

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

③6 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
```

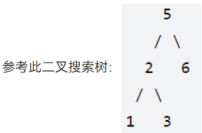
输出样例:

```
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。



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:

```
7086652
```

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

```

1  #include <iostream>
2  using namespace std;
3  void Swap(int* p1, int* p2)
4  {
5      int tmp = *p1;
6      *p1 = *p2;
7      *p2 = tmp;
8  }
9  int PartSort(int* a, int begin, int end)
10 {
11     int left = begin, right = end;
12     int keyi = left;
13     while (left < right)
14     {
15         while (left < right && a[right] >= a[keyi])
16         {
17             --right;
18         }
19         while (left < right && a[left] <= a[keyi])
20         {
21             ++left;
22         }
23         Swap(&a[left], &a[right]);
24     }
25     Swap(&a[left], &a[keyi]);
26     keyi = left;
27     return keyi;
28 }
29 void QuickSort(int* a, int begin, int end, int n)
30 {
31     if (begin >= end)
32     {
33         return;
34     }
35     int keyi = PartSort(a, begin, end);
36     for (int i = 0; i < n; ++i)
37     {
38         cout << a[i];
39         if (i != n - 1)
40         {
41             cout << ' ';
42         }
43     }
44     cout << endl;
45     QuickSort(a, begin, keyi - 1, n);
46     QuickSort(a, keyi + 1, end, n);
47 }
48 int main()
49 {

```

```

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

提交时间: 2023/03/20 16:06:01

状态: 答案正确

分数: 10

题目: 7-1

编译器: C++ (g++)

内存: 604 KB

用时: 5 ms

用户: 202103151422

测试点	结果	分数	耗时	内存
1	答案正确	5	5 ms	436 KB
2	答案正确	5	4 ms	604 KB

代码:

```

1  #include <iostream>
2  using namespace std;
3  void Swap(int* p1, int* p2)
4  {
5      int tmp = *p1;
6      *p1 = *p2;
7      *p2 = tmp;
8  }
9  int PartSort(int* a, int begin, int end)
10 {
11     int left = begin, right = end;
12     int key = left;
13     while (left < right)
14     {
15         while (left < right && a[right] >= a[key])

```

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

```
1  #include <iostream>
2  using namespace std;
3
4  int ret = 0;
5  void solve(int n)
6  {
7      if (n == 1)
8      {
9          ret++;
10     }
11     else
12     {
13         for (int i = 2; i <= n; ++i)
14         {
15             if (n % i == 0)
16             {
17                 solve(n / i);
18             }
19         }
20     }
21 }
22
23 int main()
24 {
25     int n;
26     cin >> n;
27     solve(n);
28     cout << ret << endl;
29     return 0;
30 }
```

2.2 PTA提交代码结果截图

提交结果

提交时间	状态	分数	题目	编译器	内存	用时	用户
2023/03/20 14:17:13	答案正确	10	7-2	C++ (g++)	576 KB	7 ms	202103151422

测试点	结果	分数	耗时	内存
test5	答案正确	2	7 ms	448 KB
test4	答案正确	2	4 ms	436 KB
test3	答案正确	2	4 ms	436 KB
test2	答案正确	2	5 ms	576 KB
test1	答案正确	2	5 ms	568 KB

代码

```
1 #include <iostream>
2 using namespace std;
3
4 int ret = 0;
5 void solve(int n)
6 {
```

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。

$$12 = \begin{cases} 2 \times 6 \rightarrow 6 = \begin{cases} 2 \times 3 \rightarrow 3 = 3 \times 1 \text{ } ret++ \\ 3 \times 2 \rightarrow 2 = 2 \times 1 \text{ } ret++ \\ 6 \times 1 \text{ } ret++ \end{cases} \\ 3 \times 4 \rightarrow 4 = \begin{cases} 2 \times 2 \rightarrow 2 = 2 \times 1 \text{ } ret++ \\ 4 \times 1 \text{ } ret++ \end{cases} \\ 4 \times 3 \rightarrow 3 = 3 \times 1 \text{ } ret++ \\ 6 \times 2 \rightarrow 2 = 2 \times 1 \text{ } ret++ \\ 12 \times 1 \text{ } ret++ \end{cases}$$

3. 二叉搜索树的后序遍历序列

3.1 PTA提交代码截图


