数据结构 A
作业 13 参考答案

作业情况

教材：数据结构教程（C++ 语言描述）李春葆等
题目范围：排序
邮箱：wjyyy1@126.com
授课教师：彭蓉 教授
助教：王骏峣

习题 1 单选题

对序列（15，9，7，8，20，-1，4）进行排序，进行一趟后数据的排列变为（4，9，-1，8，20，7，15），则采用的是（　　）排序

A. 简单选择排序
B. 冒泡排序
C. 希尔排序
D. 快速排序

答案：C

解析：A 项，选择排序的过程需要把最小的元素找到并放在前面，所以排除；

B 项，冒泡排序过程也较容易模拟，显然不符合这一过程，可以排除；

D 项，快速排序需要找一个基准数字，但可以发现这一趟排序没有符合要求的数字。

这一趟排序中，第 1,4,7；2,5；3,6 这三组元素都变得有序，可以是增量为 3 的希尔排序。

习题 2 单选题

数据序列（8，9，10，4，5，6，20，1，2）只能是（　　）算法的两趟排序后的结果

A. 简单选择排序
B. 冒泡排序
C. 直接插入排序
D. 堆排序

答案：C

解析：排除法，比较各算法细节。

A 项：如果是选择排序，那么每次会找最小值放在前面，两趟过后前面两个位置一定分别是最小值和次小值。可以排除。

B 项：冒泡排序是指每趟交换相邻的两个元素，而且最大值一定会在第一趟被冒到最右侧，此时 20 仍然不在最右的位置，所以排除。

C 项：如果是直接插入排序，那么第一趟排序后，前两个元素一定是有序的，符合题意。

D 项：堆排序的根节点一定是最小值，可以排除。

习题 3 单选题

在下列排序方法中，若待排序的数据已经有序，花费时间反而最多的是（　　）

A. 快速排序

B. 希尔排序

C. 冒泡排序

D. 堆排序

答案：A

解析：快速排序在待排序的数据已经有序时，其性能可能会变得最差，考虑到其对数据基准点（pivot）的选择方式。

快速排序的时间复杂度和其基准点的选择密切相关。在常见的快速排序实现中，如果每次都选择最左边或最右边的元素作为基准点，当这个数组已经有序时，就会导致分割不平衡，无法砍半以达到优化复杂度的目的。每次划分只能减少一个元素，这会导致算法退化为 $O(n^2)$ 的时间复杂度。

冒泡排序的最坏复杂度永远都是 $O(n^2)$，不受影响。

希尔排序建立在插入排序基础上，如果有序时算法会更快。

堆排序的时间复杂度无论如何不会超过 $O(n \log n)$，不受影响。

习题 4 单选题

在下列排序方法中，执行时间不受数据初始状态影响，总为 $O(n \log_2 n)$ 的是（　　）

A. 堆排序

B. 冒泡排序

C. 简单选择排序

D. 快速排序

答案：A

解析：对于堆排序，无论如何都要检查每一个位置是否满足堆的条件，所以其时间复杂度是 $O(n\log n)$。

对于冒泡排序，当数据有序时只需要 $O(n)$；当数据无序时仍然需要 $O(n^2)$，两个都不满足题意。

简单选择排序的时间复杂度无论如何都是 $O(n^2)$，不满足题意。

快速排序的时间复杂度是 $O(n\log n)$，但是在数据有序时可能会退化为 $O(n^2)$（参考上一题），所以也不满足题意。

习题 5 单选题

在下列排序方法中，某一趟结束后未必能选出一个元素放在其最终位置上的是（　　）

A. 堆排序

B. 冒泡排序

C. 直接插入排序

D. 快速排序

答案：C

解析：A 项，堆排序的根节点一定是最小/最大值，所以有确定元素。

B 项，最大的数字一定在第一趟就到了最后一个位置，所以有确定元素。

C 项，插入排序在第 i 趟只会考虑前 $i+1$ 个数据，而后面的元素还可能对已经排好的那些元素造成比较大的影响，比如出现了最小的元素，就会把已排序的所有值都往后移动，因此错误。

