

HUNAN UNIVERSITY

程序设计训练

报 告

学生姓名 郭江浩

学生学号 201926010206

专业班级 软件1902班

指导老师 陆绍飞

助 教 \_ \_刘园春、汪国庆、吴敏、陈冉龙\_\_\_\_\_\_

2020 年 9 月 5 日

目 录

[题1：数塔 3](#_Toc517944648)

[一、 问题描述 3](#_Toc517944649)

[二、 问题分析及边界条件 3](#_Toc517944650)

[三、 算法设计 3](#_Toc517944651)

[四、 详细设计（从算法到程序） 4](#_Toc517944652)

[五、 样例设计与测试 5](#_Toc517944653)

[六、 分析与总结 6](#_Toc517944654)

[题2：小希的数表 6](#_Toc517944648)

[一、 问题描述 7](#_Toc517944649)

[二、 问题分析及边界条件 7](#_Toc517944650)

[三、 算法设计 7](#_Toc517944651)

[四、 详细设计（从算法到程序） 7](#_Toc517944652)

[五、 样例设计与测试 8](#_Toc517944653)

[六、 分析与总结 9](#_Toc517944654)

[题3：二叉树遍历 9](#_Toc517944648)

[一、 问题描述 9](#_Toc517944649)

[二、 问题分析及边界条件 10](#_Toc517944650)

[三、 算法设计 10](#_Toc517944651)

[四、 详细设计（从算法到程序） 10](#_Toc517944652)

[五、 样例设计与测试 11](#_Toc517944653)

[六、 分析与总结 12](#_Toc517944654)

[题4：合法的括号串 12](#_Toc517944648)

[一、 问题描述 13](#_Toc517944649)

[二、 问题分析及边界条件 13](#_Toc517944650)

[三、 算法设计 13](#_Toc517944651)

[四、 详细设计（从算法到程序） 14](#_Toc517944652)

[五、 样例设计与测试 14](#_Toc517944653)

[六、 分析与总结 15](#_Toc517944654)

[题5：错误的里程表 15](#_Toc517944648)

[一、 问题描述 15](#_Toc517944649)

[二、 问题分析及边界条件 16](#_Toc517944650)

[三、 算法设计 16](#_Toc517944651)

[四、 详细设计（从算法到程序） 16](#_Toc517944652)

[五、 样例设计与测试 17](#_Toc517944653)

[六、 分析与总结 17](#_Toc517944654)

# 题1：数塔

## 问题描述

【问题描述】

给定一个数塔，如下图所示。在此数塔中，从顶部出发，在每一节点可以选择走左下或右下，一直走到底层。请找出一条路径，使路径上的数值和最大。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 9 |  |  |  |  |
|  |  |  | 12 |  | 15 |  |  |  |
|  |  | 10 |  | 6 |  | 8 |  |  |
|  | 2 |  | 18 |  | 9 |  | 5 |  |
| 19 |  | 7 |  | 10 |  | 4 |  | 16 |

