# 本章主要为中等DP,数据结构

# 译题 - prefix

## 描述

在生物学中，一些生物的结构是用包含其要素的大写字母序列来表示的。生物学家对于把长的序列分解成较短的（称之为元素的）序列很感兴趣。

如果一个集合 P 中的元素可以通过串联（允许重复；串联，相当于 Pascal 中的 “+” 运算符）组成一个序列 S ，那么我们认为序列 S 可以分解为 P 中的元素。并不是所有的元素都必须出现。举个例子，序列 ABABACABAAB 可以分解为下面集合中的元素：

{A, AB, BA, CA, BBC}

序列 S 的前面 K 个字符称作 S 中长度为 K 的前缀。设计一个程序，输入一个元素集合以及一个大写字母序列，计算这个序列最长的前缀的长度。

## 格式

PROGRAM NAME: prefix

INPUT FORMAT

输入数据的开头包括 1..200 个元素（长度为 1..10 ）组成的集合，用连续的以空格分开的字符串表示。字母全部是大写，数据可能不止一行。元素集合结束的标志是一个只包含一个 “.” 的行。集合中的元素没有重复。接着是大写字母序列 S ，长度为 1..200,000 ，用一行或者多行的字符串来表示，每行不超过 76 个字符。换行符并不是序列 S 的一部分。

OUTPUT FORMAT

只有一行，输出一个整数，表示 S 能够分解成 P 中元素的最长前缀的长度。

## SAMPLE INPUT (file prefix.in)

```
A AB BA CA BBC
.
ABABACABAABC
```

SAMPLE OUTPUT (file prefix.out)

11

# 题解 - prefix

## 分析

### 动态规划

设dp[i]表示主串S中从i开始的最长的可组成的前缀,dp[1]就是所求,状态转移方程:

dp[i]=max{dp[j]+j-i}，其中 i 到 j-1 的字串是 primitive。

用Hash表判断primitive,将每个primitive转成10位27进制数,再mod一个大质数。在判断primitive的时候只需把i到j-1的字串的Hash值算出来就可以了。

因为primitive最多只有10位，所以时间复杂度是O(10n)，其中n是主串的长度。

其实朴素的字符串匹配就可以搞定了

另一个转移方程：f[i]表示s前i个字符能否取得 f[i]:=(f[i-length(p[j])]) and (copy(s,i-lenght[j]+1,ll[j])=p[j]);

i:=length(s); j:=P集合中元素个数;

初始化： fillchar(f,sizeof(f),false); f[0]:=true;

最后取得最大的值为真的f[ans]就行了（当i-ans>10时即可break了）

### Trie

# 译题 - nocows

Cow Pedigrees 奶牛家谱

## 描述

农民约翰准备购买一群新奶牛。 在这个新的奶牛群中，每一个母亲奶牛都生两小奶牛。这些奶牛间的关系可以用二叉树来表示。这些二叉树总共有 N 个节点(3 <= N < 200)。这些二叉树有如下性质：

每一个节点的度是 0 或 2。度是这个节点的孩子的数目。

树的高度等于 K(1 < K < 100)。高度是从根到任何叶子的最长的路径上的节点的数目；叶子是指没有孩子的节点。

有多少不同的家谱结构？如果一个家谱的树结构不同于另一个的，那么这两个家谱就是不同的。输出可能的家谱树的个数除以 9901 的余数。

# 格式

PROGRAM NAME: nocows

INPUT FORMAT (file nocows.in)

第 1 行：两个空格分开的整数，N 和 K。

OUTPUT FORMAT (file nocows.out)

