

浙江工业大学

算法分析与设计实验报告

(2021 级)



实验题目

实验 1

学生姓名

温家伟

学生学号

202103151422

专业班级

大数据分析 2101

所在学院

理学院

提交日期

2023-3-8

实验目的

1. 掌握递归法与分治的解题步骤
2. 掌握递归算法、二分搜索算法、全排列算法、插入和归并排序算法

第一章 递归与分治策略

实验内容

1.斐波那契数列 (Fibonacci Sequence) , 又称黄金分割数列, 指的是这样一个数列: 1、1、2、3、5、8、13、21、.....。在数学上, 斐波纳契数列以递推的方法定义为: $F(1)=1$, $F(2)=1$, $F(n)=F(n-1)+F(n-2)$ ($n \geq 2$, $n \in \mathbb{N}$)。计算斐波那契数列第 n 项的值。

输入格式:

输入一个大于等于1, 小于等于60的整数 n 。

输出格式:

输出第 n 项的数列值, 数列值为double类型, 不输出小数位数。

2.请编写程序输出前 n 个正整数的全排列 ($n < 10$) , 并通过9个测试用例 (即 n 从1到9) 观察 n 逐步增大时程序的运行时间。

输入格式:

输入给出正整数 n (< 10) 。

输出格式:

输出1到 n 的全排列。每种排列占一行, 数字间无空格。排列的输出顺序为字典序, 即序列 a_1, a_2, \dots, a_n 排在序列 b_1, b_2, \dots, b_n 之前, 如果存在 k 使得 $a_1=b_1, \dots, a_k=b_k$ 并且 $a_{k+1} < b_{k+1}$ 。

3.设 $a[0:n-1]$ 是已排好序的数组, 请改写二分搜索算法, 使得当 x 不在数组中时, 返回小于 x 的最大元素位置 i 和大于 x 的最小元素位置 j 。当搜索元素在数组中时, i 和 j 相同, 均为 x 在数组中的位置。

输入格式:

输入有两行:

第一行是 n 值和 x 值; 第二行是 n 个不相同的数组成的非降序序列, 每个整数之间以空格分隔。

输出格式:

输出小于 x 的最大元素的最大下标 i 和大于 x 的最小元素的最小下标 j 。当搜索元素在数组中时, i 和 j 相同。提示: 若 x 小于全部数值, 则输出: -1 0 若 x 大于全部数值, 则输出: $n-1$ 的值 n 的值

4.插入排序是迭代算法, 逐一获得输入数据, 逐步产生有序的输出序列。每步迭代中, 算法从输入序列中取出一元素, 将之插入有序序列中正确的位置。如此迭代直到全部元素有序。

归并排序进行如下迭代操作: 首先将原始序列看成 N 个只包含 1 个元素的有序子序列, 然后每次迭代归并两个相邻的有序子序列, 直到最后只剩下 1 个有序的序列。

现给定原始序列和由某排序算法产生的中间序列, 请你判断该算法究竟是哪种排序算法?

输入格式:

输入在第一行给出正整数 N (≤ 100); 随后一行给出原始序列的 N 个整数; 最后一行给出由某排序算法产生的中间序列。这里假设排序的目标序列是升序。数字间以空格分隔。

输出格式:

首先在第 1 行中输出Insertion Sort表示插入排序、或Merge Sort表示归并排序; 然后在第 2 行中输出用该排序算法再迭代一轮的结果序列。题目保证每组测试的结果是唯一的。数字间以空格分隔, 且行首尾不得有多余空格。

实验结果及相应代码

1.函数-斐波那契数列

1.1 PTA提交代码截图

```

C++ (g++)
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      cin >> n;
8      long long int* dp = new long long int[n];
9      dp[0] = 1;
10     dp[1] = 1;
11     for (int i = 2; i < n; ++i)
12     {
13         dp[i] = dp[i - 1] + dp[i - 2];
14     }
15     cout << dp[n - 1] << endl;
16     return 0;
17 }

