解题报告

一. 试题翻译：

一棵树由结点和与它相连子树（有 0，1 或者 2）组成，这些子树被称作儿子。树的描述是一系列数字，如果这个树的儿子数是：

1、0，那么树的描述仅仅是一列 0；

2、1，那么树的描述是由 1 开始后面跟着其儿子的描述；

3、2，那么树的描述是由 2 开始后面跟着第一个儿子的描述，然后是第二个儿子的描述。

树的每个点都必须涂成红、绿或者蓝色，但是我们涂的时候必须遵守下面的规则：

1、结点和它的儿子不能有一样的颜色，

2、如果结点有两个儿子，那么它们必须有不同的颜色。

问应该有多少个结点是绿色的？

【任务】：编写一个程序

1、从文件 TRO．IN 读入树的描述；

2、计算能够涂成绿色的结点的最大数和最小数；

3、将结果写入文件 TRO．OUT。

【输入】：

文件 TRO．IN 唯一一行由一个单词组成（不超过 10000 字母），该单词是树的描述。

【输出】：

文件 TRO．OUT 的唯一一行有两个由空格分开的整数

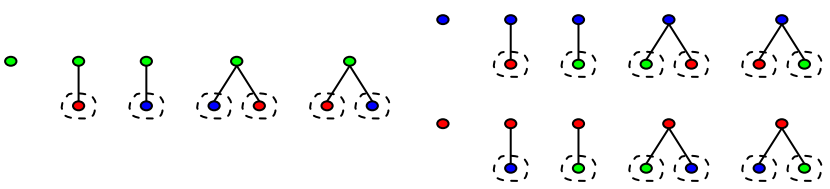
二. 试题分析：

这显然是一道求最优解的题目，而且明显是树形的结构，所以可以用动态规划来解决。

先来看求最大值：

设 $F1[i]$、$F2[i]$、$F3[i]$ 分别表示把结点 i 涂成绿色、蓝色、红色则以结点 i 为根的子树中最多有多少个绿色结点。由于只要把以 i 为根的子树中所有蓝色结点的颜色换成红色、红色结点的颜色换成蓝色，就可以得到一棵满足条件并且含绿色结点数同样多的树，所以 $F2[i]=F3[i]$，也就是说可以用 $F2[i]$ 表示把结点 i 涂成蓝色或红色则以结点 i 为根的子树中最多有多少个绿色结点。

那么，由于结点和它的儿子以及儿子之间不能有一样的颜色，所以有某个结点与其儿子结点的着色情况只有以下几种情况：



所以也就不难得到：

$$F1[i]=\begin{cases}1 & \text{结点 } i\text{无儿子结点}\\ F2[l]+1 & \text{结点 } i\text{有一个儿子结点}\\ F2[l]+F2[r]+1 & \text{结点 } i\text{有两个儿子结点}\end{cases}$$

$$F2[i]=\begin{cases}0 & \text{结点 } i\text{无儿子结点}\\ Max\{F1[l],F2[l]\} & \text{结点 } i\text{有一个儿子结点}\\ Max\{F1[l]+F2[r],F2[l]+F1[r]\} & \text{结点 } i\text{有两个儿子结点}\end{cases}$$

l为i的第一个儿子结点编号　　(如果存在儿子结点）

r为i的第二个儿子结点编号　　(如果存在两个儿子结点　　）

Max 表示取最大值

根据题目中的给定这棵树的方法，可知所有的点都是按照Top排序排好了的，所以可以按编号从大到小的顺序，依次算出每个 $F1[i]$、$F2[i]$ 。现在只剩下一个问题，如何确定方程中 l、r 的值呢？由题目中给定结点的方法显然有：

