NOIP2005 信息学奥林匹克分区联赛

解题报告

第一题：谁拿了最多的奖学-Scholar

[问题评估]

这个题目据问题本身而言是相当简单的，没有牵涉到过多的算法，属于普及型试题。同时也是对实际问题一种分析和判断。总的来看，本题在方向上，向现实问题迈出了一步，是信息学和生活有了更多的联系。

问题的算法是模拟。当中唯一的难点就是数据处理，考察点为数据库的建立和统计。

[程序实现]

由于程序数据范围只有 100，当中不牵涉到数据移动，所以用一个纪录型数组，或者多个数组均可，在这里我们使用纪录型来描述。

对于输入数据有两种方式来实现。

法一〉逐个字符累加。

首先定义 C:char; 然后利用 Until c=‘ ’;作为终止符，将读入的字符连接存储到 a[i].name 中。

代码为：

Repeat read(c); a[i].name:=a[i].name+c; until c=’ ‘;
a[i].name:=copy(a[i].name,1,length(a[i].s)-1);

这样做的好处是，后面的值可以直接用 read 语句读入。但是最后一个值后，要记得 readln;

法二〉一次读入,然后分离。

这样做需要逐个分离，对本题来说稍显复杂，但对 NOIP 来说此方法必须掌握，有的时候一定要用。

具体实现，读入一个字符串 S。利用 pos(‘ ‘,s);找出空格位置。再利用 Copy 函数，和 Val 函数进行截取，和转换。

部分代码：(s:string;j,ok:integer)

```
readln(s);
j:=pos(‘ ‘,s);
a[i].name:=copy(s,1,j-1);
s:= copy(s,j+1,50); //当长度〉字符串长度是，为后面全部截取。
j:=pos(‘ ‘,s);
Val(copy(s,1,j-1),a[i].qp,ok);
s:= copy(s,j+1,50);
```

…..

对于符号用 if 语句作一下判断就是了，太 easy 不写了，后面还有几个值，用同样方法处理就可以了。

以上完成了数据库的建库工作，后面是统计，当然，我们在没读完一行数据后就可进行统计。用 If 语句判断他是否能得到相应的分值即可。分 5 条 If 语句写，每回可以就加入相应的分值。

将每个的分值汇总计入到总数变量 ZD 当中。与当前最大值进行比较，得到 Max 对应的 I 值。后面就是输出的问题了。

[小结、注意]

本题为简单题，只要思路明确清晰，就可 AC。时间复杂度 O(n)。但有一个细节，ZD 变量必须定义 Longint 或以上类型否则会 Error201 的。

第二题 过河-River

[问题分析]

此题初看是一个典型的搜索题。从河的一侧到河的另一侧，要找最少踩到的石头数。但从数据范围来看。1..109 长度的桥。就算是 O(n)的算法也不能在一秒内出解。

如果搜索石子，方法更困难。这要考虑到前面以及后面连续的石子。若换一种方法。用动态规划，以石子分阶段的一维动规，时间复杂度是 O(n2)。最多也只有 100×100 的时间。但是这样分状态就十分复杂。因为石头的分布是没有任何规律，而且会有后效性。

这样只好有回到搜索。搜索石子会和动规一样没有规律。我们一桥的长度为对象进行搜索然后再加上一个巧妙的剪枝就可以在很短的时间内出解。可以号称为 O(m2)。[批注：号称一词已成为湖南 OI 本世纪流行词汇]

[题目实现]

先以时间为对象进行搜索。时间复杂度为 O(L)。从桥的一侧到另一侧，中间最多只有 100 个石子。假设桥长为最大值(109)，石头数也为最大值(100)。这样中间一定会有很多“空长条” (两个石子中的空地)，处理时把这些跳过，就只会有 M 次运算。关键是找出每一个可以跳过的“空长条”。

我们可以先把青蛙可以跳出的所有可能求出，然后就可以求出可以忽略的“空长条”。

[特殊算法]

a[i]:前 i 个坐标中石子最小个数,初始为第 i 个坐标的石子个数

b[i]:第 i 个石子坐标

动规

a[0]=0;