【输入形式】

输入时第一行一个整数n，表示该数塔的行数，其余n行表示该塔每行的数值。

【输出形式】

输出包含两行，第一行为最大路径上的数值之和， 第二行n个数字为从上而下最大路径数值。

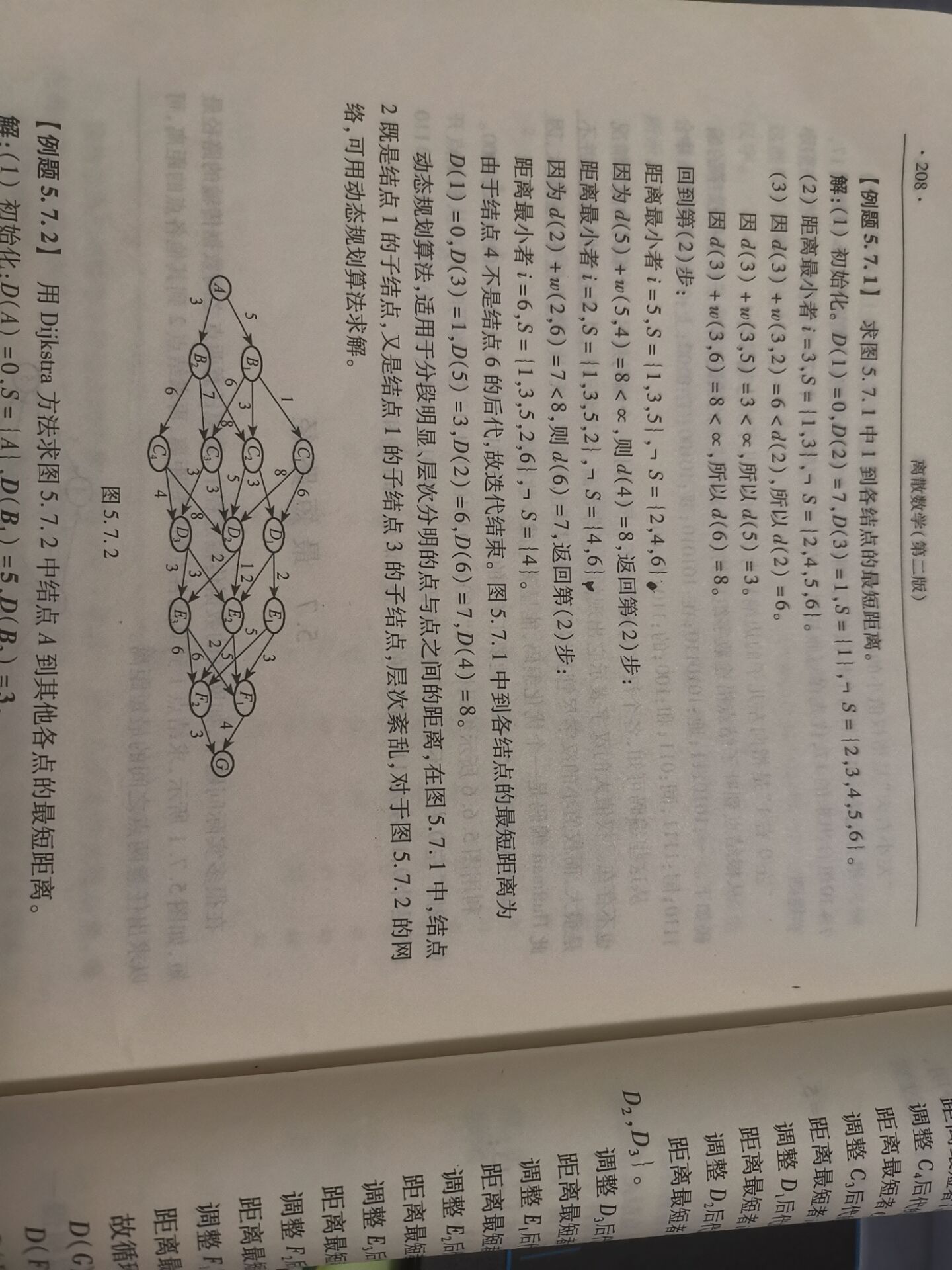
## 问题分析及边界条件

该题看上去比较简单，逻辑很简单，找到路径最大的即可。但是实际操作却比较复杂。这是一道动态规划的题。因为逻辑较为简单，且每一次计算都具有相似性，所以采用递归的算法较好。

