数据结构与算法作业 4

杨庆龙 1500012956

1. (1)设归并路数为 k，则三趟归并可归并k^3个归并段。又由共有 100 个归并段得，k>5,则归并路数至少为 6.

(2)归并是一个 k 个文件入，1 个文件出的过程，所以最多同时输入 12 个文件，又有 12*12=144>100，则至少需要两趟。

2. (1)

```
//定义一个既能表示数值又能表示出现该值出现次数的结构体
struct node{
   Value value;
   int count;
   node(Value _value = 0,int _count = 0):value(_value),count(_count) {}
}

node* merge(Value* input,int n,int k) {
   node* output;
   int count = 0;
   SearchCountTree<Node> sc_tree = SearchCountTree<Node>();//建立一个以 node::value 排序的搜索二叉树，并在插入已出现数据时让相应 node::count 加一，同时完成排序和计数的工作。
   for(int i = 0;i < n;i ++) {
      sc_tree.insert(input[i]);
   }
   output = sc_tree.mid();//中序遍历即可得到排好序的结果
   return output;
}
```

搜索二叉树最好情况下的进行查询操作时间复杂度为 O(logn)，对于 n 次查找，总耗时为 O(nlogn)。而中序遍历的时间复杂度为 O(n)，则时间复杂度为Ω(nlogn)

(2)

```
Value* expand(node* input,int n,int k) {
   Value* output = new Value[n];
   int point = 0;//用于指示位置
   for(int i = 0;i < k;i ++) {
      for(int p = 0;p < input[k].count;p ++) {
         output[point ++] = input[k].value;//按照计数器结果，重复 node::value 数次
      }
   }
   return output;
}
```