D 项，一趟快速排序之后，基准点的位置就确定了，因为左边都是比它小的，右边都比它大。

习题 6 快速排序

【问题描述】

有一个含 n（$n \le 200000$）个整数的无序序列，采用快速排序实现递增排序

【输入形式】

一行字符串，包含多个整数，每个数之间用空格分开。

【输出形式】

递增排序的结果，每个数之间用空格分开。

【样例输入】

```
9 4 7 6 2 5 8 1 3
```

【样例输出】

```
1 2 3 4 5 6 7 8 9
```

【样例说明】
测试数据的文件名为 in.txt，输出文件名为 out.txt
【评分标准】
该题目有 10 个测试用例，每通过一个测试用例得 10 分

答案：参考快速排序的代码如下：

```cpp
#include<bits/stdc++.h>
#include <string>
#include <stdio.h>
#include <vector>

using namespace std;
int partion(vector<int>& nums, int start, int end)
{   //分治合并过程
    int target = nums[start];
    int i = start, j = end;
    while (i < j)
    {
        while (i<j && nums[j]>target) j--;
        while (i < j && nums[i] <= target) i++;
        if (i < j)
            swap(nums[i], nums[j]);
    }
    swap(nums[i], nums[start]);
    return i;
}

void quickSort(vector<int>& nums, int start, int end)
{   //快速排序递归过程
    if (start < end)
    {
        int i = partion(nums, start, end);
        quickSort(nums, start, i - 1);
        quickSort(nums, i + 1, end);
    }
}

int main()
{
    freopen("in.txt", "r", stdin);
    vector<int> nums;
    int num;
    while (cin >> num)
    {
```

```
        nums.push_back(num);
    }
    freopen("out.txt", "w", stdout);
    quickSort(nums, 0,nums.size()-1);
    int i = 0;
    for (; i < nums.size() - 1; i++)
    {
        cout << nums[i] << ' ';
    }
    cout << nums[i] << '\n';;
    return 0;

}
```

解析：请注意掌握快速排序的算法和思路，并对其实现有比较扎实的理解。快速排序的基本思路是选取一个基准点，然后将比基准点小的数放在左边，比基准点大的数放在右边，然后递归地对左右两边进行排序。

但为了避免时间过长，每次选择基准点可以不按上述代码写，而是可以随机在**区间内**选择一个点，以免退化为 $O(n^2)$ 的时间复杂度。

习题 7 希尔排序

【问题描述】

有一个含 n（$n \leq 200000$）个整数的无序序列，采用希尔排序实现递增排序

【输入形式】

一行字符串，包含多个整数，每个数之间用空格分开。

【输出形式】

递增排序的结果，每个数之间用空格分开。

【样例输入】

```
9 4 7 6 2 5 8 1 3
```

【样例输出】

```
1 2 3 4 5 6 7 8 9
```

【样例说明】

测试数据的文件名为 `in.txt`，输出文件名为 `out.txt`

【评分标准】

该题目有 10 个测试用例，每通过一个测试用例得 10 分

答案：参考快速排序的代码如下：

```
#include <iostream>
#include <string>
#include <fstream>
#include <stdio.h>
```

```cpp
#include <vector>

using namespace std;
void shellSort(vector<int>& R, int n)
{
    int d = n / 2;
    while (d > 0)//选择增量
    {
        for (int i = d; i < n; i++)
        {
            if (R[i] < R[i - d])
            {
                int tmp = R[i];
                int j = i - d;
                do
                {
                    R[j + d] = R[j];
                    j = j - d;
                } while (j >= 0 && R[j] > tmp);
                R[j + d] = tmp;
            }
        }
        d = d / 2;
    }
}

int main()
{
    freopen("in.txt", "r", stdin);
    vector<int> nums;
    int num;
    while (cin >> num)
    {
        nums.push_back(num);
    }
    freopen("out.txt", "w", stdout);
    shellSort(nums, nums.size());
    int i = 0;
    for (; i < nums.size() - 1; i++)
    {
        cout << nums[i] << ' ';
    }
    cout << nums[i] << endl;;
    return 0;

}
```