## 算法设计

由上文分析可知，可采用递归的思想解答此问题。将每一行都视为第一行，只需求出下一行的最大路径即可。

当然，还可以使用其他算法。该题使用离散数学的Dijkstra算法也可以。只需将每一个数字看成下图中的一个点即可。

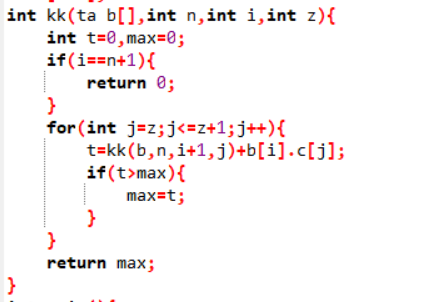


这两种方法都可以求出最短路径。

关于最长路径的记录，使用递归时暂未想到记录的方法，只能由结果反推出路径。而使用Dijkstra算法可能会有方法记录。

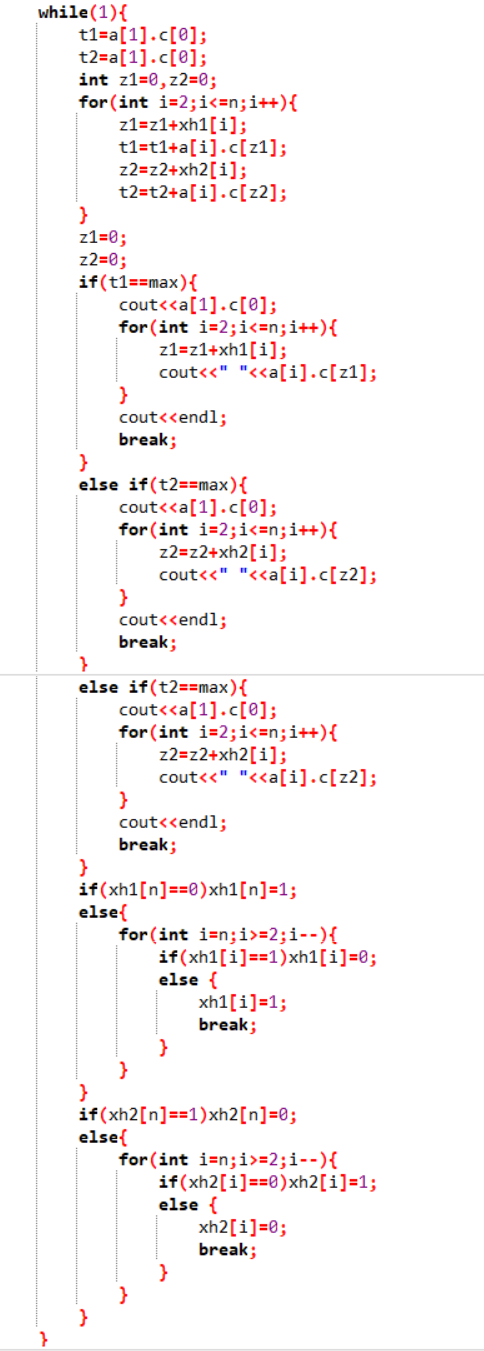
## 详细设计（从算法到程序）

这里使用的是递归法来求最大值



将每一行都看为第一行，求出该种情况下的最大值即可。这就是递归的思想。

对于记录路径，采用了利用循环进行穷举，反推出路径的方法。



## 样例设计与测试

【样例输入1】

5

9

12 15

10 6 8

2 18 9 5

19 7 10 4 16

【样例输出1】

59

9 12 10 18 10

【样例输入2】

3

8

15 16

11 18 19

【样例输出2】

43

8 16 19

【样例输入3】

4

8

4 5

6 9 7

8 1 2 4

【样例输出3】

26

8 4 6 8

## 分析与总结

该题存在的遗憾就是还未想到在递归的方法下找出记录路径的方法。

通过该题，练习了解决动态规划题目的方法。并且还运用了一些离散数学的知识，很有收获。

# 题2：小希的数表

## 问题描述

【问题描述】

Gardon 昨天给小希布置了一道作业，即根据一张由不超过 5000 的 N(3<=N<=100)个正整数组成的数表两两相加得到 N\*(N-1)/2 个和，然后再将它们排序。例如，如果数表里含有四个数 1，3，4，9，那么正确答案是 4，5，7，10，12，13。小希做完作业以后出去玩了一阵，可是下午回家时发现原来的那张数表不见了，好在她做出的答案还在，你能帮助她根据她的答案计算出原来的数表么？

【输入形式】

包含多组数据，每组数据以一个 N 开头，接下来的一行有按照大小顺序排列的 N\*(N-1)/2 个数，是小希完成的答案。文件最后以一个 0 结束。  
假设输入保证解的存在性和唯一性。

【输出形式】

对于每组数据，输出原来的数表。它们也应当是按照顺序排列的。

## 问题分析及边界条件

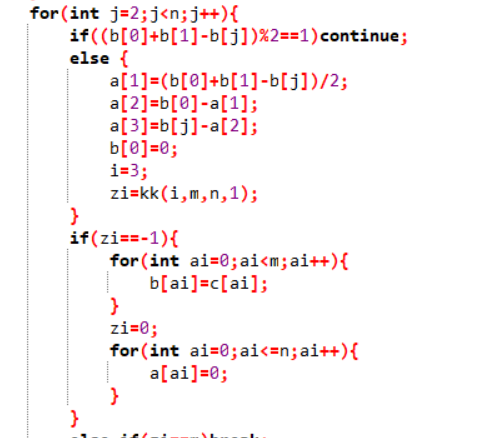
该问题本质是一个非齐次线性方程组的求解问题。显然，输入的最小的一个数和倒数第二小的数，必定是a1和a2、a1和a3的和，如果找到a2和a3的和，就可以求出这三个数，从而求出剩下的a4、a5······an

