《数据结构》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 李爽 | 班级 | 22级软件工程专升本1班 | 学号 | 2206831522 |
| 实验名称 | **实验六 图** | | | | |
| 实验时间 | 2023年5月12日 | | 成绩 | |  |
| 1. **实验目的：**   1.掌握顺序查找、折半查找算法的基本思想；  2.掌握顺序查找、折半查找算法的实现方法；  3.掌握顺序查找、折半查找算法的时间性能。  **二、实验工具：**  软件平台：Windows7或以上版本，Visual Studio 2019  **三、实验原理：**  1、算法描述：  （1）顺序查找函数：从给定的集合中依次遍历每个元素，直到找到要查找的目标元素或遍历完所有元素。具体实现上，该算法使用一个循环来遍历整个集合，每次循环将当前遍历的元素与目标元素进行比较，若相等则返回当前元素的位置，若遍历完所有元素都没有找到目标元素，则返回-1表示查找失败。  （2）折半查找函数：基于有序数组的特性，采用二分法查找目标值 k。该函数接收三个参数：有序整数数组 numbers、数组长度 n 和要查找的目标值 k。在查找过程中，将数组的中间位置与目标值进行比较，如果中间位置的元素值等于目标值，则直接返回中间位置的下标；如果中间位置的元素值小于目标值，则在数组的右半部分继续查找；如果中间位置的元素值大于目标值，则在数组的左半部分继续查找。通过不断将查找范围缩小一半，最终在数组中找到目标值或者确定目标值不存在。函数中使用 while 循环实现折半查找，循环条件是左边界 left 小于等于右边界 right，每次将查找范围缩小一半。  **四、实验步骤和内容：**   1. 随机产生n（n>500）个整数，完成以下操作:   （1）对给定的查找集合，顺序查找与给定值k相等的元素,并返回查找所用时间；  （2）对给定的有序查找集合，折半查找与给定值k相等的元素,并返回查找所用时间；  部分参考代码如下：  #include<ctime> //使用函数库ctime  ……  int main()  {  clock\_t start,finish; //定义查找开始时间和结束时间变量  ……  start=clock(); //记录查找算法的开始时间  查找算法  finish=clock(); //记录查找算法的结束时间  ……  }  代码如下  #include <iostream>  //rand和srand函数  #include <cstdlib>  //time函数和clock函数  #include <ctime>  //setprecision函数  #include <iomanip>  //sort函数  #include <algorithm>  using namespace std;  //顺序查找算法  int sequentialSearch(int numbers[], int n, int k) {  for (int i = 0; i < n; i++) {  if (numbers[i] == k) {  return i;  }  }  return -1;  }  //折半查找算法  int binarySearch(int numbers[], int n, int k) {  int left = 0, right = n - 1;  while (left <= right) {  int mid = (left + right) / 2;  if (numbers[mid] == k) {  return mid;  } else if (numbers[mid] < k) {  left = mid + 1;  } else {  right = mid - 1;  }  }  return -1;  }  int main() {  const int n = 10000;  int numbers[n];  //初始化随机数种子  srand(time(NULL));  for (int i = 0; i < n; i++) {  //生成0~99999之间的随机数  numbers[i] = rand() % 100000;  }  //生成0~99999之间的随机数作为要查找的值  int k = rand() % 100000;  //顺序查找  //记录查找开始时间  clock\_t start = clock();  //定义顺序查找用时变量  double sequentialTime = 0;  //重复执行1000次  for (int i = 0; i < 1000; i++) {  sequentialSearch(numbers, n, k);  }  //计算顺序查找用时，单位为秒  sequentialTime = (clock() - start) \* 1000.0 / CLOCKS\_PER\_SEC / 1000.0;  //折半查找  //将数组排序  sort(numbers, numbers + n);  //记录查找开始时间  start = clock();  //定义折半查找用时变量  double binaryTime = 0;  //重复执行1000次  for (int i = 0; i < 1000; i++) {  binarySearch(numbers, n, k);  }  //计算折半查找用时，单位为秒  binaryTime = (clock() - start) \* 1000.0 / CLOCKS\_PER\_SEC / 1000.0;  //输出结果  cout << setprecision(3) << fixed;//设置输出精度  cout << "顺序查找用时：" << sequentialTime << " s" << endl;  cout << "折半查找用时：" << binaryTime << " s" << endl;  return 0;  }    **五、实验总结：**  本次实验主要是使用C++语言编写代码，完成随机产生n个整数，并使用顺序查找和折半查找算法查找给定值k相等的元素，最终输出两种算法的查找时间。  实验中，我们使用了srand()函数生成随机数种子，用rand()函数产生0~99999之间的随机数填充长度为n的数组。然后，我们生成一个随机数k作为要查找的值。  对于顺序查找，我们使用了for循环遍历整个数组，逐一比较数组中每个元素和给定值k是否相等。如果找到了相等的元素，就返回该元素的索引值；如果没有找到，就返回-1。在实验中，我们通过计时函数clock()来记录顺序查找所用的时间。  对于折半查找，我们先要将数组排序，这里使用了sort()函数来完成排序操作。然后，我们用while循环进行折半查找。在每一次循环中，我们计算数组中间元素的索引值mid，然后判断该元素与给定值k的大小关系。如果mid处的元素等于k，就返回mid；如果mid处的元素小于k，就继续在右半部分进行查找；如果mid处的元素大于k，就继续在左半部分进行查找。在实验中，我们同样使用clock()函数来记录折半查找所用的时间。  最终，我们将两种算法的查找时间输出，得到了实验的结果。通过实验，我们发现在同样的数据集合下，折半查找算法的时间复杂度更小，效率更高。  **六、教师评语：** | | | | | |