第 1 行：一个整数，表示可能的家谱树的个数除以 9901 的余数。

## SAMPLE INPUT

5 3

## SAMPLE OUTPUT

2

## OUTPUT DETAILS

有 5 个节点，高为 3 的两个不同的家谱：

# 题解 - nocows

## 分析

（USACO 题解， 译 by 巨菜逆铭）

这是一个 DP 问题。我们所关心的树的性质是深度和节点数，所以我们可以做这样一张表：table[i][j]表示深度为 i、节点数为 j 的树的个数。根据给定的约束条件，j 必须为奇数。你如何构造一棵树呢？当然是由更小的树来构造了。一棵深度为 i、节点数为 j 的树可以由两个子树以及一个根结点构造而成。当 i、j 已经选定时，我们选择左子树的节点数 k。这样我们也就知道了右子树的节点数，即 j-k-1。至于深度，至少要有一棵子树的深度为 i-1 才能使构造出的新树深度为 i。有三种可能的情况：左子树深度为 i-1 ，右子树深度小于 i-1；右子树深度为 i-1，左子树深度小于 i-1；左右子树深度都为 i-1。事实上，当我们在构造一棵深度为 i 的树时，我们只关心使用的子树深度是否为 i-1 或更小。因此，我们使用另一个数组 smalltrees[i-2][j]记录所有深度小于 i-1 的树，而不仅仅是深度为 i-2 的树。知道了上面的这些，我们就可以用以下三种可能的方法来建树了：

```
table[i][j] += smalltrees[i-2][k]*table[i-1][j-1-k];
  // 左子树深度小于 i-1，右子树深度为 i-1
table[i][j] += table[i-1][k]*smalltrees[i-2][j-1-k];
  // 左子树深度为 i-1，右子树深度小于 i-1
table[i][j] += table[i-1][k]*table[i-1][j-1-k];
  // 左右子树深度都为 i-1
```

另外，如果左子树更小，我们可以对它进行两次计数，因为可以通过交换左右子树来得到不同的树。总运行时间为 O(K*N^2)，且有不错的常数因子。（USACO 标程见源码-c 部分）

首先明确一下题目的意思：用 N 个点组成一棵深度为 K 的二叉树，求一共有几种方法？设 dp[i,j]表示用 i 个点组成深度最多为 j 的二叉树的方法数，则：

$dp[i,j]=\Sigma(dp[k,j-1]\times dp[i-1-k,j-1])(k\in 1..i-2)$
边界条件：dp[1,i]=1

我们要求的是深度恰好为 K 的方法数 S，易知 $S=dp[n,k]-dp[n,k-1]$。

# 译题 - zerosum

## 描述

请考虑一个由 1 到 N（N=3, 4, 5 ... 9）的数字组成的递增数列：1 2 3 ... N。 现在请在数列中插入“+”表示加，或者“-”表示减，抑或是“ ”表示空白(例如 1-2 3 就等于 1-23)，来将每一对数字组合在一起（请不在第一个数字前插入符号）。 计算该表达式的结果并注意你是否得到了和为零。 请你写一个程序找出所有产生和为零的长度为 N 的数列。

# 格式

PROGRAM NAME: zerosum

INPUT FORMAT

单独的一行表示整数 N (3 <= N <= 9)。

OUTPUT FORMAT

按照 ASCII 码的顺序，输出所有在每对数字间插入“+”, “-”, 或 “ ”后能得到和为零的数列。

## SAMPLE INPUT (file zerosum.in)

7

## SAMPLE OUTPUT (file zerosum.out)

1+2-3+4-5-6+7
1+2-3-4+5+6-7
1-2 3+4+5+6+7
1-2 3-4 5+6 7
1-2+3+4-5+6-7
1-2-3-4-5+6+7

# 题解 - zerosum

## 分析

DFS

每层只有 3 个状态，层数最多只有 8 层，不会超时。

枚举时按照’’,’+’,’-’的顺序添加当前空格处的运算符，保证输出的顺序正确。用 sum 记录当前加和，用 last 记录当前的连续数，就是之前添’’连成的数。

’’：sum=sum-last+last*10+s+1(last>0)；sum=sum-last+last*10-s-1(last<0)；
last=last*10+s+1(last>0)； last=last*10-s-1(last<0)
’+’： sum=sum+s+1； last=s+1
’-’： sum=sum-s-1； last=-s-1

