武汉大学计算机学院

本科生课程设计报告

XXX 系统总体设计与实现

专 业 名 称 :计算机科学与技术

课 程 名 称 :嵌入式系统设计

指 导 教 师 一:张三　副教授

学 生 学 号 :2023XXXXXXXXX

学 生 姓 名 :XXX

二〇二五 年 十 月

目录

1 分治法

1.1 数组中的第 K 个最大元素

问题描述: 给定整数数组 nums 和整数 k，请返回数组中第 k 个最大的元素。

算法思想: 随机给一个基准值 piviot, 容易得到大于 piviot, 等于 piviot , 小于 piviot 三部分的数量，从而将目标的“数组中的第 K 个最大元素”所可能出现的范围缩小到三者之一并继续分支，如果在等于 piviot 的区间内则直接返回

伪代码:

输入：数组 num, 整数 k。

输出：第 k 个最大元素的值。

```cpp
int k_largest(vector<int> num, int k){
  return quick_sort(num, k);
}

int quick_sort(vector<int>num, int k){
  int idx = rand() % num.size();
  int x = num[idx];
  vector<int> smalls, equals, bigs;
  for (int ele in num){
    if (ele < x){
      smalls.push_back(ele);
    }else if(ele == x){
      equals.push_back(ele);
    }else{
      bigs.push_back(ele);
    }
  }
  if (k < bigs.size()){
    return quick_sort(bigs, k);
  }else if(k < bigs.size() + equals.size()){
    return equals[0];
  }else{
    return quick_sort(smalls, k);
  }
}

```

时间复杂度分析

对于长度为 N 的数组执行哨兵划分操作的时间复杂度为 $O(N)$。而向下分支子数组的平均长度是 $\frac{N}{2}$, 则平均的总时间是 $N + \frac{N}{2} + ... + 1 = 2N - 1 = O(N)$

1.2 寻找峰值

问题描述:峰值元素是指其值严格大于左右相邻值的元素。给你一个整数数组 nums，找到峰值元素并返回其索引。数组可能包含多个峰值，在这种情况下，返回 **任何一个峰值** 所在位置即可。假设 nums[−1] = nums[n] = -∞

算法思想:在边界不为峰值点时，能确定中间必有峰值点,从而在更小的区间分治，直到真正找到峰值元素。

伪代码:

cpp

```cpp
int findPeakElement(vector<int>& arr) {
    int n = arr.size();
    if(n == 1) return 0;
    if(arr[0] > arr[1]) return 0;
    else if(arr[n-1] > arr[n-2]) return n - 1;
    /*在边界不为峰值点时，能确定中间必有峰值点*/
    int l = 1, r = n - 2, ans = 0;
    while(l <= r){
        int m = l + (r - l) / 2;
        if(arr[m-1] > arr[m]){
            r = m - 1;
        }else if(arr[m+1] > arr[m]){
            l = m + 1;
        }else{
            ans = m;
            break;
        }
    }
    return ans;
}
```

时间复杂度分析: 每次二分后，待寻找的范围[l, r]都缩小一半，易知时间复杂度为$\Theta(\log n)$

1.3 逆序对数量

问题描述:：对于给定的一段正整数序列，逆序对就是序列中 $a_i > a_j$ 且 $i < j$ 的有序对。

算法思想:如果我们想要将一个序列排成从小到大有序的，那么每次划分后合并时左右子区间都是从小到大排好序的，我们只需要统计右边区间每一个数分别会与左边区间产生多少逆序对即可。由于两边分别有序，所以求解两边区间产生的逆序对数量能够很快。

伪代码:

输入: 第一行，一个数 n，表示序列中有 n 个数。第二行 n 个数，表示给定的序列。序列中每个数字不超过 10^9。

输出:输出序列中逆序对的数目

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long

int n;
const int MAX = 5e5 + 1;
int arr[MAX]={0};
int help[MAX] = {0};
int compute();
int f(int l, int r);
int merge(int l, int m, int r);
signed main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>arr[i];
    }
    cout<<compute()<<endl;
    return 0;
}
int compute(){
    return f(1, n);
}
int f(int l, int r){
    if(l==r){
        return 0;
    }
    int mid = l + (r - l) / 2;

    return f(l, mid) + f(mid+1, r) + merge(l, mid, r);
}
int merge(int l, int m, int r){
    int ans = 0;
    for(int i=m, j=r; i>=l;i--){
        while(j>=m+1&&arr[i]<=arr[j]){
            j--;
        }
        ans += j-m;
    }
    int i = l;
    int a = l, b = m+1;
    while(a<=m&&b<=r){
        help[i++] = arr[a]<=arr[b]?arr[a++]:arr[b++];
    }
    while(a<=m){
        help[i++] = arr[a++];
    }
    while(b<=r){
        help[i++] = arr[b++];
    }
    for(int i=l;i<=r;i++){
```

```
51          arr[i] = help[i];
52      }
53      return ans;
54  }
55
56
```

时间复杂度分析：

f 函数的时间复杂度$T(n)$满足(merge 函数的执行次数为 n):

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

求得 $T(n) = \Theta(n \log n)$