对 n>=t

a[n]=min{a[n]+a[n-s],a[n]+a[n-s-1], ...,a[n]+a[n-t]}

对 s=<n<t

a[n]=max{a[n]+a[n-s],a[n]+a[n-s-1],...,a[n]+a[0]}

但由于 n 较大直接动规会超时。所以要将 n 压缩

查看坐标，可以发现，如果 b[i]-b[i-1]>t，显然对于 b[i-1]+t<n<b[i]，a[n]总是等于 a[b[i-1]]..a[b[i-1]+t]中的数，因此可对其进行压缩。

注意，在计算过程中，由于其中有一些坐标是永远走不到的，因此需要用一个布尔型的数组 c[n]进行判断。方法是，对于 c[n],如果 0<n<s,则 c[n]为 false，如果 n>s,c[n-t],c[n-t+1],...,c[n-s]都为 false,则 c[n]也为 false。

第三题 篝火晚会-fire

[问题评估]

此题或许大多数人会觉得很麻烦。或许有人会选择搜索来做，显然，50000 的数据量不可

能允许搜索不超时。或许有人会用贪心，但是却无从下手。
动态规划？怎么划阶段更是一个难题。然而，此题却不是考察选手的算法的，而是考察你从题目中找出基本核心的能力。

[题目实现]
题目给你的初始状态是一个回路，从第一个同学前断开，不难看出这是一个严格的上升序列。而输入的数据也可以将之构成一个包含所有同学的回路，否则就达不到没个人的愿望。

我们可以用两的数组来储存两个数组的状态，初始状态为 st，目标状态为 en。st[i]=i, i<=n。而输入数据我门可以先用一个二维 E 数组储存，E[I，1]即表示第 I 个人的第一个愿望。我们将目标状态数组 en 的第一个元素赋值为 1，然后就可以把 s[1]的第一个愿望加入数组为 s[2]，依次我们可以逐个加入，加入没个元素的时候，还要判断一下每个元素是否在数组当中，如果在，那就取第 2 个愿望。如果第二个愿望也在数组当中，那么我们的目标状态的数组也就构造完成了。

如果每个人的愿望都能实现，显然，目标状态的数组的元素必定是 N，而假如不是，那么就可以输出-1 了。
此时，问题就显的简单些了，如何让一个数组从一中状态变成另一种状态，相信有很多方法，可还是个麻烦事。
从目标状态转换成初始状态的步数是等同于初始状态转换成目标状态，而此时再看看初始状态的数组，相信你已经看出些疑端了吧！

排序！！！

对，其实从目标状态转换成初始状态的过程就是一个排序的过程，而且还是一个最简单的冒泡排序的过程！
到了这了，问题已经明了了，题目所求就是每次进行连续交换的人数总和，这样，一个看似复杂的题目就变的异常的简单了！而题目 2 秒的时间限制更是保证了冒泡排序经过一些优化以及剪枝后不会超时。
但是，千万不能用其他的排序法来解决。虽然能让你的程序变的更快，却同时你也得不正确的解！

第四题 等价表达式-Equal
[问题分析]
这道题目拿到手后，一般可以想到的方法就是可不可以将所有的表达式全部转化为最简形式，这时，你就想到了一种一般的解决方案。即将所有的表达式全部化为最简，然后再计算，这种方法是一种准确的方法。但要在考场上实现，有一些麻烦，需要一些时间。这种方法的解决过程是：先将阶乘化乘，再展开。计算同时合并同类项，留下一个数组。然后比较每一个表达式的数组与题目数组是否相同，时间效率也并不高。那么怎么办呢？

[模型建立和实现]
我们这里介绍一种利用必要条件的解决方案。
即两个表达式如果等价，那么无论 a 为何值，两个表达式计算出的值都相等。这时，我们以不同的 a 值代入各式，可以快速排斥那些不同的表达式，留下的便是等价的了。

