LRU

2021年3月21日 10:39

Author：pengkang
pkdlmu@163.com

```
import java.util.*;

public class Solution {
    private Map<Integer, Node> map = new HashMap<>();
    private Node head = new Node(-1,-1);
    private Node tail = new Node(-1,-1);
    private int k;
    public int[] LRU (int[][] operators, int k) {
        this.k = k;
        head.next = tail;
        tail.prev = head;
        int len = (int)Arrays.stream(operators).filter(x -> x[0] == 2).count();
        int[] res = new int[len];
        for(int i = 0, j = 0; i < operators.length; i++) {
            if(operators[i][0] == 1) {
                set(operators[i][1], operators[i][2]);
            } else {
                res[j++] = get(operators[i][1]);
            }
        }
        return res;
    }

    private void set(int key, int val) {
        if(get(key) > -1) {
            map.get(k).val = val;
        } else {
            if(map.size() == k) {
                int rk = tail.prev.key;
                tail.prev.prev.next = tail;
                tail.prev = tail.prev.prev;
                map.remove(rk);
            }
            Node node = new Node(key, val);
            map.put(key, node);
            moveToHead(node);
        }
    }

    private int get(int key) {
        if(map.containsKey(key)) {
            Node node = map.get(key);
            node.prev.next = node.next;
            node.next.prev = node.prev;
            moveToHead(node);
            return node.val;
        }
        return -1;
    }

    private void moveToHead(Node node) {
        node.next = head.next;
```

```
        head.next.prev = node;
        head.next = node;
        node.prev = head;
    }

    static class Node{
        int key, val;
        Node prev, next;
        public Node(int key, int val) {
            this.key = key;
            this.val = val;
        }
    }
}
```

compareTo

2021年3月21日 10:27

排序，0，实现compareable API

Comparable接口

1：所有可以 “排序” 的类都实现了java.lang.Comparable接口，Comparable接口中只有一个方法。
2：public int compareTo(Object obj) ;
该方法:
返回 0 表示 this == obj
返回整数表示 this > obj
返回负数表示 this < obj
3:实现了 Comparable 接口的类通过实现 comparaTo 方法从而确定该类对象的排序方式。

屏幕剪辑的捕获时间: 2021/3/21 15:06

a.comparaTo(b)

排序算法

2021年3月21日 10:27

快速排序

```
import java.util.*;


public class Solution {
    /**
     * 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即
可
     * 将给定数组排序
     * @param arr int整型一维数组 待排序的数组
     * @return int整型一维数组
     */
    public int[] MySort (int[] arr) {
        // write code here
        quickSort(arr, 0, arr.length -1 );
        return arr;

    }
    //快排
    private void quickSort(int[] arr, int start, int end){
        if(start >= end){
            return;
        }
        int standard = arr[start];
        int i = start;
        int j = end;
        while(i<j){
            while(i<j && arr[j]> standard){
                j--;
            }
            while(i<j && arr[i] <= standard){
                i++;

            }
            swap(arr,i, j);
        }
        swap (arr, i ,start);
        //递归
        quickSort(arr, start, i-1);
        quickSort(arr, i+1, end);

    }
    private void swap(int[] arr, int i , int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

冒泡on2

2021年3月21日 15:05

1，冒泡，两两比较。交换On2

```
packagecom.ucas.sort;

publicclassBubbleSort{

//比较方法
privatestaticbooleangreater(Comparablev,Comparablew){
returnv.compareTo(w)>0;
//int res=v.compareTo(w);res>0,true;
}

//交换值；
privatestaticvoidswap(Comparable[]a,inti,intj){
Comparabletemp;
temp=a[i];
a[i]=a[j];
a[j]=temp;
}

//sort方法里面调用前两个方法，每个冒泡会少一个元素
publicstaticvoidsort(Comparable[]a){
for(inti=a.length-1;i>0;i--){
for(intj=0;j<i;j++){
//比较交换
if(greater(a[j],a[j+1])){
swap(a,j,j+1);
}
}
}
}


}
```

选择排序On2

2021年3月21日 14:41

每次选择最小的，放在合适的位置，
第一次选择最小的，放在第一个
选择第二小的，放在第二个
。。。。。。

On2

选择排序API设计：

类名	Selection
构造方法	Selection()：创建Selection对象
成员方法	1.public static void sort(Comparable[] a)：对数组内的元素进行排序 2.private static boolean greater(Comparable v,Comparable w):判断v是否大于w 3.private static void exch(Comparable[] a,int i,int j)：交换a数组中，索引i和索引j处的值

