算法设计与分析

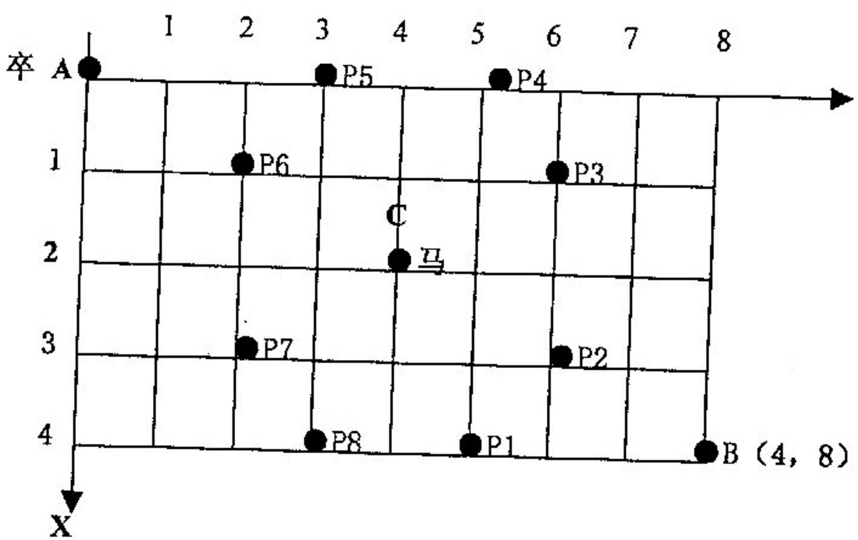
动态规划2

例题六. 马拦过河卒

实例

[问题描述]:

如图，**A** 点有一个过河卒，需要走到目标 **B** 点。卒行走规则：可以向下、或者向右。同时在棋盘上的任一点有一个对方的马（如上图的**C**点），该马所在的点和所有跳跃一步可达的点称为对方马的控制点。例如上图 **C** 点上的马可以控制 **9** 个点（图中的**P1**，**P2** … **P8** 和 **C**）。卒不能通过对方马的控制点。



棋盘用坐标表示，A 点（0，0）、B 点（n,m）(n,m 为不超过 20 的整数，并由键盘输入)，同样马的位置坐标是需要给出的（约定: C<>A，同时C<>B）。现在要求你计算出卒从 A 点能够到达 B 点的路径的条数。

[输入]:

键盘输入

B点的坐标（n,m）以及对方马的坐标（X,Y）

[输出]:

屏幕输出

一个整数（路径的条数）。

[输入输出样例]:

输入：

6 6 3 2

输出：

17

分析：阶段：棋盘上的每个可走的点；

每个阶段的求解；

步骤1：用F（X,Y）表示到棋盘上每个阶段(X,Y)的路径条数；

步骤2：状态转移方程：

$$F(X,Y) = F(X-1,Y) + F(X,Y-1)$$

其中，$F(0,0) = 1$

$$F(0,Y) = 1$$

$$F(X,0) = 1$$

马的影响怎么体现？

步骤3：以自底向上的方法来计算最优解

例题七：数字三角形问题

1． 问题描述

设有一个三角形的数塔，顶点结点称为根结点，每个结点有一个整数数值。从顶点出发，可以向左走，也可以向右走。如图10—1所示。

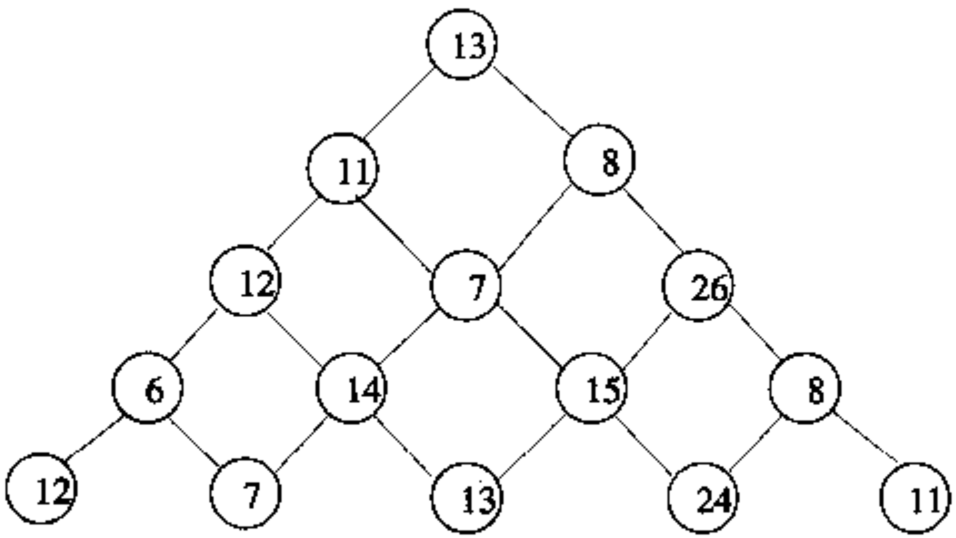


图 10 – 1

问题：当三角形数塔给出之后，找出一条从第一层到达底层的路径，使路径的值最小。若这样的路径存在多条，任意给出一条即可。

步骤1

二维数组 $\boldsymbol{D(x,y)}$描述问题，$\boldsymbol{D(x,y)}$表示从顶层到达第$\boldsymbol{x}$层第$\boldsymbol{y}$个位置的最小路径得分。

阶段分析：$D(1,1)$ =13

到第x层的第y个位置有两种可能，要么走右分支得到，要么走左分支得到。

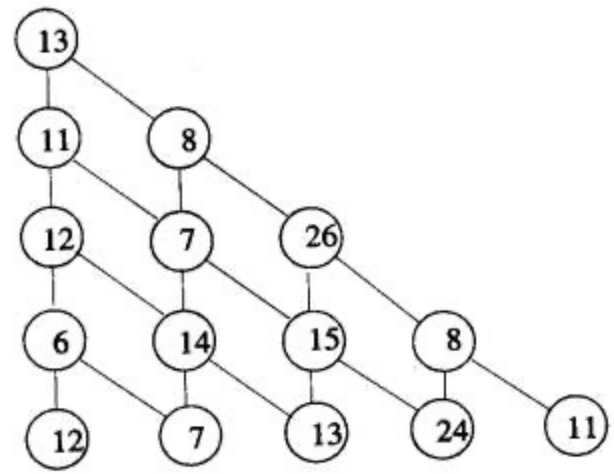


图 10 - 4

步骤2：状态转移方程

- $D(x,y)$＝$\min\{D(x-1,y)$，$D(x-1,y-1\}+a(x,y)$
- D(1, 1)＝a(1, 1)

例题八：最长公共子序列

一个应用背景：基因序列比对。

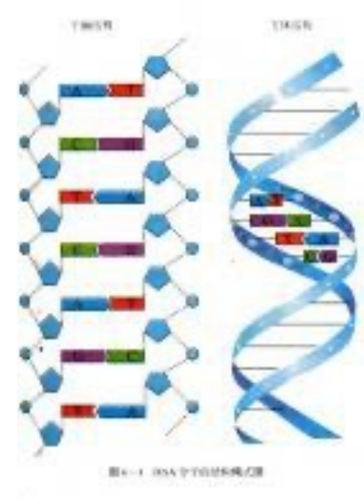
DNA（Deoxyribonucleic Acid，脱氧核糖核酸）是染色体的主要组成成分。DNA又是由腺嘌呤(adenine)、鸟嘌呤(guanine)、胞嘧啶(cytosine)、胸腺嘧啶(thymine)等四种碱基分子（canonical bases）组成。用它们单词的首字母A、C、G、T来代表这四种碱基，

这样一条DNA上碱基分子的排列被表示为有穷字符集{A,C,G,T}上的一个串进行表示。

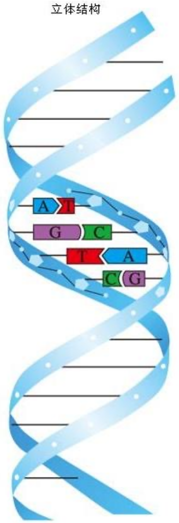
如：两个有机体的DNA分别为

S_1=ACCGGTCGAGTGCGCGGAAGCCGGCCGAA

S_2=GTCGTTCGGAATGCCGTTGCTCTGTAAA



可以通过比较两个有机体的DNA来确定这两个有机体有多么“相似”。这在生物学上叫做“基因序列比对”，而用计算机的话讲就是把比较两个DNA相似性的操作看作是对两个由A、C、G、T组成的字符串的比较。



- 度量DNA的相似性：
  - 如果一个DNA螺旋是另一个DNA螺旋的子串，就说这两个DNA（串）相似。
  - 当两个DNA螺旋互不为对方子串的时候，怎么度量呢？

