浙江工业大学

算法分析与设计实验报告

(2021 级)

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 2
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	2023-3-20

实验目的

1. 掌握递归法与分治的解题步骤
2. 掌握快速排序算法、大整数乘法算法、二叉树算法、因式分解算法

第二章 递归与分治策略

实验内容

1.快速排序的每一趟输出

给定一个整数序列，请按非递减序输出采用快速排序（递归法）的各趟排序后的结果。注意：每趟排序以排序区间的首元素为基准，让L和R分别从数组头部和尾部出发，向中间移动，L找比key大的值，R找比key小的值，然后进行交换。

例如：6 1 2 7 9 3 4 5 10 8

第一趟的交换元素步骤如下：

①[6] 1 2 **7** 9 3 4 **5** 10 8 -> [6] 1 2 **5** 9 3 4 **7** 10 8

②[6] 1 2 5 **9** 3 **4** 7 10 8 -> [6] 1 2 5 **4** 3 **9** 7 10 8

③**6** 1 2 5 4 **3** 9 7 10 8 -> **3** 1 2 5 4 **6** 9 7 10 8 即 {3 1 2 5 4} 6 {9 7 10 8}

每结束一次以头元素作为基准的交换时，作为一趟的排序的结果并输出3 1 2 5 4 6 9 7 10 8

输入格式:

测试数据有一组，测试数据第一行输入一个整数n（2≤n≤30），第二行输入n个整数。

输出格式:

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1:

```
4
8 7 2 1
```

输出样例1:

```
1 7 2 8
1 7 2 8
1 2 7 8
```

输入样例2:

```
10
6 1 2 5 4 3 9 7 10 8
```

输出样例2:

```
3 1 2 5 4 6 9 7 10 8
2 1 3 5 4 6 9 7 10 8
1 2 3 5 4 6 9 7 10 8
1 2 3 4 5 6 9 7 10 8
1 2 3 4 5 6 8 7 9 10
1 2 3 4 5 6 7 8 9 10
```

2.正整数n不同分解式的个数

对于大于1的正整数n,可以分解为n=x1* x2 …… xm，其中xi>=2。例如n=12时有8种不同的分解，即12=12,12=6 * 2,12=4 * 3,12=3*4,12=3 * 2 * 2,12=2 * 6,12=2 * 3 * 2,12=2 * 2 * 3;设计一个算法求n的不同分解式的个数。（来源于《算法设计与分析(第2版）李春葆》）

输入格式:

输入一个正整数n

输出格式:

输出1个正整数,表示n的不同分解式的个数

输入样例:

```
12
```

输出样例:

```
8
```

3.二叉搜索树的后序遍历序列

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历结果。如果是则输出 `true`，否则输出 `false`。假设输入的数组的任意两个数字都互不相同。

例如：[1,6,3,2,5]不是某二叉搜索树的后序遍历结果，则输出`false`。[1,3,2,6,5]是某二叉搜索树的后序遍历结果，则输出`true`。

参考此二叉搜索树：

```
    5
   / \
  2   6
 / \
1   3
```

输入格式:

输入第一行给出正整数N，随后一行给出N个整数，其间以空格分隔。

输出格式:

若该数组是某二叉搜索树的后序遍历结果则输出`true`,否则输出`false`

输入样例1:

```
5
1 6 3 2 5
```

输出样例1:

```
false
```

输入样例2:

```
5
1 3 2 6 5
```

输出样例2:

```
true
```

4.PG048 大整数乘法

求两个不超过200位的非负整数的积。

输入格式:

输入有多组数据，每组有两行，每行是一个不超过200位的非负整数，没有多余的前导0。

输出格式:

一行，即相乘后的结果。结果里不能有多余的前导0，即如果结果是342，那么就不能输出为0342。

输入样例:

在这里给出一组输入。例如:

```
1234
5678
```

输出样例:

在这里给出相应的输出。例如:

```
7006652
```

实验结果及相应代码

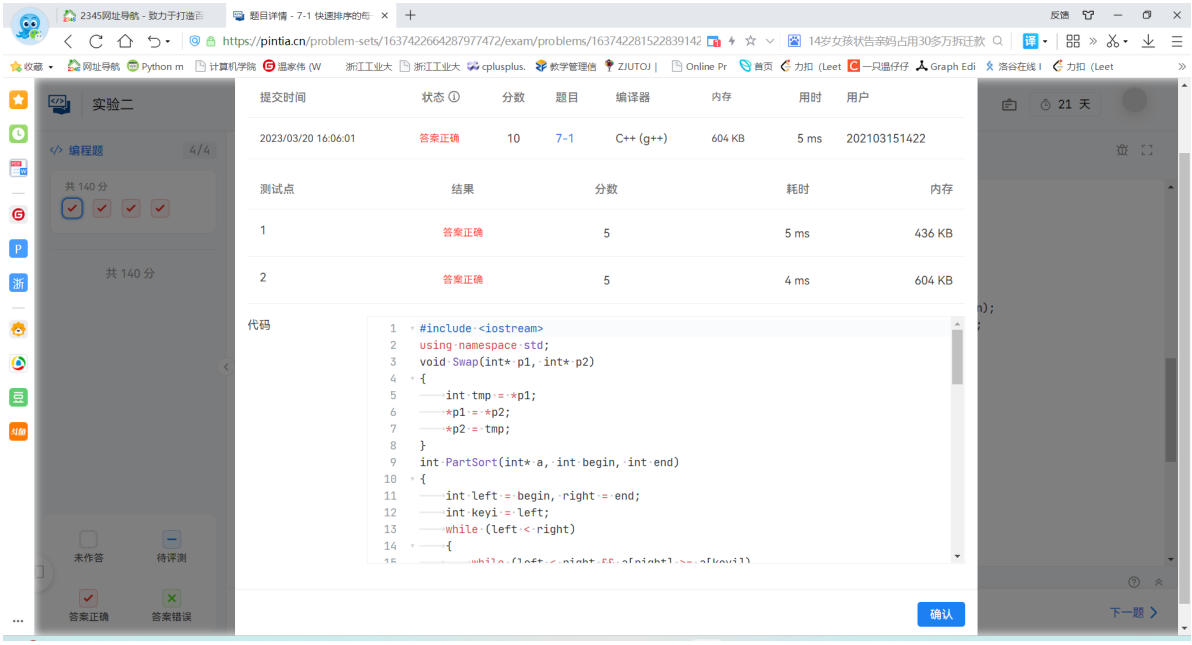
1.快速排序的每一趟输出

1.1 PTA提交代码截图

```cpp
#include <iostream>
using namespace std;
void Swap(int* p1, int* p2)
{
    int tmp = *p1;
    *p1 = *p2;
    *p2 = tmp;
}
int PartSort(int* a, int begin, int end)
{
    int left = begin, right = end;
    int keyi = left;
    while (left < right)
    {
        while (left < right && a[right] >= a[keyi])
        {
            --right;
        }
        while (left < right && a[left] <= a[keyi])
        {
            ++left;
        }
        Swap(&a[left], &a[right]);
    }
    Swap(&a[left], &a[keyi]);
    keyi = left;
    return keyi;
}
void QuickSort(int* a, int begin, int end, int n)
{
    if (begin >= end)
    {
        return;
    }
    int keyi = PartSort(a, begin, end);
    for (int i = 0; i < n; ++i)
    {
        cout << a[i];
        if (i != n - 1)
        {
            cout << ' ';
        }
    }
    cout << endl;
    QuickSort(a, begin, keyi - 1, n);
    QuickSort(a, keyi + 1, end, n);
}
int main()
{
```

```
50      int n;
51      cin >> n;
52      int* arr = new int[n];
53      for (int i = 0; i < n; ++i)
54      {
55          cin >> arr[i];
56      }
57      QuickSort(arr, 0, n - 1, n);
58      return 0;
59  }
```