```
for(int i= 0; i< a.length -2; i++){
        //记录最小元素的索引，默认第一个
        int minIndex = i;
        for(int j = i+1; j< a.length-1;j++){
           if(greater(a[minIndex],a[j])){
              minIndex = j;
           }

        }

        swap(a, i, minIndex);
    }
```

插入排序on2

2021年3月21日 14:59

扑克牌，已排序和未排序的两种。。。
默认0出是已排序的，未排序的第一个倒叙遍历已排序的，比较

插入排序API设计：

类名	Insertion
构造方法	Insertion()：创建Insertion对象
成员方法	1.public static void sort(Comparable[] a)：对数组内的元素进行排序 2.private static boolean greater(Comparable v,Comparable w):判断v是否大于w 3.private static void exch(Comparable[] a,int i,int j)：交换a数组中，索引i和索引j处的值

屏幕剪辑的捕获时间: 2021/3/21 15:02

```
for(int i = 1; i<a.length; i++){
        //内循环，倒叙遍历
        for(int j = i; j> 0; j--){
           //比较j,j-1,
           if(greater(a[j-1],a[j])){
              swap(a, j-1,j);
           }else{
              break;
           }
        }
     }
```

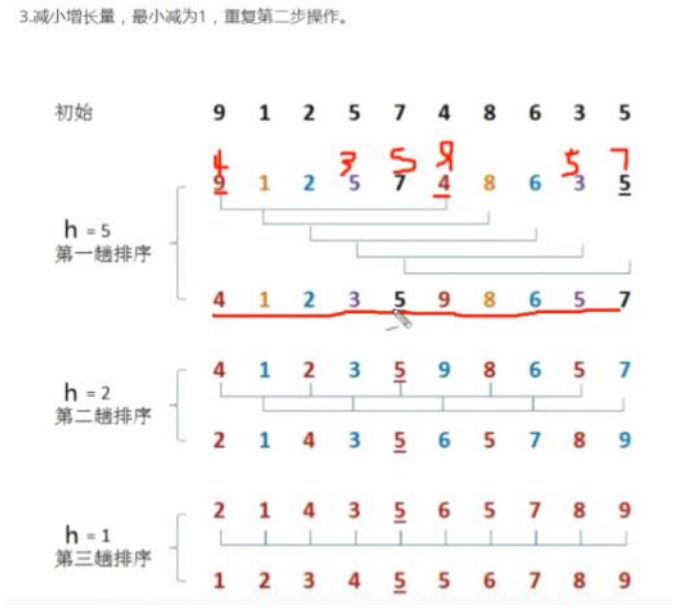
高级排序，希尔排序0n1.3

2021年3月21日 15:32

减少插入次数，

分组，

缩小分组



屏幕剪辑的捕获时间: 2021/3/21 15:39

屏幕剪辑的捕获时间: 2021/3/21 15:42

```
public static void sort(Comparable [] a){
    int h = 1;
    //确定h
    while(h< 5){
       h= 2*h +1;
    }


    while(h>=1){
       //
       for(int i = h; i< a.length;i++){
          for(int j = i; j>=h;j-=h){
             if(greater(a[j-h], a[j])){
                //交换
                swap(a, j-h, j);
             }else{
                break;
             }
          }
       }