## 算法设计

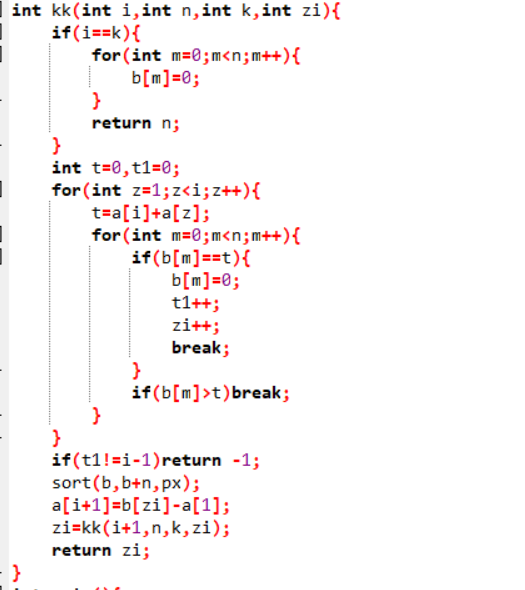
由以上分析可知，最小的一个数和倒数第二小的数，必定是a1和a2、a1和a3的和，但是a2和a3的和不确定，因为题目确定只有一个解，所以，必定只有一个数可以得出解，所以使用穷举的方法，在剩下的数里找出符合的，即a2与a3的和。之后便可求出a1、a2、a3。之后再从数表中将只包含a1、a2、a3的和去掉，那么剩下的最小的必定是a1与a4的和，然后再将只包含a1、a2、a3、a4的和从数表中去掉，剩下最小的就是a1与a5的和，然后以此类推，就可得到所有的数。

## 详细设计（从算法到程序）

首先认为第三数就是a2+a3从而计算出a1、a2、a3，然后带入递归函数，进行计算，如果无解，那么下一个数认为是a2+a3，以此类推，直到找到解为止。



递归算法即为每次算出ai，并把只含有a1、a2···ai的删去。



即可算出。

## 样例设计与测试

【样例输入1】

4

4 5 7 10 12 13

4

5 6 7 8 9 10

0

【样例输出1】

1 3 4 9

2 3 4 6

【样例输入2】

3

5 6 9

0

【样例输出2】

1 4 5

【样例输入3】

4

4 6 8 10 12 14

0

【样例输出3】

1 3 5 9

## 分析与总结

该题考查了递归函数的使用，并不复杂，逻辑清楚之后就比较简单了

# 题3：二叉树遍历

## 问题描述

【问题描述】

二叉树是一种非常重要的[数据结构](http://lib.csdn.net/base/datastructure)，非常多其他数据结构都是基于二叉树的基础演变而来的。对于二叉树，深度遍历有前序、中序以及后序三种遍历方法。

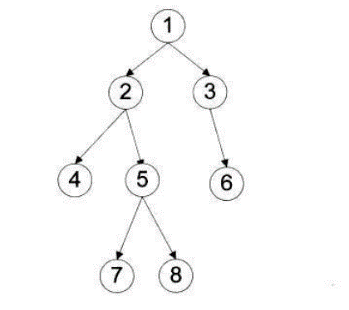
三种基本的遍历思想为：

前序遍历：根结点 ---> 左子树 ---> 右子树

中序遍历：左子树---> 根结点 ---> 右子树

后序遍历：左子树 ---> 右子树 ---> 根结点

比如，求以下二叉树的各种遍历



前序遍历：1  2  4  5  7  8  3  6

中序遍历：4  2  7  5  8  1  3  6

后序遍历：4  7  8  5  2  6  3  1

需要你编写程序解决的问题是：已知一个二叉树的前序遍历和中序遍历的结果，给出该二叉树的后序遍历的结果。

【输入形式】

有多组测试数据，每组测试数据三行，每组测试数据第一行只有一个正整数n，表示二叉树节点的数目，n=0意味着输入结束并且不需要处理。

每组测试数据第二行是二叉树的前序遍历的结果，是一个长度为n的字符串，每个节点由一个字符表示，字符是大小写英文字母及10个数字,不同的节点用不同的字符表示，也即无论前序遍历和中序遍历的字符串中没有重复的字符。