    方法一：如果将其中一个转换成另一个所需改变的工作量小，则可称其相似（参见 *Edit distance 15-5*）。

方法二：在S_1和S_2中找出第三个存在S_3，使得S_3中的基以同样的先后顺序出现在S_1和S_2中，但不一定连续。

然后视S_3的长度，确定S_1和S_2的相似度。S_3越长，S_1和S_2的相似度越大，反之越小。

- 如上面的两个DNA串中，最长的公共存在是

S_3=GTCGTCGGAAGCCGGCCGAA。

S_1=ACCGGTCGAGTGCGCGGAAGCCGGCCGAA

S_2=GTCGTTCGGAATGCCGTTGCTCTGTAAA

怎么找最长的公共存在——两个字符串的最长公共非连续子串，称为最长公共子序列。

1、最长公共子序列的定义

(1) 子序列

给定两个序列X=<$x_1, x_2, ..., x_n$>和序列Z=<$z_1, z_2, ..., z_k$>，若存在X的一个严格递增下标序列<$i_1, i_2, ..., i_k$>，使得对所有j=1, 2, . . . , k，有$x_{i_j} = z_j$，则称Z是X的子序列。

如：Z=<B, C, D, B>是X=<A, B, C, B, D, A, B>的一个子序列，

相应的下标序列为<2, 3, 5, 7>。

2）公共子序列

对给定的两个序列X和Y，若序列Z既是X的的子序列，也是Y的子序列，则称Z是X和Y的公共子序列。

如，X=<A, B, C, B, D, A, B>, Y=<B, D, C, A, B, A>，则序列<B, C, A>是X和Y的一个公共子序列。

3）最长公共子序列

两个序列的长度最大的公共子序列称为它们的最长公共子序列。

如，<B, C, A>是上面X和Y的一个公共子序列，但不是X和Y的最长公共子序列。最长公共子序列是<B, C, B, A>。

怎么求最长公共子序列？

2、最长公共子序列问题（Longest-Common-Subsequence,LCS）

——求（两个）序列的最长公共子序列

前缀：给定一个序列$X=\langle x_1, x_2, ..., x_m\rangle$，对于$i=0, 1, ..., m$，定义X的第i个前缀为$X_i=\langle x_1, x_2, ..., x_i\rangle$，即前i个元素构成的子序列。

如，$X=\langle A, B, C, B, D, A, B\rangle$，则

$X_4=\langle A, B, C, B\rangle$。

$X_0=\Phi$。

1）LCS问题的最优子结构性

设有序列$X=\langle x_1, x_2, ..., x_m\rangle$和$Y=\langle y_1, y_2, ..., y_n\rangle$，并

设序列$Z=\langle z_1, z_2, ..., z_k\rangle$为X和Y的任意一个LCS。

（1）若$x_m = y_n$，则$z_k=x_m=y_n$，且Z_{k-1}是X_{m-1}和Y_{n-1}的一个LCS。

（2）若$x_m \neq y_n$，则$z_k \neq x_m$蕴含Z是X_{m-1}和Y的一个LCS。

（3）若$x_m \neq y_n$，则$z_k \neq y_n$蕴含Z是X和Y_{n-1}的一个LCS。

子问题的定义：从“X_m和Y_n的LCS”到“X_{m-1}和Y_{n-1}的LCS”、“X_{m-1}和Y_n的LCS”、“X_m和Y_{n-1}的LCS”

2）递推关系式

记，c[i,j]为前缀序列X_i和Y_j的一个LCS的长度。则有

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

含义：

1）若i=0或j=0，即其中一个序列的长度为零，则二者的LCS的长度为0，LCS=Φ；

2）若$x_i=y_j$，则X_i和Y_j的LCS是在X_{i-1}和Y_{j-1}的LCS之后附加将x_i （也即y_j）得到的，所以

c[i,j]=c[i-1,j-1]+1；

3）若$x_i\neq y_j$，则X_i和Y_j的LCS的最后一个字符不会是x_i或y_j(不可能同时等于两者，或与两者都不同)，此时该LCS应等于X_{i-1}和Y_j的LCS与X_i和Y_{j-1}的LCS之中的长者。所以

c[i,j]=max(c[i-1,j],c[i,j-1])；

3）LCS的求解

X_m和Y_n的LCS是基于X_{m-1}和Y_{n-1}的LCS、或X_{m-1}和Y_n的LCS、或X_m和Y_{n-1}的LCS求解的。

下述过程**LCS-LENGTH(X,Y)**求序列$X=\langle x_1, x_2, ..., x_m\rangle$和$Y=\langle y_1, y_2, ..., y_n\rangle$的LCS的长度，表**c[1..m,1..n]**中包含每一阶段的LCS长度，c[m,n]等于X和Y的LCS的长度。

同时，还设置了一个表**b[1..m,1..n]**，记录当前c[i,j]的计值情况，以此来构造该LCS。

LCS-LENGTH(X, Y)

```
m = X.length
n = Y.length
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i-1, j-1] + 1
            b[i, j] = "↖"
        elseif c[i-1, j] ≥ c[i, j-1]
            c[i, j] = c[i-1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j-1]
            b[i, j] = "←"
return c and b
```

j		0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

LCS-LENGTH的时间复杂度为O(mn)