       h= h/2;
    }


 }
```

递归

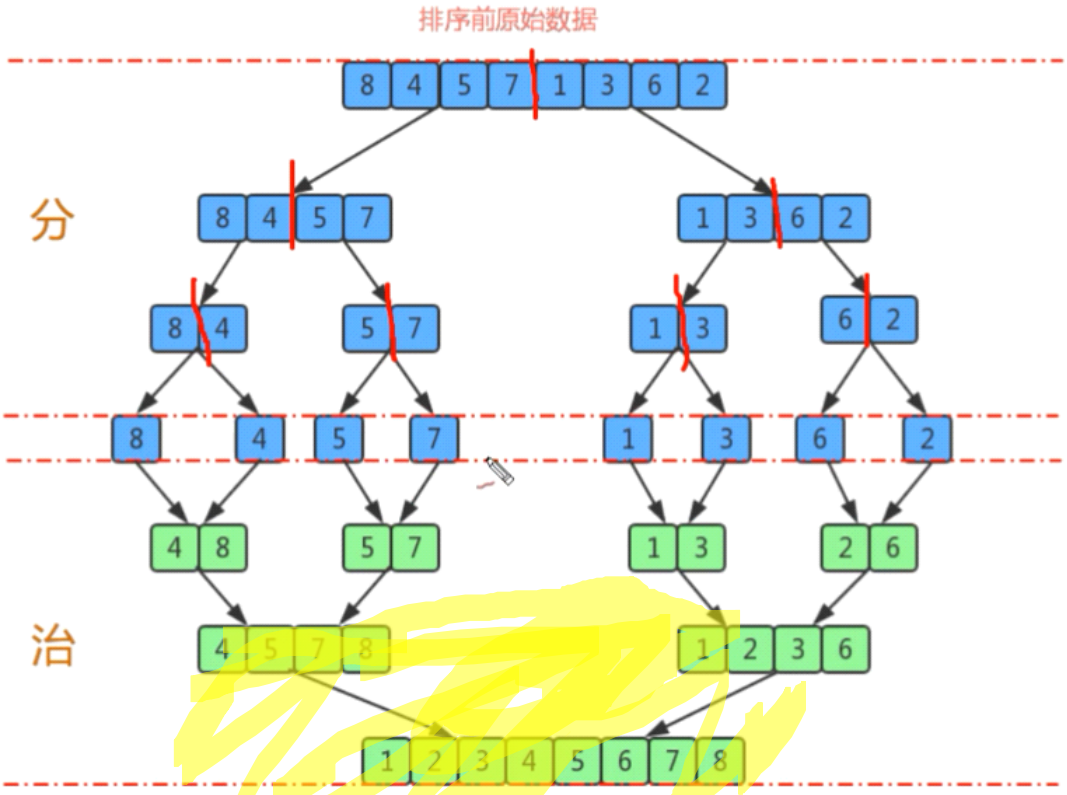
2021年3月21日 16:12

层级太深，内存栈溢出

归并排序Onlogn

2021年3月21日　　16:13

归并



屏幕剪辑的捕获时间: 2021/3/21 16:13

分，等分的
长度除2.

归并的时候怎么排序？

归并排序API设计：

类名	Merge
构造方法	Merge()：创建Merge对象
成员方法	1.public static void sort(Comparable[] a)：对数组内的元素进行排序 2.private static void sort(Comparable[] a, int lo, int hi)：对数组a中从索引lo到索引hi之间的元素进行排序 3.private static void merge(Comparable[] a, int lo, int mid, int hi):从索引lo到所以mid为一个子组，从索引mid+1到索引hi为另一个子组，把数组a中的这两个子组的数据合并成一个有序的大组（从索引lo到索引hi） 4.private static boolean less(Comparable v,Comparable w):判断v是否小于w 5.private static void exch(Comparable[] a,int i,int j)：交换a数组中，索引i和索引j处的值
成员变量	1.private static Comparable[] assist：完成归并操作需要的辅助数组

屏幕剪辑的捕获时间: 2021/3/21 16:16

屏幕剪辑的捕获时间: 2021/3/22 23:05

三个指针，把归并好的放入辅助数组中；
比较p1,p2, 小的后移
I 后移，

快排Onlogn

2021年3月22日 23:47

快速排序API设计:

类名	Quick
构造方法	Quick()：创建Quick对象