1.2 PTA提交代码结果截图



1.3 算法分析

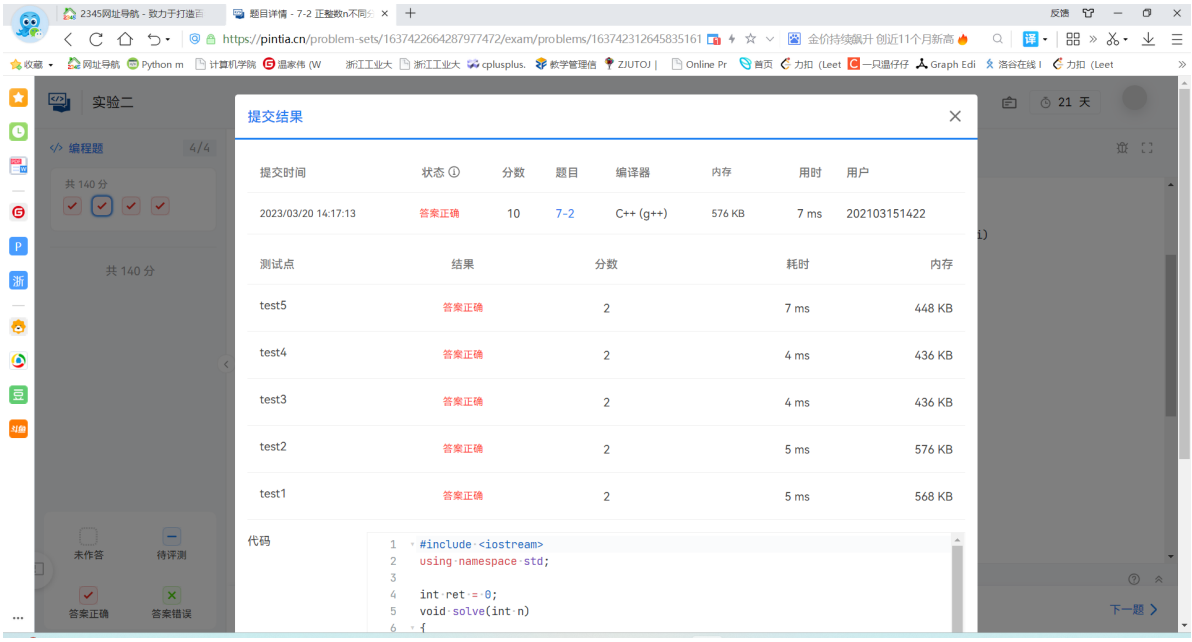
> 快排hoare版本，由基准值把序列分成两部分，然后再递归下去，每趟排好后输出（注意格式）。

2.正整数n不同分解式的个数

2.1 PTA提交代码截图

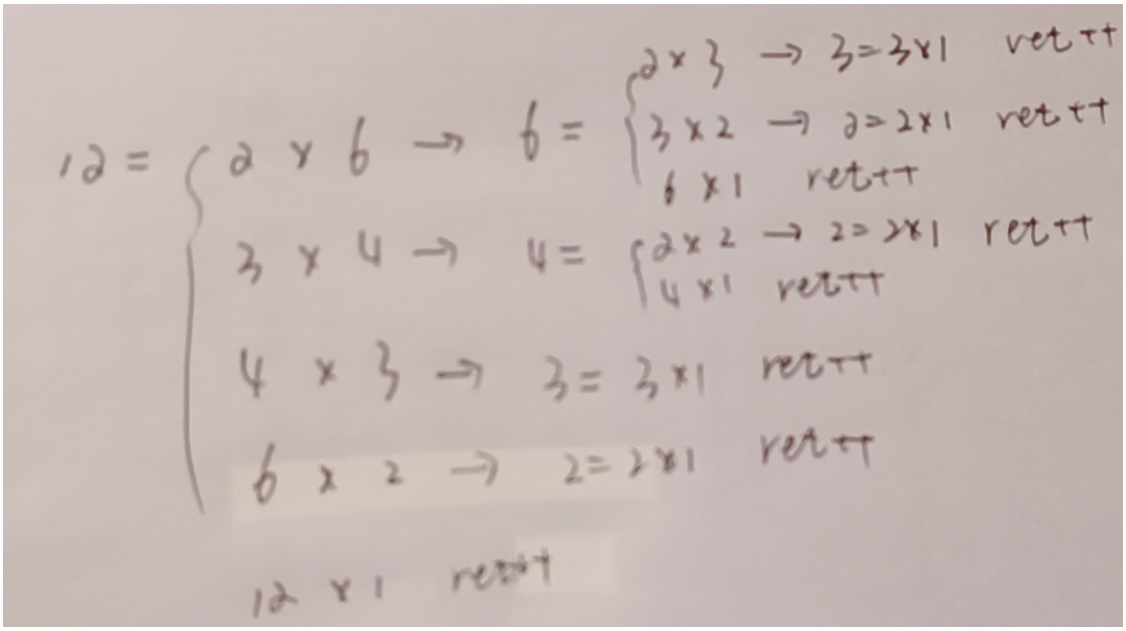
```cpp
#include <iostream>
using namespace std;

int ret = 0;
void solve(int n)
{
    if (n == 1)
    {
        ret++;
    }
    else
    {
        for (int i = 2; i <= n; ++i)
        {
            if (n % i == 0)
            {
                solve(n / i);
            }
        }
    }
}

int main()
{
    int n;
    cin >> n;
    solve(n);
    cout << ret << endl;
    return 0;
}
```