我们怎样取值呢？这里推荐几种有效的方法：
1>取随机函数生成的数列。这种方法比较有效，无规律。
2>取伪随机数列。这是一种比较便于人工控制的手段。
3>取实数。由于其他皆为整数，小数部分便成为判断的优越条件。
一般情况下取 4~7 组值便可通过极大部分情况，实数需要更小。如果取更多组的值，便可以通过几乎所有的情况（将两式连立，只有当取值都为方程的解时才会出现误判，显然这样的几率是极小的）。
补充：这道题可能会有选手在数据类型上选择不当，导致一些情况会出现溢出。

[表达式求值]
经过上面的叙述，难点落在了表达式求值上，在这里我们介绍一下最一般、最简单的方法，栈运算。

用栈实现表达式求值的方法：

首先，我们要给每一个符号一个优先级：

符号
+ -
* /
^
(
)

栈内级别
2
4
6
0
8

栈外级别
1
3
5
8
0

可以看到，优先级高的符号先算。为了方便起见，我们定义特殊符号#，它级别最低（赋-1）
先将它置栈底，然后依次读入每个字符，如果是数字则入数栈。如果是符号，就与栈顶符号比较优先级。如果相等，则退栈，读下一字符。如果栈外大，则入栈。如果栈内大，则取栈顶元素与数栈最顶 2 元素运算，结果入数栈。这个符号继续处理（再与栈顶比较）。

直到读到最后符号#使栈底#出栈时。数栈顶即为表达式结果。

NOIP2005 信息学奥林匹克分区联赛

解题报告

OIBH.KuYe.Cn［麓山 NOI 战队］

第一题：谁拿了最多的奖学-Scholar

[问题评估]

这个题目据问题本身而言是相当简单的，没有牵涉到过多的算法，属于普及型试题。同时也是对实际问题一种分析和判断。总的来看，本题在方向上，向现实问题迈出了一步，是信息学和生活有了更多的联系。

问题的算法是模拟。当中唯一的难点就是数据处理，考察点为数据库的建立和统计。

[程序实现]

由于程序数据范围只有 100，当中不牵涉到数据移动，所以用一个纪录型数组，或者多个数组均可，在这里我们使用纪录型来描述。

对于输入数据有两种方式来实现。

法一〉逐个字符累加。

首先定义 C:char; 然后利用 Until c=‘ ’;作为终止符，将读入的字符连接存储到 a[i].name 中。

代码为：

```
Repeat read(c); a[i].name:=a[i].name+c; until c=’ ‘;
a[i].name:=copy(a[i].name,1,length(a[i].s)-1);
```

这样做的好处是，后面的值可以直接用 read 语句读入。但是最后一个值后，要记得 readln;

法二〉一次读入,然后分离。

这样做需要逐个分离，对本题来说稍显复杂，但对 NOIP 来说此方法必须掌握，有的时候一定要用。

具体实现，读入一个字符串 S。利用 pos(‘ ‘,s);找出空格位置。再利用 Copy 函数，和 Val 函数进行截取，和转换。

部分代码：(s:string;j,ok:integer)

```
readln(s);
j:=pos(‘ ‘,s);
a[i].name:=copy(s,1,j-1);
s:= copy(s,j+1,50); //当长度〉字符串长度是，为后面全部截取。
j:=pos(‘ ‘,s);
Val(copy(s,1,j-1),a[i].qp,ok);
s:= copy(s,j+1,50);
…..
```

对于符号用 if 语句作一下判断就是了，太 easy 不写了，后面还有几个值，用同样方法处理就可以了。

以上完成了数据库的建库工作，后面是统计，当然，我们在没读完一行数据后就可进行统计。用 If 语句判断他是否能得到相应的分值即可。分 5 条 If 语句写，每回可以就加入相应的分值。

将每个的分值汇总计入到总数变量 ZD 当中。与当前最大值进行比较，得到 Max 对应的 I 值。后面就是输出的问题了。

[小结、注意]