成员方法	1.public static void sort(Comparable[] a)：对数组内的元素进行排序 2.private static void sort(Comparable[] a, int lo, int hi)：对数组a中从索引lo到索引hi之间的元素进行排序 3.public static int partition(Comparable[] a,int lo,int hi):对数组a中，从索引 lo到索引 hi之间的元素进行分组，并返回分组界限对应的索引 4.private static boolean less(Comparable v,Comparable w):判断v是否小于w 5.private static void exch(Comparable[] a,int i,int j)：交换a数组中，索引i和索引j处的值

切分原理：

把一个数组切分成两个子数组的基本思想：

1.找一个基准值，用两个指针分别指向数组的头部和尾部；

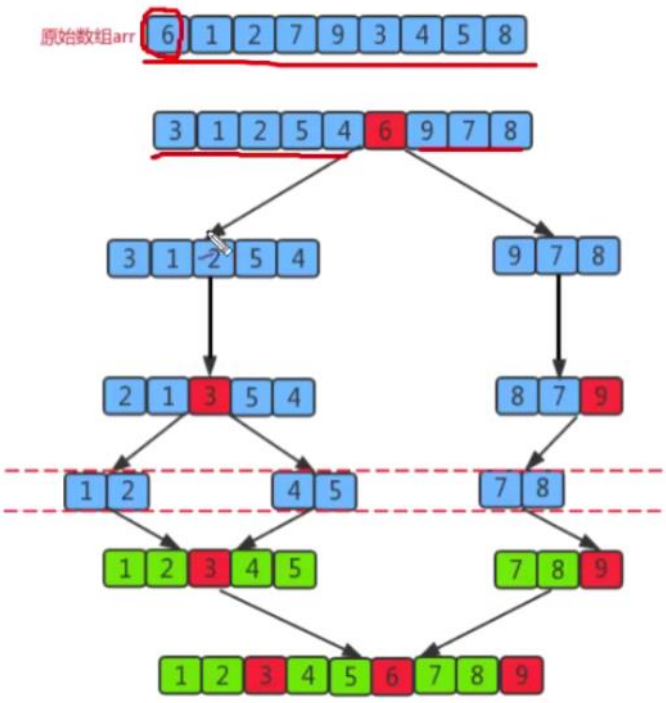
2.先从尾部向头部开始搜索一个比基准值小的元素，搜索到即停止，并记录指针的位置；

3.再从头部向尾部开始搜索一个比基准值大的元素，搜索到即停止，并记录指针的位置；

4.交换当前左边指针位置和右边指针位置的元素；

5.重复2,3,4步骤，直到左边指针的值大于右边指针的值停止。

屏幕剪辑的捕获时间: 2021/3/22 23:48

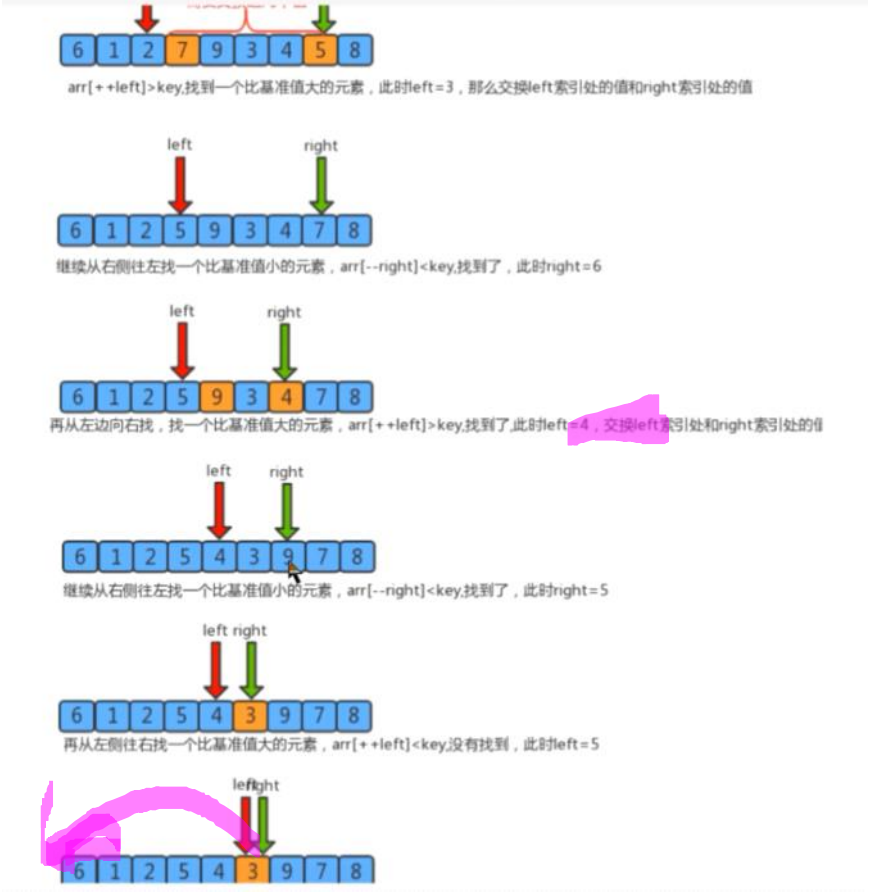


屏幕剪辑的捕获时间: 2021/3/22 23:47

核心代码

两个指针，left,指向第一个，right,指，

一个右移动，



屏幕剪辑的捕获时间: 2021/3/23 0:04

堆

2021年7月8日 星期四　　上午12:34

如果是海量数据中查找出最小的k个数，那么这种办法是效率很低的。**接下来介绍另外一种算法：**

** 创建一个大小为k的数组，遍历n个整数，如果遍历到的数小于大小为k的数组的最大值，则将此数与其最大值替换。**

由于每次都要拿n个整数和数组中的最大值比较，所以选择大根堆这一数据结构(大家要分清楚大根堆这一数据结构和堆排序之间的区别：堆排序是在大根堆这一数据结构上进行排序的一种排序算法，一个是数据结构，一个是算法)