$l=i+1$，但 r 并不是可以直接得到的。稍加分析后便会发现，只要知道以 l 为根的子树中最后一个结点的编号那就容易得到 r 的值了——

$r=Link[l]+1$，而以 l 为根的子树中最后一个结点的编号 $Link[l]$ 是容易用递推式求解的。

$$Link[l]=\begin{cases}l & \text{结点 } l\text{无儿子结点}\\ Link[l+1] & \text{结点 } l\text{有一个儿子结点}\\ Link[Link[l+1]+1] & \text{结点 } l\text{有两个儿子结点}\end{cases}$$

计算 $Link[l]$ 的过程可以在动态规划求解之前进行，所以总的复杂度只有 $O(N)$，N 为结点总数。

而求最小值（最少绿色结点数）的方法与求最大值（最多绿色结点数）的方法大同小异，只要将动态规划方程中的 Max 换成 Min （取较小值）就可以了。

上述算法对空间的需求量也相当小，只需要 $7NBytes<70KB$ 。

三.　总结及延伸：

像树形结构的求最优解的题目是比较多的，比如 Sgoi 中的“皇宫看守”等，他们一般都可以用动态规划来实现，并且一般来说都是从底向上的计算每一棵子树的最有值。

四.　程序清单：

```
program Tro;

const
```

```
  Fn1 = 'Tro.In';
  Fn2 = 'Tro.Out';
  MaxN = 10000;

type
  List = array[1 .. MaxN] of Integer;

var
  Min, Max, N: Integer;
  Num: array[1 .. MaxN] of Byte;
  F1, F2: List;
  Link: ^List;

procedure Init;
var
  Ch: Char;
begin
  Assign(Input, Fn1); Reset(Input);
  while not Eoln do
    begin
      Read(Ch); Inc(N); Num[N] := Ord(Ch) - 48
    end;
  Close(Input)
end;

procedure Prepare;
var
  i: Integer;
begin
  New(Link);
  for i := N downto 1 do
    case Num[i] of
      0: Link^[i] := i;
      1: Link^[i] := Link^[i + 1];
      2: Link^[i] := Link^[Link^[i + 1] + 1]
    end
end;

procedure CalcMin;
var
  i, j: Integer;
begin
  for i := N downto 1 do
    case Num[i] of
```

```
      0: begin
        F1[i] := 1; F2[i] := 0
      end;
      1: begin
        F1[i] := F2[i + 1] + 1;
        if F1[i + 1] < F2[i + 1] then F2[i] := F1[i + 1]
        else F2[i] := F2[i + 1]
      end;
      2: begin
        j := Link^[i + 1] + 1;
        F1[i] := F2[i + 1] + F2[j] + 1;
        if F1[i + 1] + F2[j] < F2[i + 1] + F1[j] then
          F2[i] := F1[i + 1] + F2[j]
        else F2[i] := F2[i + 1] + F1[j]
      end
    end;
  if F1[1] < F2[1] then Min := F1[1]
  else Min := F2[1]
end;

procedure CalcMax;
var
  i, j: Integer;
begin
  for i := N downto 1 do
    case Num[i] of
      0: begin
        F1[i] := 1; F2[i] := 0
      end;
      1: begin
        F1[i] := F2[i + 1] + 1;
        if F1[i + 1] > F2[i + 1] then F2[i] := F1[i + 1]
        else F2[i] := F2[i + 1]
      end;
      2: begin
        j := Link^[i + 1] + 1;
        F1[i] := F2[i + 1] + F2[j] + 1;
        if F1[i + 1] + F2[j] > F2[i + 1] + F1[j] then
          F2[i] := F1[i + 1] + F2[j]
        else F2[i] := F2[i + 1] + F1[j]
      end
    end;
  if F1[1] > F2[1] then Max := F1[1]
  else Max := F2[1]
```

```
end;

procedure Print;
begin
  Assign(Output, Fn2); Rewrite(Output);
  Writeln(Max, ' ', Min);
  Close(Output)
end;

begin
  Init;
  Prepare;
  CalcMin;
  CalcMax;
  Print
end.
```