每组测试数据第二行是二叉树的中序遍历的结果，也是一个长度为n的字符串。

40%的测试数据1 ≤ n≤ 10；

30%的测试数据1 ≤ n≤ 20；

20%的测试数据1 ≤ n≤ 40；

10%的测试数据1 ≤ n≤ 62；

【输出形式】

对于每组测试数据，输出一行，是一个长度为n的字符串，表示二叉树后序遍历的结果。

## 问题分析及边界条件

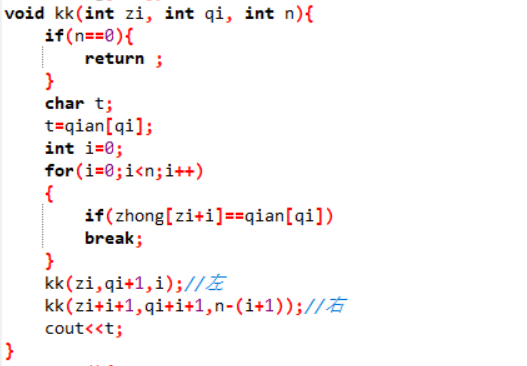
由题目中给出的提示，根节点在前序遍历的第一个，在中序遍历的中间，在后序遍历的最后一个。而对于中序遍历来说，左边为左子树，右边为右子树，对于每个子树来说，他们的遍历同样有之前的规律。所以，可以运用递归的方法找到根节点来完成这个题目。

## 算法设计

由以上分析可知，重点是找到每棵树的根节点，把子树都看成一棵新的数即可。在只要在中序遍历中寻找到与前序遍历第一个数字相同的数字，即可确定根节点，然后在前序遍历中找到中序遍历根节点左边的数字，即可确定子树的根节点······以此类推，最后可以写出后序遍历。

## 详细设计（从算法到程序）

采用递归的方法



每次输出根节点即可

## 样例设计与测试

【样例输入1】

8

12457836

42758136

4

abcd

abcd

4

abcd

dcba

0

【样例输出1】

47852631

dcba

dcba

【样例输入2】

9

zsho7pwmG

so7hpzmwG

6

gNkoVO

kNVogO

0

【样例输出2】

7ophsmGwz

kVoNOg

【样例输入3】

1

i

i

7

0DYV4lJ

04VYDJl

0

【样例输出3】

i

4VYJlD0

## 分析与总结

该题需要有一定的离散数学知识。除此之外还需要熟练掌握递归函数。

# 题4：合法的括号串

## 问题描述

【问题描述】

一个合法的括号串，是指只包含括号的串，如果满足如下条件：

（1）<> () [] {} 这四对括号是合法的；

（2）如果r是合法括号串，则<r> (r) [r] {r}也是；

（3）如果r，s是合法括号串，则rs也是；

所以<<>> , [<>{}(())],[({<>})]是合法的括号串，而)(,[( ])就不是。

【输入形式】

输入第一行正整数t (10 ≤ n ≤ 100)，表示有多少组测试数据。

后面有t行，每行一个只包含8种括号符号的括号串。

40%的括号串的长度L 2 ≤ L≤ 20；

30%的括号串的长度L 2 ≤ L≤ 200；

20%的括号串的长度L 2 ≤ L≤ 2000；

10%的括号串的长度L 2 ≤ L≤ 20000；

【输出形式】

对于每组测试数据，如果括号串是合法的，输出“**Yes**”（输出没有引号）占一行，否则，输出“**No**”（输出没有引号）占一行。

## 问题分析及边界条件

该问题看起来比较复杂，仔细分析可得，只要找出合法的括号，然后删去，继续找就行。

## 算法设计

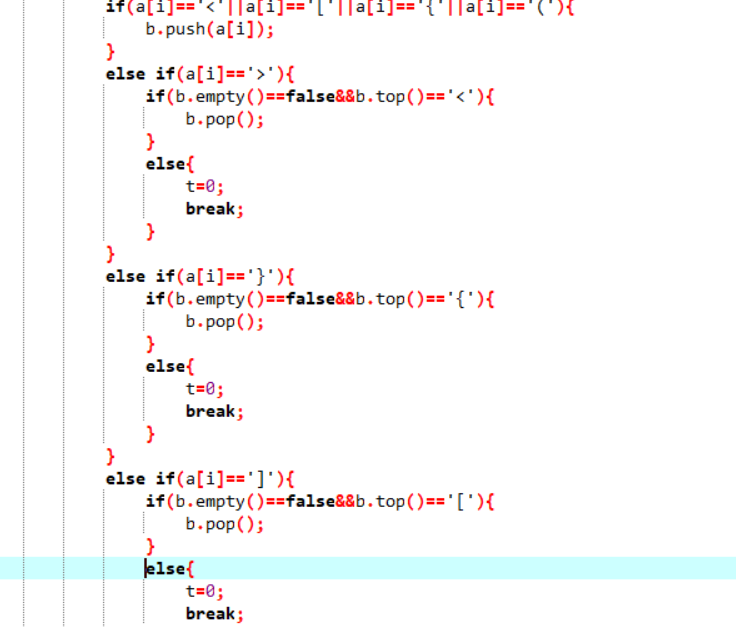
有两种想法。第一种，将这串括号串看成一个字符串，通过find函数找出合法的括号，然后删掉，继续寻找，知道找不到为止，再判断字符串是否为空，即可得到是否是合法的括号串

