l++;
        }
        else{
            nums[i]=array[r];
            r++;
        }
        i++;
    }
    while(l<=mid){
        nums[i]=array[l];
        l++;
        i++;
    }
    while(r<=right){
        nums[i]=array[r];
        r++;
        i++;
    }
    for(int i=0; i<nums.length; i++){
        array[left+i]=nums[i];
    }
}
```

6. 快速排序

   过程描述：从数组中找一个元素作为pivot，然后对数组进行重新排序，所有小于该pivot的元素放在pivot的前面，大于该pivot的元素放在pivot的后面，然后再递归的对小于和大于该pivot的两个子序列进行排序。

```
void quickSort(int array[], int left, int right){
    if(left>=right) return;
    int mid=(left+right)/2;
    int pivot=array[mid];
    int i=left, j=right;
    while(i<=j){
        while(i<=j && array[i]<=pivot){
            i++;
        }
        while(i<=j && array[j]>=pivot){
            j--
        }


    }
}
```

有没有自己实现过什么复杂的数据结构，需要注意哪些细节

1. 创建包含min函数的栈

介绍一些基础的数据结构

1. 数组

   数组中每个元素的位置都会有一个标号，作为这个元素的索引，数组可以有一维数组也可以有多维数组

2. 栈

   栈是一种运算受限的线性表，只能在栈顶进行插入和删除的操作，栈中的元素采用后进先出的方式，数据进栈时要根据规则压到栈的底部，取出数据的时候会先取出上面的元素。数据和链表都可以生成栈。

3. 队列

   队列与栈类似，都是采用线性结构存储数据，区别是队列采用先进先出的方式，数组和链表都可以生成队列。

4. 链表

   链表也是线性结构，是一系列节点组成的链，链表中的数据是不连续的，每一个节点保存了数据以及指向下一个节点的指针。链表头指针指向第一个节点，如果链表为空，则头指针为空或者为null。

5. 图

   图由多个节点组成，节点之间可以由边进行连接组成一个网络，边可能会有权值。图可能有两种形式来表示：邻接矩阵，邻接表。

6. 树

   树(是一个分层的数据结构，由节点和连接节点的边组成。树是一种特殊的图，它与图最大的区别是没有循环。

7. 哈希表

8. 堆

   堆是一个完全二叉树，所有父节点都满足大于等于其子节点的堆叫大根堆，所有父节点都满足小于等于其子节点的叫小根堆。堆通常用数组来表示，父节点和子节点的关系通过数组下标来确定。

   > 找父节点：parent(i)=(i-1)/2
   >
   > 父节点找左子：leftChild(i)=i*2+1
   >
   > 父节点找右子：rightChild(i)=i*2+2

哪些树必须是二叉树

可选项有b树，avl树，b+树，红黑树

ArrayList和LinkedList的区别

常问的数组问题：

首先放个链接 如何在最短的时间内搞定数据结构和算法，应付面试? - 知乎 (zhihu.com)

1. 找到数组第二小的元素

   维护两个变量，第一小和第二小 O(n)时间复杂度

2. 找到数组中第一个没有重复的整数

   维护一个哈希表，记录每个数字对应的出现位置，再维护一个boolean数组，和哈希表中的位置相对应，存储该位置的数字是否重复出现 O(n)时间复杂度

3. 重新排列数组中的正数和负数，使正数排在后面，但不改变原有顺序