2.2 PTA提交代码结果截图



2.3 算法分析

> 从前往后，每次固定一个数字，即第一次固定2、3、4、6、12，他们的分法取决于后面数字的分法，以此类推，当n == 1时是递归出口，ret++即可。



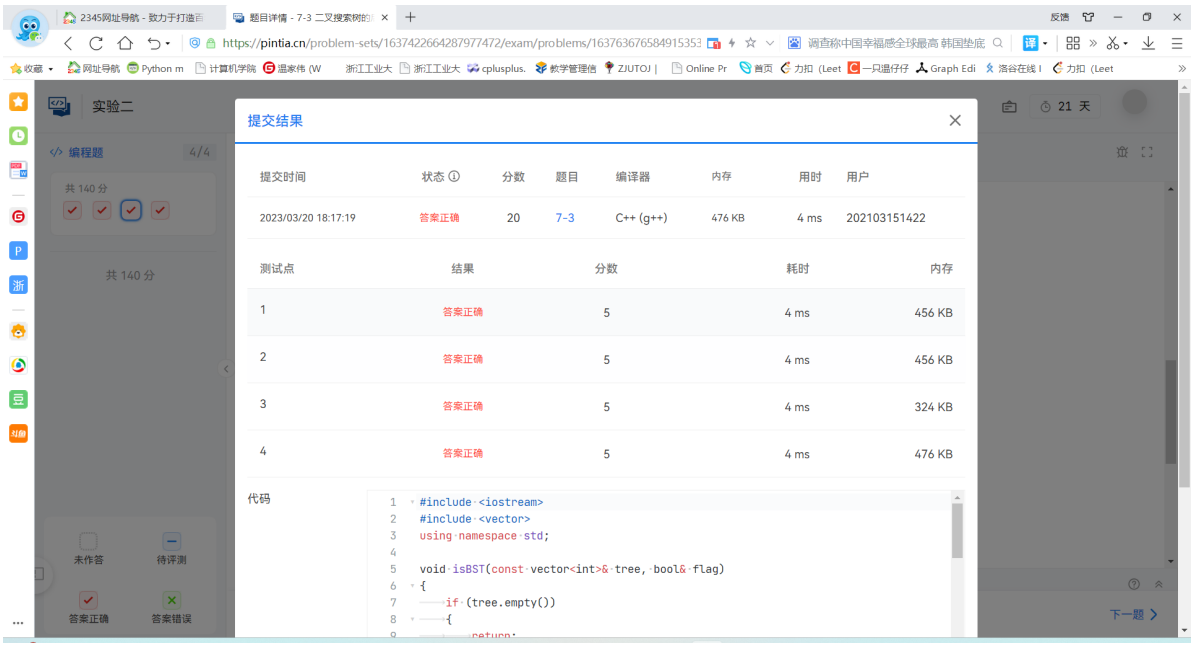
3.二叉搜索树的后序遍历序列

3.1 PTA提交代码截图

```cpp
#include <iostream>
#include <vector>
using namespace std;

void isBST(const vector<int>& tree, bool& flag)
{
    if (tree.empty())
    {
        return;
    }
    int root = tree.back();
    vector<int> left_child_tree;
    vector<int> right_child_tree;
    for (size_t i = 0; i < tree.size() - 1; ++i)
    {
        if (tree[i] < root)
        {
            if (!right_child_tree.empty())
            {
                flag = false;
                return;
            }
            left_child_tree.push_back(tree[i]);
        }
        else
        {
            right_child_tree.push_back(tree[i]);
            if (tree[i + 1] < root)
            {
                flag = false;
                return;
            }
        }
    }
    isBST(left_child_tree, flag);
    isBST(right_child_tree, flag);
}
int main()
{
    int n;
    cin >> n;
    vector<int> v;
    v.resize(n, 0);
    for (auto& e : v)
    {
        cin >> e;
    }
```

```
    isBST(v, flag);
    if (flag)
    {
        cout << "true" << endl;
    }
    else
    {
        cout << "false" << endl;
    }
    return 0;
}
```

3.2 PTA提交代码结果截图



3.3 算法分析

> 若是给定一个后序遍历序列，又知道它是二叉搜索树，那么这棵树是可以画出来的。因为是后续遍历（左-右-根），所以可以知道序列的最右端一定是根，然后根据搜索树的规则序列的前n-1个元素前半段小于根节点的值，后半段大于根节点的值。传入一个flag变量初始化为true，若是违反了上述规则，直接flag = false;