```

1.2 PTA提交代码结果截图(下图为例子)

提交结果

提交时间	状态	分数	题目	编译器	内存	用时	用户
2023/03/08 14:54:25	答案正确	20	7-1	C++ (g++)	600 KB	6 ms	202103151422

测试点	结果	分数	耗时	内存
0	答案正确	5	6 ms	308 KB
1	答案正确	5	5 ms	484 KB
2	答案正确	5	5 ms	600 KB
3	答案正确	5	4 ms	320 KB

代码

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      cin >> n;
8      long long int* dp = new long long int[n];
9      dp[0] = 1;

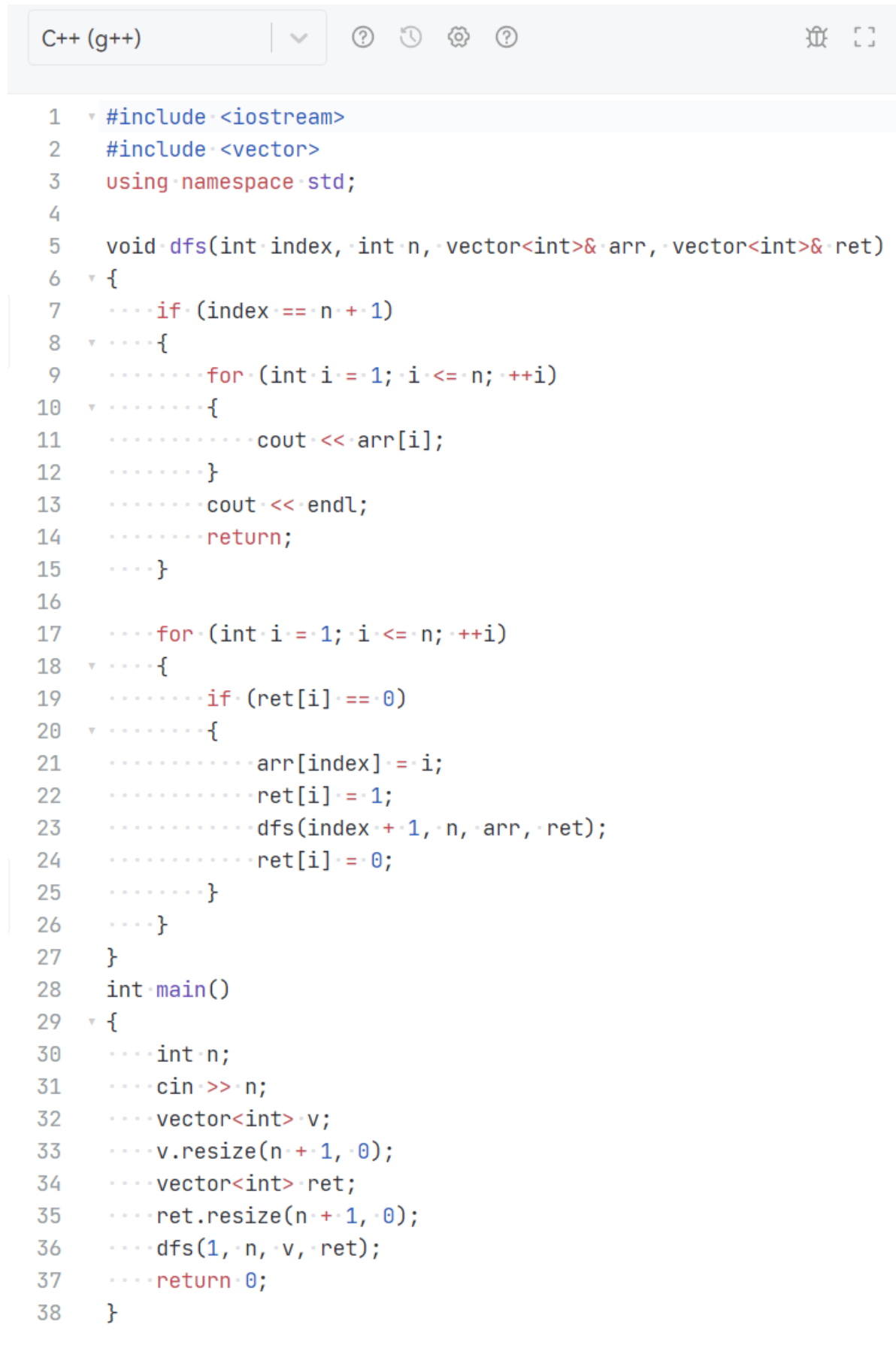
```

1.3 算法分析

动态规划，转移方程题目已经给出，要注意int可能会超出范围，改long long int才能过。

2.输出全排列

2.1 PTA提交代码截图

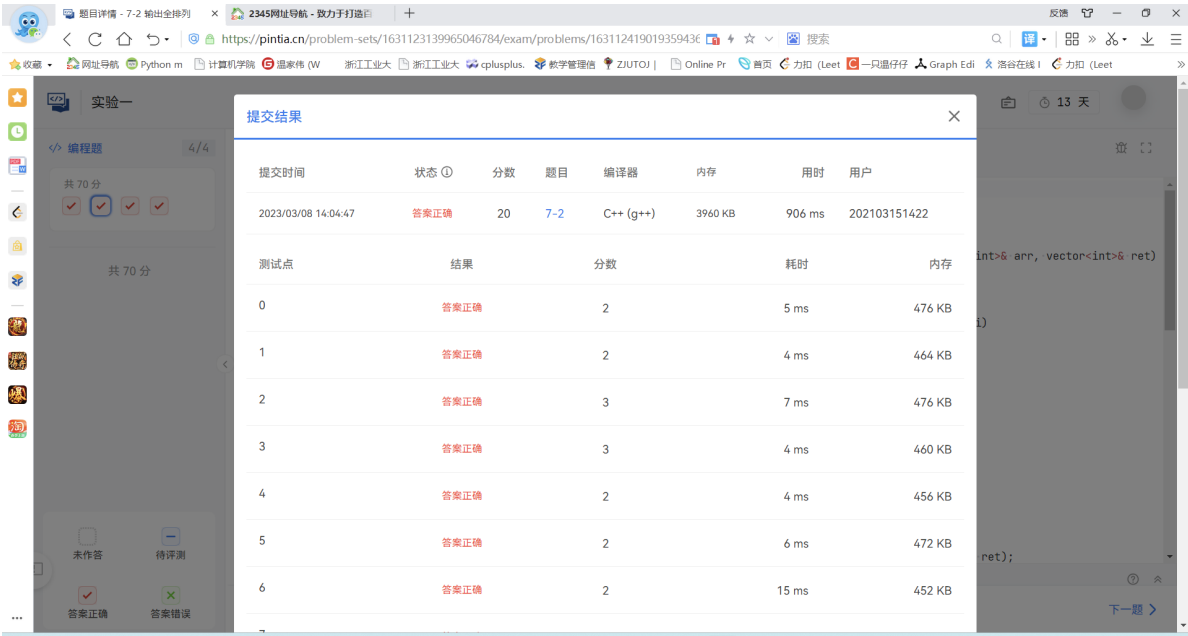


