# Java面试题总结

## 算法

### 二分查找算法

二分查找又称折半查找

二分查找要求：（1）必须采用顺序存储结构 （2）.必须按关键字大小有序排列。

原理：将数组分为三部分，依次是中值（所谓的中值就是数组中间位置的那个值）前，中值，中值后；将要查找的值和数组的中值进行比较，若小于中值则在中值前 面找，若大于中值则在中值后面找，等于中值时直接返回。然后依次是一个递归过程，将前半部分或者后半部分继续分解为三部分。

### 红黑树算法

红黑树(Red-Black Tree，简称R-B Tree)，它一种特殊的二叉查找树。

一个数组长度为N，K是指定的整数，你要找出其中出现次数大于 N/K 的所有数.

/\*\*

\*

\*\*/

public void method(int[] arr, int K){

    Map<Integer, Integer> countMap = new HashMap<Integer, Integer>();

    int N = arr.length();

    if(K == 0){

        throw Exception("K  is  invalid");

        return；

    }

    double count = N/K;

    for(int i = 0, len = N; i <  len; i++){

        Integer valueCount = countMap.get(arr[i]);

        if(null == valueCount){

             countMap.put(arr[i], 1);

         }

         if(null != valueConut){

             countMap.put(arr[i], countMap.get(arr[i])  + 1);

         }

    }

    boolean isFind = false;

    for(Integer key: countMap.keySet()){

         if(countMap.get(key)  > count) {

              System.out.println("exist number is  upper  N/K : " + key);

              isFind = true;

          }

     }

     if(!isFind){

          System.out.println("ddddd")

      }

}

计算出树的左右节点的最长距离

                   1

                2      3

             4    5  6   7

public int method(Node head, int sum){

    if(head == null){

         return 0;

     }

    if(head.getLeftNode() == null  && head.getRightNode == null){

        return sum + 1;

    }

    return method(head.getLeftNode() , sum)  +  method(head.getLeft(), sum);

}