# 译题 - money

Money Systems 货币系统

译 by timgreen

## 描述

母牛们不但创建了他们自己的政府而且选择了建立了自己的货币系统。由于他们特殊的思考方式，他们对货币的数值感到好奇。

传统地，一个货币系统是由 1,5,10,20 或 25,50, 和 100 的单位面值组成的。

母牛想知道有多少种不同的方法来用货币系统中的货币来构造一个确定的数值。

举例来说, 使用一个货币系统 {1,2,5,10,...}产生 18 单位面值的一些可能的方法是:18x1, 9x2, 8x2+2x1, 3x5+2+1,等等其它。 写一个程序来计算有多少种方法用给定的货币系统来构造一定数量的面值。保证总数将会适合 long long (C/C++) 和 Int64 (Free Pascal)。

## 格式

PROGRAM NAME: money

INPUT FORMAT:

(file money.in)

货币系统中货币的种类数目是 V (1<= V<=25)。要构造的数量钱是 N (1<= N<=10,000)。

第 1 行: 二整数， V 和 N

第 2 行： 可用的货币的面值 。

OUTPUT FORMAT:

(file money.out)

单独的一行包含那个可能的构造的方案数。

## SAMPLE INPUT

3 10
1 2 5

## SAMPLE OUTPUT

10

# 题解 - money

## 分析

背包问题

设 dp[i,j]表示前 i 种货币构成 j 的方法数，用 cc 记录货币的面值，状态转移方程为：

dp[i,j]=dp[i-1,j]　　不用第 i 种货币

　　+dp[i,j-cc[i]]　　用第 i 种货币，j>=cc[i]

时间复杂度是 O(VN)的，如果把面值相同的货币记做一种，可以更快一些。

# 译题 - concom

Controlling Companies

控制公司

译 by TinyTony {刚开始真的没看懂，有点紊乱，就试着修改了一下以便于理解}

## 题目

有些公司是其他公司的部分拥有者，因为他们获得了其他公司发行的股票的一部分。例如，福特公司拥有马自达公司 12%的股票。据说，如果至少满足了以下三个条件之一，公司 A 就可以控制公司 B 了：

（I）、公司 A = 公司 B。

（II）、公司 A 拥有大于 50%的公司 B 的股票。

（III）、公司 A 控制 K(K >= 1)个公司，记为 C1, ..., CK，每个公司 Ci 拥有 xi%的公司 B 的股票，并且 x1+ .... + xK > 50%。 给你一个表，每行包括三个数（i,j,p)；表明公司 i 享有公司 j 的 p%的股票。计算所有的数对（h，s），表明公司 h 控制公司 s。至多有 100 个公司。

写一个程序读入三对数（i，j，p），i，j 和 p 是都在范围(1..100)的正整数，并且找出所有的数对（h，s），使得公司 h 控制公司 s。

# INPUT FORMAT

第一行： N，表明接下来三对数的数量。{即（i,j,p）的数量}

第二行到第 N+1 行： 每行三个整数作为一个三对数（i，j，p），如上文所述。{表示公司 i 拥有公司 j p%的股份}

# SAMPLE INPUT (file concom.in)

```
3

1 2 80

2 3 80

3 1 20
```

# OUTPUT FORMAT

输出零个或更多个的控制其他公司的公司。每行包括两个整数 A、B,表示 A 公司控制了 B 公司。将输出的数对以升序排列。

请不要输出控制自己的公司。

# SAMPLE OUTPUT (file concom.out)

1 2

1 3

2 3

# 题解 - concom

## 分析

（USACO 题解 译 by 逆铭）
这里使用的解题方法如下。我们记录哪些公司控制哪些公司，且当每次我们得知某公司控制了某公司百分之多少的股份，就更新我们的信息。

数组“owns”记录了 i 公司拥有 j 公司的多少股份，包括直接控制以及间接控制。数组"controls"记录哪些公司被哪些公司控制。

DFS

con[i,j]记录 i 控制 j 的股份

对于每个公司 i，搜索 i 直接控股的所有公司，用 cx[j]记录公司 i 控制了公司 j 的股份（直接和间接），将股份加到 cx 上，如果超过 50 就递归搜索公司 j，每个公司至多只递归一次，所以用 vis 判重。最后搜索所有 cx 超过 50 的 j，那么 i 控制 j。