```

C++ (g++)
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void dfs(int index, int n, vector<int>& arr, vector<int>& ret)
6  {
7      if (index == n + 1)
8      {
9          for (int i = 1; i <= n; ++i)
10         {
11             cout << arr[i];
12         }
13         cout << endl;
14         return;
15     }
16
17     for (int i = 1; i <= n; ++i)
18     {
19         if (ret[i] == 0)
20         {
21             arr[index] = i;
22             ret[i] = 1;
23             dfs(index + 1, n, arr, ret);
24             ret[i] = 0;
25         }
26     }
27 }
28 int main()
29 {
30     int n;
31     cin >> n;
32     vector<int> v;
33     v.resize(n + 1, 0);
34     vector<int> ret;
35     ret.resize(n + 1, 0);
36     dfs(1, n, v, ret);
37     return 0;
38 }

```

2.2 PTA提交代码结果截图



2.3 算法分析

DFS入门题，递归每一种排列的可能。

3.改写二分搜索算法

3.1 PTA提交代码截图

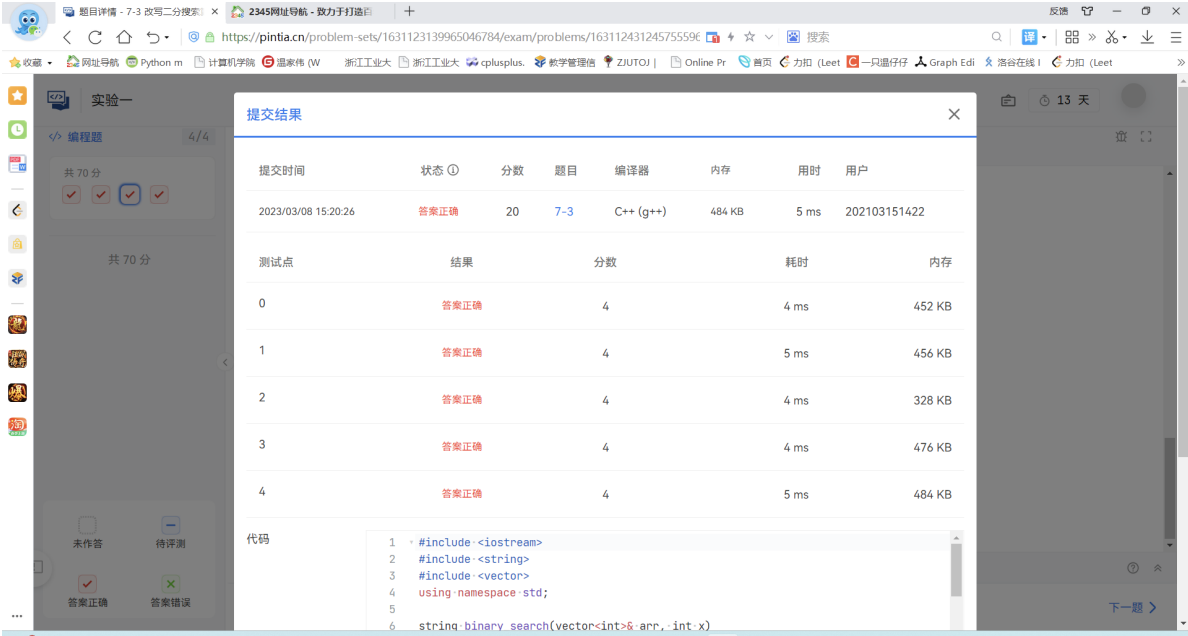
```

1  ▾ #include <iostream>
2    #include <string>
3    #include <vector>
4    using namespace std;
5
6    string binary_search(vector<int>& arr, int x)
7  ▾ {
8      → if (x < arr.front())
9  ▾ → {
10     → return "-1 0";
11     → }
12     → if (x > arr.back())
13  ▾ → {
14     → string ret;
15     → ret += to_string(arr.size() - 1);
16     → ret += ' ';
17     → ret += to_string(arr.size());
18     → return ret;
19     → }
20     → int left = 0;
21     → int right = arr.size() - 1;
22     → while (left <= right)
23  ▾ → {
24     → int mid = (left + right) / 2;
25     → if (arr[mid] == x)
26  ▾ → {
27     → string ret;
28     → ret += to_string(mid);
29     → ret += ' ';
30     → ret += to_string(mid);
31     → return ret;
32     → }
33     → else if (arr[mid] > x)
34  ▾ → {
35     → right = mid - 1;
36     → }
37     → else
38  ▾ → {
39     → left = mid + 1;
40     → }
41     → }
42     → string temp;
43     → temp += to_string(right);
44     → temp += ' ';
45     → temp += to_string(left);
46

```