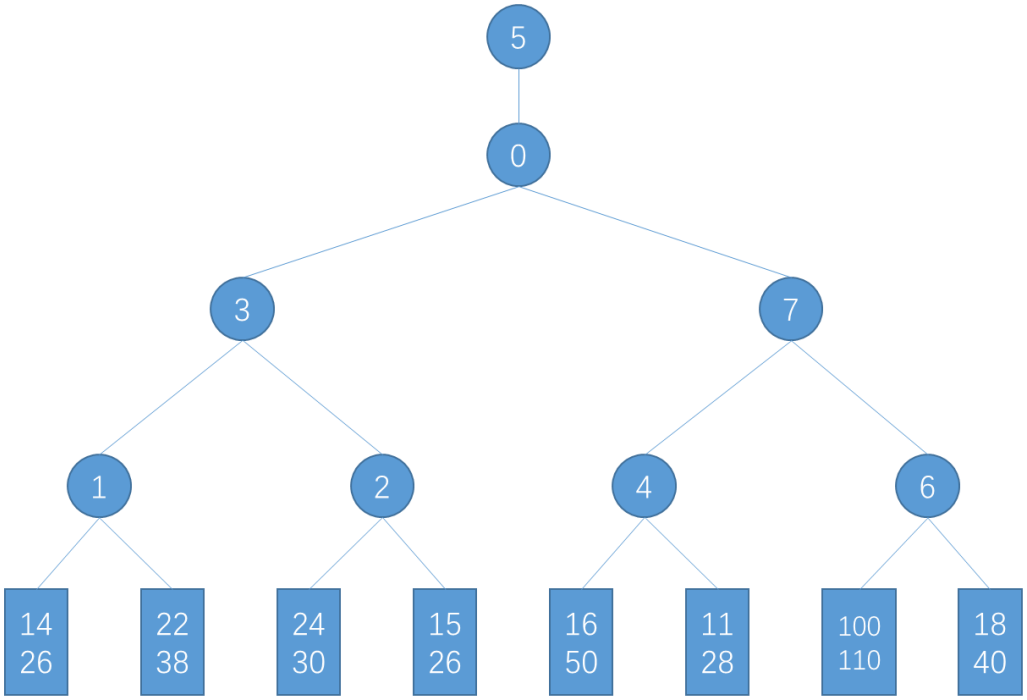
3.快排第一趟结果为{22,19,13,6,24,38,43,32}

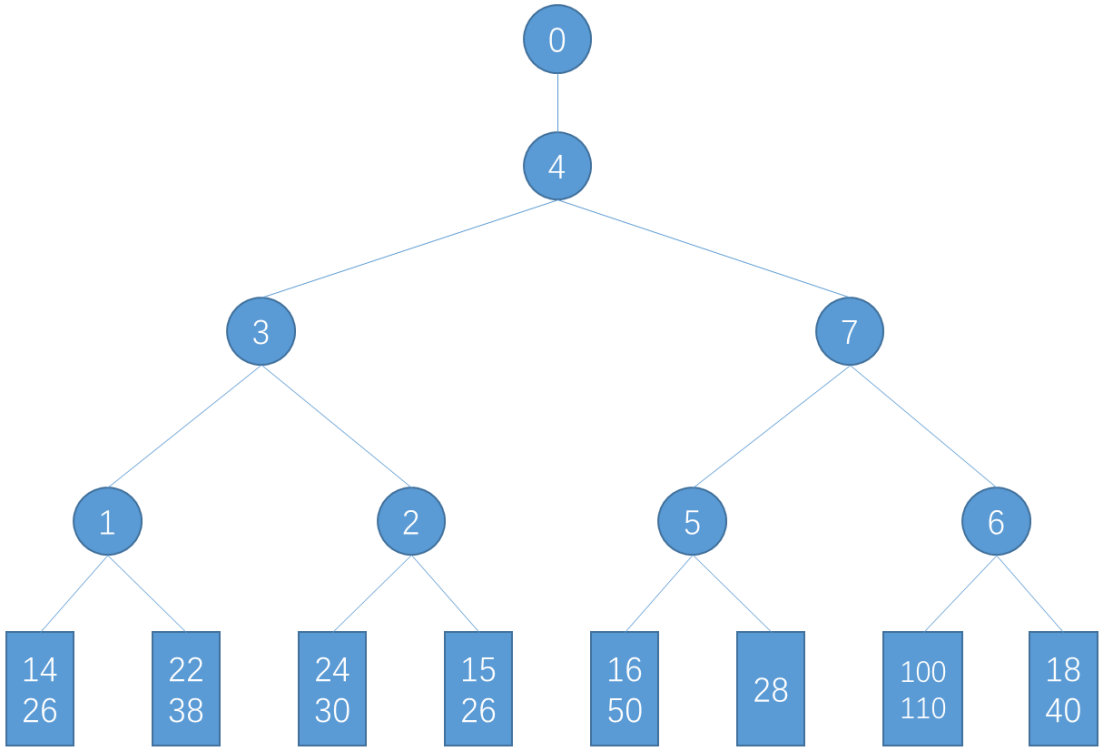
最大堆的序列为{43,38,32,22,24,6,13,19}

快排最坏情况为Θ(n^2)，最大堆的最坏情况为Θ(nlogn)，快排明显劣于最大堆排序

4.
第一遍:{31,22,212,102,14,435,5,346,157,568,8}
第二遍:{102,5,8,212,14,22,31,435,346,157,568}
第三遍:{5,8,14,22,31,102,157,212,346,435,568}

5.
```
int getNOk(int* input,int start,int end,int k) {
    int tempValue = input[start];//选取第一个值为轴
    int p = start;
    int q = end - 1;
    for(;p > q;p --) {//外层循环为右到左的指针
        if(input[p] < tempValue) {
            input[q] = input[p];
            for(;q < p;q ++) {//内层循环为左到右的指针
                if(input[q] > tempValue) {
                    input[p] = input[q];
                    break;
                }
            }
        }
    }
    if(p == k)
        return input[k];//若在 k 处相遇，即可得到第 k 小的数
    else if(p < k)
        return getNOK(input,p,end,k);//第 k 小的数在当前分界点的右侧
    else
        return getNOK(input,start,p,k);//第 k 小的数在当前分界点的左侧
}
```
6.


5
0
3
7
1
2
4
6
14
26
22
38
24
30
15
26
16
50
11
28
100
110
18
40


0
4
3
7
1
2
5
6
14
26
22
38
24
30
15
26
16
50
28
100
110
18
40