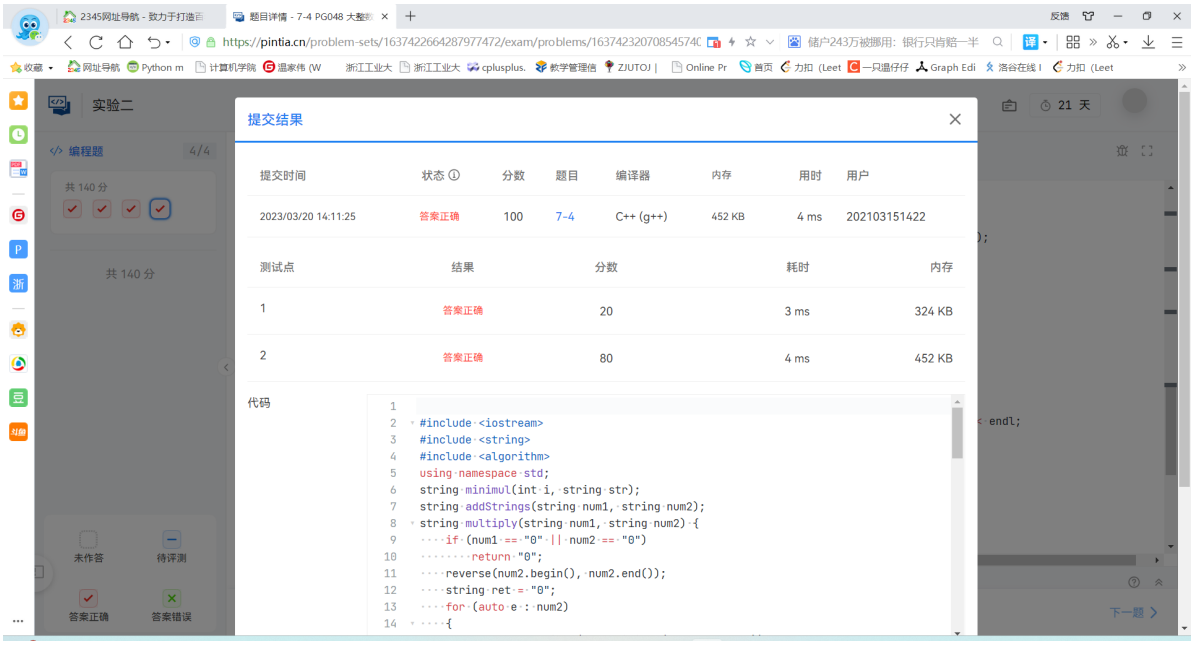
4.PG048 大整数乘法

4.1 PTA提交代码截图

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
string minimul(int i, string str);
string addStrings(string num1, string num2);
string multiply(string num1, string num2) {
    if (num1 == "0" || num2 == "0")
        return "0";
    reverse(num2.begin(), num2.end());
    string ret = "0";
    for (auto e : num2)
    {
        ret = addStrings(ret, minimul(e - '0', num1));
        num1 += '0';
    }
    return ret;
}
string minimul(int i, string str)
{
    if (i == 0)
        return "0";
    reverse(str.begin(), str.end());
    string ret;
    int carry = 0;
    for (auto e : str)
    {
        ret += (e - '0') * i % 10 + '0' + carry;
        carry = (e - '0') * i / 10;
    }
    if (carry)
        ret += (carry + '0');
    reverse(ret.begin(), ret.end());
    return ret;
}
string addStrings(string num1, string num2)
{
    int jinwei = 0;
    int a, b;
    string ret;
    for (int i = 0; i < max(num1.size(), num2.size()); i++)
    {
        a = i < num1.size() ? num1[num1.size() - 1 - i] : '0';
        b = i < num2.size() ? num2[num2.size() - 1 - i] : '0';
        ret.push_back(((a - '0' + b - '0' + jinwei) % 10) + '0'
        jinwei = (a - '0' + b - '0' + jinwei) / 10;
    }
    if (jinwei)
```

```
        ret += '1';
    reverse(ret.begin(), ret.end());
    return ret;
}
int max(int a, int b)
{
    return a > b ? a : b;
}
int main()
{
    string s1, s2;
    while(cin >> s1 >> s2)
        cout << multiply(s1, s2) << endl;
    return 0;
}
```

4.2 PTA提交代码结果截图



4.3 算法分析

> 纯模拟，先实现大整数的加法和迷你乘法（大整数乘一位数），然后模拟竖式乘法，累加得到结果。