```
47     return temp;
48 }
49 int main()
50 {
51     int n, x;
52     cin >> n >> x;
53     vector<int> arr;
54     arr.resize(n);
55     for (int i = 0; i < n; ++i)
56     {
57         cin >> arr[i];
58     }
59     cout << binary_search(arr, x) << endl;
60     return 0;
61 }
```

3.2 PTA提交代码结果截图



3.3 算法分析

二分查找，大一学的，控制好边界即可。

4.插入排序还是归并排序

4.1 PTA提交代码截图


```

1  ▽ #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void merge(vector<int>& data, int start, int mid
6  , int end, vector<int>& result)
7  ▽ {
8  — int i, j, k;
9  — i = start;
10 — j = mid + 1;
11 — k = 0;
12 — while (i <= mid && j <= end)
13 ▽ — {
14 — — if (data[i] <= data[j])
15 ▽ — — {
16 — — — result[k++] = data[i++];
17 — — }
18 — — else
19 ▽ — — {
20 — — — result[k++] = data[j++];
21 — — }
22 — }
23 — while (i <= mid)
24 ▽ — {
25 — — result[k++] = data[i++];
26 — }
27 — while (j <= end)
28 ▽ — {
29 — — result[k++] = data[j++];
30 — }
31 — for (i = 0; i < k; ++i)
32 ▽ — {
33 — — data[start + i] = result[i];
34 — }
35 — for (size_t o = 0; o < data.size(); ++o)
36 ▽ — {
37 — — cout << data[o];
38 — — — if (o != data.size() - 1)
39 ▽ — — — {
40 — — — — cout << ' ';
41 — — — }
42 — }
43 — cout << endl;
44 }
45 void mergesort(vector<int>& data, int start
46 , int end, vector<int>& result)
47 ▽ {

```

```

48     if (start > end)
49     {
50         int mid = (start + end) / 2;
51         mergesort(data, start, mid, result);
52         mergesort(data, mid + 1, end, result);
53         merge(data, start, mid, end, result);
54     }
55 }
56 int main()
57 {
58     int n;
59     while (cin >> n)
60     {
61         vector<int> arr;
62         arr.resize(n);
63         vector<int> result;
64         result.resize(n);
65         for (int i = 0; i < n; ++i)
66         {
67             cin >> arr[i];
68         }
69         mergesort(arr, 0, n - 1, result);
70     }
71     return 0;
72 }

```

4.2 PTA提交代码结果截图

The screenshot shows a web browser window displaying a problem page on the Pintia platform. The problem is titled "7-4 二路归并排序" (Merge Sort) and is worth 10 points. The author is 黄龙军 (Huang Longjun) from 绍兴文理学院 (Shaoxing University of Arts and Sciences).

问题描述: 给定一个整数序列，请按非递减序输出采用二路归并排序（递归法）的各趟排序后的结果（每完成一次归并操作就输出归并后的结果）。

输入格式: 测试数据有多组，处理到文件尾。每组测试数据第一行输入一个整数 n ($1 \leq n \leq 100$)，第二行输入 n 个整数。

输出格式: 对于每组测试，输出若干行，每行是一趟排序后的结果，每行的每两个数据之间留一个空格。

输入样例:

```
6
73 12 27 98 81 64
```

输出样例:

```
12 73 27 98 81 64
12 27 73 98 81 64
42 27 73 81 98 64
```

The right side of the image shows a C++ solution code for the problem:

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 void merge(vector<int>& data, int start, int mid, int end, vect
6 {
7     int i, j, k;
8     i = start;
9     j = mid + 1;
10    k = 0;
11    while (i <= mid && j <= end)
12    {
13        if (data[i] <= data[j])
14        {
15            result[k++] = data[i++];
16        }
17        else
18        {
19            result[k++] = data[j++];
20        }
21    }
22    while (i <= mid)
23    {
```

4.3 算法分析

分治二路递归，单趟排序后输出结果，注意每行最后一个元素后不能有空格（这个好难发现）。