```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void isBST(const vector<int>& tree, bool& flag)
6  {
7      if (tree.empty())
8      {
9          return;
10     }
11     int root = tree.back();
12     vector<int> left_child_tree;
13     vector<int> right_child_tree;
14     for (size_t i = 0; i < tree.size() - 1; ++i)
15     {
16         if (tree[i] < root)
17         {
18             if (!right_child_tree.empty())
19             {
20                 flag = false;
21                 return;
22             }
23             left_child_tree.push_back(tree[i]);
24         }
25         else
26         {
27             right_child_tree.push_back(tree[i]);
28             if (tree[i + 1] < root)
29             {
30                 flag = false;
31                 return;
32             }
33         }
34     }
35     isBST(left_child_tree, flag);
36     isBST(right_child_tree, flag);
37 }
38 int main()
39 {
40     int n;
41     cin >> n;
42     vector<int> v;
43     v.resize(n, 0);
44     for (auto& e : v)
45     {
46         cin >> e;
47     }
48     bool flag = true;
49     isBST(v, flag);
50     if (flag)
51         cout << "Yes" << endl;
52     else
53         cout << "No" << endl;
54     return 0;
55 }

```

```

49  isBST(v, flag);
50  if (flag)
51  {
52      cout << "true" << endl;
53  }
54  else
55  {
56      cout << "false" << endl;
57  }
58  return 0;
59  }

```

3.2 PTA提交代码结果截图

提交结果

提交时间	状态	分数	题目	编译器	内存	用时	用户
2023/03/20 18:17:19	答案正确	20	7-3	C++ (g++)	476 KB	4 ms	202103151422

测试点	结果	分数	耗时	内存
1	答案正确	5	4 ms	456 KB
2	答案正确	5	4 ms	456 KB
3	答案正确	5	4 ms	324 KB
4	答案正确	5	4 ms	476 KB

代码

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void isBST(const vector<int>& tree, bool& flag)
6  {
7      if (tree.empty())
8      {
9          return;

```

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

```

1  #include <iostream>
2  #include <string>
3  #include <algorithm>
4  using namespace std;
5  string minimul(int i, string str);
6  string addStrings(string num1, string num2);
7  string multiply(string num1, string num2) {
8      if (num1 == "0" || num2 == "0")
9          return "0";
10     reverse(num2.begin(), num2.end());
11     string ret = "0";
12     for (auto e : num2)
13     {
14         ret = addStrings(ret, minimul(e - '0', num1));
15         num1 += '0';
16     }
17     return ret;
18 }
19 string minimul(int i, string str)
20 {
21     if (i == 0)
22         return "0";
23     reverse(str.begin(), str.end());
24     string ret;
25     int carry = 0;
26     for (auto e : str)
27     {
28         ret += (e - '0') * i % 10 + '0' + carry;
29         carry = (e - '0') * i / 10;
30     }
31     if (carry)
32         ret += (carry + '0');
33     reverse(ret.begin(), ret.end());
34     return ret;
35 }
36 string addStrings(string num1, string num2)
37 {
38     int jinwei = 0;
39     int a, b;
40     string ret;
41     for (int i = 0; i < max(num1.size(), num2.size()); i++)
42     {
43         a = i < num1.size() ? num1[num1.size() - 1 - i] : '0';
44         b = i < num2.size() ? num2[num2.size() - 1 - i] : '0';
45         ret.push_back(((a - '0' + b - '0' + jinwei) % 10) + '0');
46         jinwei = (a - '0' + b - '0' + jinwei) / 10;
47     }
48     if (jinwei)

```

```

49     ....ret += '1';
50     ....reverse(ret.begin(), ret.end());
51     ....return ret;
52 }
53 int max(int a, int b)
54 {
55     ....return a > b ? a : b;
56 }
57 int main()
58 {
59     ....string s1, s2;
60     ....while(cin >> s1 >> s2)
61         ....cout << multiply(s1, s2) << endl;
62     ....return 0;
63 }

```

4.2 PTA提交代码结果截图

提交结果

提交时间	状态	分数	题目	编译器	内存	用时	用户
2023/03/20 14:11:25	答案正确	100	7-4	C++ (g++)	452 KB	4 ms	202103151422

测试点	结果	分数	耗时	内存
1	答案正确	20	3 ms	324 KB
2	答案正确	80	4 ms	452 KB

代码

```

1  #include <iostream>
2  #include <string>
3  #include <algorithm>
4  using namespace std;
5  string minmul(int i, string str);
6  string addStrings(string num1, string num2);
7  string multiply(string num1, string num2) {
8      ....if (num1 == "0" || num2 == "0")
9          ....return "0";
10     ....reverse(num2.begin(), num2.end());
11     ....string ret = "0";
12     ....for (auto e : num2)
13         ....{

```

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。