**算法分析与设计实验报告**

**第 1 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 邹林壮 | 学号 | 202208040412 | | 班级 | 计算机科学与技术（拔尖班）2201 |
| 时间 | 2024.10.19 | 地点 | 院楼432 | | | |
| 实验名称 | 分治法查找最大最小值 | | | | | |
| 实验目的 | 通过上机实验，要求掌握分治算法的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 利用分治法查找数组元素的最大值和最小值，并计算出程序运行所需要的时 间 | | | | | |
| 实验步骤 | 1. 确定输入输出，输入为数组a，输出为最大最小值，约束为使用分治法 2. 首先将问题进行分解，具体可以分解到数组中只有一个元素的情况，即l==r的时候，可以确定最小值和最大值均为该元素；当数组中有两个元素以上时，将其划分为两部分，分解得到每一个部分的最大值和最小值，然后将两部分依次合并，即二者之中最小的作为最小值，最大的作为最大值，直到复原为整个数组，得到最终的最大值和最小值 | | | | | |
| 关键代码 | void findmaxmin(vector<int>& a,int l,int r,int& maxi,int& mini){  if(l==r){  maxi=a[l];  mini=a[l];  return;  }else{  int max1,min1,max2,min2;  int mid=l+(r-l)/2;  findmaxmin(a,l,mid,max1,min1);  findmaxmin(a,mid+1,r,max2,min2);  maxi=max(max1,max2);  mini=min(min1,min2);  return;  }  } | | | | | |
| 测试结果 | 1. 保证了正确性IMG_256   2)时间复杂度为O(n)  公式推到为T(n)=2T(n/2)+O(1),经过推导可以得到这个结论(每次递归调用都会将数组分成两半，所以递归的深度是 log(n)，每层推导执行了少量的比较操作（比较两个最大值和两个最小值）来合并两个子问题的解，这一步的操作是常数时间，即O(1)。最终递归调用中，当l==r时，函数直接返回当前元素作为最大值和最小值，这一步的操作是常数时间，即O(1))  空间复杂度O(n)（其中数组占用O(n),递归实现每次开辟两个空间，但是递归深度为logn，因此最终也是占用O(n))  3)构造了三个规模的数据集：10，10000，1000000  IMG_256 | | | | | |
| 实验心得 | 1. 递归是分治策略的一种实现方式。通过将大问题分解成小问题，递归可以有效地自顶向下解决这些问题。 2. 很好地反映了分治法先分解为子问题，即只剩下一个元素，其最大最小值均为该元素，然后不断进行合并，得到多个元素下的最大最小值，得到最终问题的解 3. 也在一定程度上反映了该问题不能继续优化，查找一个数组中的最大最小值时间复杂度下界为O(n) | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

**问题代码：  
#include<bits/stdc++.h>**

**using namespace std;**

**const int N=5e7;**

**vector<int> a(N+1);**

**void findmaxmin(vector<int>& a,int l,int r,int& maxi,int& mini){**

**if(l==r){**

**maxi=a[l];**

**mini=a[l];**

**return;**

**}else{**

**int max1,min1,max2,min2;**

**int mid=l+(r-l)/2;**

**findmaxmin(a,l,mid,max1,min1);**

**findmaxmin(a,mid+1,r,max2,min2);**

**maxi=max(max1,max2);**

**mini=min(min1,min2);**

**return;**

**}**

**}**

**int main(){**

**int n;**

**cin>>n;**

**for(int i=0;i<n;i++)cin>>a[i];**

**int maxi=a[0],mini=a[0];**

**findmaxmin(a,0,n-1,maxi,mini);**

**cout<<"max:"<<maxi<<" min:"<<mini;**

**}**

**计算时间代码：**

**#include <iostream>**

**#include <fstream>**

**#include <vector>**

**#include <cstdlib>**

**#include <ctime>**

**#include <chrono>**

**using namespace std;**

**using namespace chrono;**

**const int N = 5e7; // 最大规模的数据量**

**vector<int> a(N+1);**

**// 查找最大值和最小值的递归函数**

**void findmaxmin(vector<int>& a, int l, int r, int& maxi, int& mini) {**

**if (l == r) {**

**maxi = a[l];**

**mini = a[l];**

**return;**

**} else {**

**int max1, min1, max2, min2;**

**int mid = l + (r - l) / 2;**

**findmaxmin(a, l, mid, max1, min1);**

**findmaxmin(a, mid + 1, r, max2, min2);**

**maxi = max(max1, max2);**

**mini = min(min1, min2);**

**return;**

**}**

**}**

**// 随机数据生成并写入文件**

**void generateRandomData(int n, const string& filename, int minRange, int maxRange) {**

**ofstream fout(filename);**

**srand(time(0)); // 初始化随机数种子**

**for (int i = 0; i < n; i++) {**

**fout << minRange + rand() % (maxRange - minRange + 1) << " "; // 生成指定范围的随机数**

**}**

**fout.close();**

**}**

**// 从文件中读取数据**

**void readDataFromFile(vector<int>& a, const string& filename, int n) {**

**ifstream fin(filename);**

**for (int i = 0; i < n; i++) {**

**fin >> a[i];**

**}**

**fin.close();**

**}**

**int main() {**

**vector<int> sizes = {10, 1000, 1000000}; // 三种不同规模的数据：小、中、大**

**int minRange = 0; // 随机数的最小值**

**int maxRange = 1000000; // 随机数的最大值**

**for (int size : sizes) {**

**string filename = "findmaxmin" + to\_string(size) + ".txt";**

**// 1. 生成随机数据并写入文件**

**generateRandomData(size, filename, minRange, maxRange);**

**// 2. 从文件中读取数据**

**readDataFromFile(a, filename, size);**

**// 3. 输出数据规模信息**

**cout << "Data size: " << size << " numbers." << endl;**

**// 4. 测量最大值和最小值查找的执行时间**

**int maxi = a[0], mini = a[0];**

**auto start = high\_resolution\_clock::now();**

**findmaxmin(a, 0, size - 1, maxi, mini);**

**auto end = high\_resolution\_clock::now();**

**auto duration = duration\_cast<milliseconds>(end - start);**

**// 5. 输出结果**

**cout << "Max: " << maxi << ", Min: " << mini << endl;**

**cout << "Time taken to find max and min: " << duration.count() << " milliseconds." << endl;**

**cout << "-------------------------------------------" << endl;**

**}**

**return 0;**

**}**