解析：希尔排序的增量变化有很多实现方式，掌握其中一个即可。主要是要注意插入排序的过程不要写错，然后难点在于下标和增量的变化。

习题 8 快速排序

【问题描述】

有一个含 n（$n \leq 200000$）个整数的无序序列，设计一个算法利用快速排序思路求前 10 个最大的元素。

【输入形式】

一行字符串，包含多个整数，每个数之间用空格分开。

【输出形式】

前 10 个最大的元素，按递减排序，用空格分开。

【样例输入】

```
1 5 32 4 6 8 9 4 7 6 55 1 3 65 24
```

【样例输出】

```
65 55 32 24 9 8 7 6 6 5
```

【样例说明】

测试数据的文件名为 `in.txt`，输出文件名为 **out.txt**

【评分标准】

该题目有 10 个测试用例，每通过一个测试得 10 分。

答案：实现细节同习题 6，此处仅给出输入输出部分。

```cpp
int main()
{
    freopen("in.txt", "r", stdin);
    vector<int> nums;
    int num;
    while (cin >> num)
    {
        nums.push_back(num);
    }
    freopen("out.txt", "w", stdout);
    quickSort(nums, 0, nums.size() - 1);
    int i = nums.size() - 1;
    for (; i > nums.size() - 10; i--)
    {
        cout << nums[i] << ' ';
    }
    cout << nums[i] << endl;;
    return 0;

}
```

解析：算法部分为快速排序，在输入输出时和其他题有一点点区别，可以注意一下。最后只按从大到小顺序输出 10 个数字即可。

习题 9 前 m 大的数

【问题描述】
给你 n 个整数，请按从大到小的顺序输出其中前 m 大的数。
【输入形式】
每组测试数据有两行，第一行有两个数 n 和 m（$0 < n$，$m < 1000000$），第二行包含 n 个各不相同，且都处于区间 $[-500000, 500000]$ 的整数。
【输出形式】
对每组测试数据按从大到小的顺序输出前 m 大的数。
【样例输入】

```
5 3
3 -35 92 213 -644
```

【样例输出】

```
213 92 3
```

【样例说明】
对于输入的 5 个整数，前 3 大的整数分别是 213、92、3。测试数据存放在 `in.txt` 文件中。
【评分标准】
该题目有 10 个测试用例，每通过一个测试得 10 分。

答案：如果想学习 `sort` 的使用，可以参考下面的代码：

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <functional>
#include <algorithm>

using namespace std;

int main()
{
    ifstream inFile;
    inFile.open("in.txt", ios::in);

    if(!inFile){
        cout << "error open in.txt!" << endl;
        return 1;
    }
```

```
18
19      int n,m,x;
20      vector<int> myv;
21      if(!inFile.eof())
22      {
23          inFile >> n >> m;
24          myv.clear();
25          for(int i=0;i<n;i++)
26          {
27              inFile >> x;
28              myv.push_back(x);
29          }
30          sort(myv.begin(),myv.end(),greater<int>());
31          for(int i=0;i<m;i++)
32          { if (i>0) printf(" ");
33              printf("%d",myv[i]);
34          }
35          printf("\n");
36      }
37      inFile.close();
38      return 0;
39  }
```

解析：本题思路同前面第八题。只需要输出前 m 个数即可。可以使用库中的 `sort` 函数，也可以自行实现快速排序。

总结

题目：排序
日期：2024 年 6 月 22 日
批改人：王骏峣
邮箱：wjyyy1@126.com
习题 1：对各种排序过程的掌握。
习题 2：考察排序过程和一部分逆向排序思维，主要是要考虑是否有没想到的情况。
习题 3：注意对快速排序**缺点**的掌握，快速排序运行速度很快，但缺点就是容易被有序数据这种常见问题卡掉复杂度。
习题 4, 5：对概念的理解。
习题 6, 7：对不同排序算法思路的检查，以及对排序算法的实现。
习题 8, 9：对不同算法实际应用的考核。
各位同学如有问题欢迎及时在群里提出，或者通过邮件/QQ 联系我。