本题为简单题，只要思路明确清晰，就可 AC。时间复杂度 O(n)。但有一个细节，ZD 变量必须定义 Longint 或以上类型否则会 Error201 的。

第二题 过河-River

[问题分析]

此题初看是一个典型的搜索题。从河的一侧到河的另一侧，要找最少踩到的石头数。但从数据范围来看。1..109 长度的桥。就算是 O(n)的算法也不能在一秒内出解。

如果搜索石子，方法更困难。这要考虑到前面以及后面连续的石子。若换一种方法。用动态规划，以石子分阶段的一维动规，时间复杂度是 O(n2)。最多也只有 100×100 的时间。但是这样分状态就十分复杂。因为石头的分布是没有任何规律，而且会有后效性。

这样只好有回到搜索。搜索石子会和动规一样没有规律。我们一桥的长度为对象进行搜索，然后再加上一个巧妙的剪枝就可以在很短的时间内出解。可以号称为 O(m2)。[批注：号称一词已成为湖南 OI 本世纪流行词汇]

[题目实现]

先以时间为对象进行搜索。时间复杂度为 O(L)。从桥的一侧到另一侧，中间最多只有 100 个石子。假设桥长为最大值(109)，石头数也为最大值(100)。这样中间一定会有很多“空长条” (两个石子中的空地)，处理时把这些跳过，就只会有 M 次运算。关键是找出每一个可以跳过的“空长条”。

我们可以先把青蛙可以跳出的所有可能求出，然后就可以求出可以忽略的“空长条”。

[特殊算法]

a[i]:前 i 个坐标中石子最小个数,初始为第 i 个坐标的石子个数

b[i]:第 i 个石子坐标

动规

a[0]=0;

对 n>=t

a[n]=min{a[n]+a[n-s],a[n]+a[n-s-1], ...,a[n]+a[n-t]}

对 s=<n<t

a[n]=max{a[n]+a[n-s],a[n]+a[n-s-1],...,a[n]+a[0]}

但由于 n 较大直接动规会超时。所以要将 n 压缩

查看坐标，可以发现，如果 b[i]-b[i-1]>t，显然对于 b[i-1]+t<n<b[i]，a[n]总是等于 a[b[i-1]]..a[b[i-1]+t]中的数，因此可对其进行压缩。

注意，在计算过程中，由于其中有一些坐标是永远走不到的，因此需要用一个布尔型的数组 c[n]进行判断。方法是，对于 c[n],如果 0<n<s,则 c[n]为 false，如果 n>s,c[n-t],c[n-t+1],...,c[n-s]都为 false,则 c[n]也为 false。

第三题 篝火晚会-fire

[问题评估]

此题或许大多数人会觉得很麻烦。或许有人会选择搜索来做，显然，50000 的数据量不可能允许搜索不超时。或许有人会用贪心，但是却无从下手。

动态规划？怎么划阶段更是一个难题。然而，此题却不是考察选手的算法的，而是考察你从题目中找出基本核心的能力。

[题目实现]

题目给你的初始状态是一个回路，从第一个同学前断开，不难看出这是一个严格的上升序列。而输入的数据也可以将之构成一个包含所有同学的回路，否则就达不到没个人的愿望。

我们可以用两的数组来储存两个数组的状态，初始状态为 st，目标状态为 en。st[i]=i, i<=n。而输入数据我门可以先用一个二维 E 数组储存，E[I，1]即表示第 I 个人的第一个愿望。我们将目标状态数组 en 的第一个元素赋值为 1，然后就可以把 s[1]的第一个愿望加入数组为 s[2]，依次我们可以逐个加入，加入没个元素的时候，还要判断一下每个元素是否在数组当中，如果在，那就取第 2 个愿望。如果第二个愿望也在数组当中，那么我们的目标状态的数组也就构造完成了。