该方法有时会运行时间过长。

第二种，将括号串输入栈中，每次只需要判断栈顶的字符和新的字符是否是一个合法的括号即可。若是则继续，若不是，就不是一个合法的括号串。

## 详细设计（从算法到程序）

采用第二种方法，利用栈来做。



如果是合法的括号就弹出栈顶元素，这样就可大大减少运行时间

## 样例设计与测试

【样例输入1】

2

<<>>

)(

【样例输出1】

Yes

No

【样例输入2】

2

[<>{}(())]

[({<>})]

【样例输出2】

Yes

Yes

【样例输入3】

2

[(])

<([{

【样例输出3】

No

No

## 分析与总结

通过该题，熟悉了栈的应用，并且也复习了find函数的使用。在第一次做这道题时，使用的算法运行时间过长，然后决定换成使用栈，这样大大减少了时间。

# 题5：错误的里程表

## 一、问题描述

【问题描述】

三月八日，小明买了台新车。但很快小明发现汽车的里程表有问题：里程表上每一位都不显示数字3和数字8，也就是说直接从数字2跳到数字4，直接从数字7跳到数字9。小明纳闷：这车到底行驶里程是多少。

现在，小明向你求助：根据里程表显示的数字，给出真实的行驶里程。

【输入形式】

输入有多组测试数据。

输入第一行正整数T，表示有多少组测试数据。

后面有T行，每行一个非负整数，表示里程表显示数字，里面不含有数字3和8。该数字不超过10位。

40%的测试数据组数T  10≤T≤ 102；

30%的测试数据组数T  102≤T≤ 103；

20%的测试数据组数T  103≤T≤ 104；

10%的测试数据组数T  104≤T≤ 105

【输出形式】

对于每组测试数据，输出一个整数占一行：真实的行程里程。

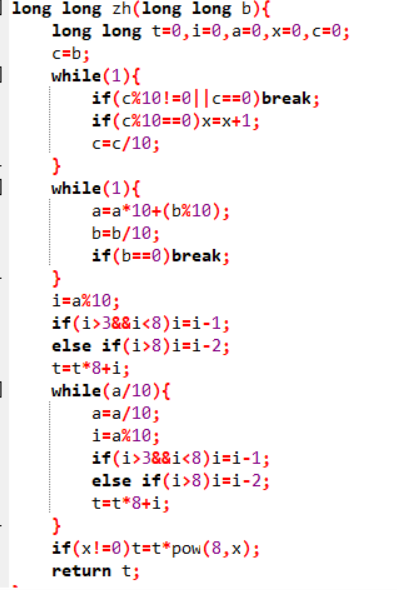
## 二、问题分析及边界条件

该问题看上去比较复杂，但实际上就是一个进制转换问题，将八进制转化为十进制即可，需要注意的是4、5、6、7对应的是3、4、5、6而9对应的是7.

## 三、算法设计

由上面的分析可知，只需要将这个八进制数转化为十进制即可，将这个数的每一位依次取出，乘对应的8的n次方，然后再加起来即可，需要注意4以后的数字的对应问题。

**四、详细设计（从算法到程序）**



输入数字后通过函数进行计算，注意4之后数字的对应关系即可。

## 五、样例设计与测试

【样例输入1】

6

0

1

12

159

111224459

124567976

【样例输出1】

0

1

10

103

19212007

21913077

【样例输入2】

5

6

569

47661

121211

11001

【样例输出2】

5

303

15721

41609

4609

【样例输入3】

3

66666

555555

4444444

【样例输出3】

23405

149796

898779

## 六、分析与总结

通过这道题复习了进制的转换问题，很有收获。