如果每个人的愿望都能实现，显然，目标状态的数组的元素必定是 N，而假如不是，那么就可以输出-1 了。

此时，问题就显的简单些了，如何让一个数组从一中状态变成另一种状态，相信有很多方法，可还是个麻烦事。

从目标状态转换成初始状态的步数是等同于初始状态转换成目标状态，而此时再看看初始状态的数组，相信你已经看出些疑端了吧！

排序！！！

对，其实从目标状态转换成初始状态的过程就是一个排序的过程，而且还是一个最简单的冒泡排序的过程！

到了这了，问题已经明了了，题目所求就是每次进行连续交换的人数总和，这样，一个看似复杂的题目就变的异常的简单了！而题目 2 秒的时间限制更是保证了冒泡排序经过一些优化以及剪枝后不会超时。但是，千万不能用其他的排序法来解决。虽然能让你的程序变的更快，却同时你也得不正确的解！

第四题 等价表达式-Equal

[问题分析]

这道题目拿到手后，一般可以想到的方法就是可不可以将所有的表达式全部转化为最简形式，这时，你就想到了一种一般的解决方案。即将所有的表达式全部化为最简，然后再计算，这种方法是一种准确的方法。但要在考场上实现，有一些麻烦，需要一些时间。这种方法的解决过程是：先将阶乘化乘，再展开。计算同时合并同类项，留下一个数组。然后比较每一个表达式的数组与题目数组是否相同，时间效率也并不高。那么怎么办呢？

[模型建立和实现]

我们这里介绍一种利用必要条件的解决方案。

即两个表达式如果等价，那么无论 a 为何值，两个表达式计算出的值都相等。这时，我们以不同的 a 值代入各式，可以快速排斥那些不同的表达式，留下的便是等价的了。

我们怎样取值呢？这里推荐几种有效的方法：

1>取随机函数生成的数列。这种方法比较有效，无规律。

2>取伪随机数列。这是一种比较便于人工控制的手段。

3>取实数。由于其他皆为整数，小数部分便成为判断的优越条件。

一般情况下取 4~7 组值便可通过极大部分情况，实数需要更小。如果取更多组的值，便可以通过几乎所有的情况（将两式连立，只有当取值都为方程的解时才会出现误判，显然这样的几率是极小的）。

补充：这道题可能会有选手在数据类型上选择不当，导致一些情况会出现溢出。

[表达式求值]

经过上面的叙述，难点落在了表达式求值上，在这里我们介绍一下最一般、最简单的方法，栈运算。

用栈实现表达式求值的方法：

首先，我们要给每一个符号一个优先级：

符号	+ -	* /	^	(	)
栈内级别	2	4	6	0	8
栈外级别	1	3	5	8	0

可以看到，优先级高的符号先算。为了方便起见，我们定义特殊符号#，它级别最低（赋-1）

先将它置栈底，然后依次读入每个字符，如果是数字则入数栈。如果是符号，就与栈顶符号比较优先级。如果相等，则退栈，读下一字符。如果栈外大，则入栈。如果栈内大，则取栈顶元素与数栈最顶 2 元素运算，结果入数栈。这个符号继续处理（再与栈顶比较）。直到读到最后符号 #使栈底#出栈时。数栈顶即为表达式结果。

由此，本题已经变得清晰了，剩下的就是具体将我们的表述变成代码。

[文档信息]

文章总排版、总发布：Clf-梁烨 In 麓山国际[长郡集团]

第一题分析：Clf-梁烨 In 麓山国际[长郡集团]

第二题分析：标准算法-Eastorn-李向东 In 麓山国际[长郡集团]

特殊算法-Whb-吴海波 In 麓山国际[长郡集团]

第三题分析：Mask@Cai-蔡湘 In 麓山国际[长郡集团]

第四题分析：Lc-刘诚 In 麓山国际[长郡集团]

鸣谢：我们的教练老师：王灿、周祖松，以及 长郡中学所有老师。

注：第一次写解题报告，时间仓促，若还有叙述不清的地方，还请各位在 OIBH.KuYe.Cn 的留言板上提出。

解题报告并非源程，这样更有助于思维的锻炼，第一题中的一些代码均是在 Word 中编写，如有语法错误，还请谅解。谢谢！