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	←1	↖ 1
2	B	0	↖ 1	←1	←1	↑ 1	↖ 2	←2
3	C	0	↑ 1	↑ 1	↖ 2	←2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	←3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

说明：

1）第i行和第j列中的方块包含了c[i, j]的值以及b[i, j]记录的箭头。

2）对于i, j>0，项c[i, j]仅依赖于是否有 $x_i=y_j$ 及项c[i-1, j]、c[i, j-1]、c[i-1, j-1]的值。

3）为了重构一个LCS，从右下角开始跟踪b[i, j]箭头即可，即如图所示中的蓝色方块给出的轨迹。

4）图中，c[7, 6]=4，

LCS(X, Y)=<B, C, B, A>

例，下图给出了在X=〈A, B, C, B, D, A, B〉和Y=〈B, D, C, A, B, A〉上运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

j		0	1	2	3	4	5	6
i	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	←1	↖ 1
2	B	0	↖ 1	←1	←1	↑ 1	↖ 2	←2
3	C	0	↑ 1	↑ 1	↖ 2	←2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	←3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 1, j = 1　　$x_1 \neq y_1$

c[1,1] = max{c[1,0], c[0,1]}
= max{0,0}
= 0

b[1,1] = '↑'

elseif $c[i-1,j] \geq c[i,j-1]$
$c[i,j] = c[i-1,j]$
$b[i,j] = $ "↑"

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上

运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 1, j = 2　　$x_1 \neq y_2$

c[1,2] = max{c[1,1], c[0,2]}
= max{0,0}
= 0

b[1,2] = '↑'

elseif $c[i-1,j] \geq c[i,j-1]$
$c[i,j] = c[i-1,j]$
$b[i,j] = $ "↑"

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 1, j = 3 $x_1 \neq y_3$

c[1,3] = max{c[1,2], c[0,3]}
= max{0,0}
= 0

b[1,3] = ‘↑’

elseif $c[i-1,j] \geq c[i,j-1]$
$c[i,j] = c[i-1,j]$
$b[i,j] =$ “↑”

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

i \ j		0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 1, j = 4 $x_1 = y_4$

c[1,4] = c[0,3] + 1
= 0 + 1
= 1

b[1,4] = '↖'

if $x_i == y_j$
$c[i,j] = c[i-1,j-1]+1$
$b[i,j] = $ "↖"

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

$$c[i,j]=\begin{cases}0 & 如果i=0或j=0\\ c[i-1,j-1]+1 & 如果i,j>0且x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & 如果i,j>0且x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 1, j = 5　　$x_1 \neq y_5$

c[1,5] = max{c[1,4], c[0,5]}
= max{1,0}
= 0

b[1,5] = '←'

$$\textbf{else } c[i,j] = c[i,j-1]$$
$$b[i,j] = \text{“}\leftarrow\text{”}$$

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

$$c[i,j]=\begin{cases}0 & \text{如果} i=0 \text{或} j=0\\ c[i-1,j-1]+1 & \text{如果} i,j>0 \text{且} x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果} i,j>0 \text{且} x_i\neq y_j\end{cases}$$

$c[0,j] = 0$

$c[i,0] = 0$

$i = 1, j = 6$ $x_1 = y_6$

$$\begin{aligned} c[1,6] &= c[0,5] + 1\\ &= 0 + 1\\ &= 1 \end{aligned}$$

$b[1,6] =$ '↖'

if $x_i == y_j$
$c[i,j] = c[i-1, j-1] + 1$
$b[i,j] =$ "↖"

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	←1	↖ 1
2	B	0	↖ 1	←1	←1	↑ 1	↖ 2	←2
3	C	0	↑ 1	↑ 1	↖ 2	←2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	←3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

$$c[i,j]=\begin{cases}0 & \text{如果}i=0\text{或}j=0\\ c[i-1,j-1]+1 & \text{如果}i,j>0\text{且}x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果}i,j>0\text{且}x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 2, j = 1 $x_2 = y_2$

c[2,1] = c[1,0] + 1
= 0 +1
= 1

b[2,1] = ‘↖’

$$\textbf{if } x_i == y_j$$
$$c[i,j] = c[i-1,j-1]+1$$
$$b[i,j] = \text{“}\nwarrow\text{”}$$

例，下图给出了在X=<A, B, C, B, D, A, B>和Y=<B, D, C, A, B, A>上运行LCS-LENGTH计算出的表。

	j	0	1	2	3	4	5	6
i	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

$$c[i,j]=\begin{cases}0 & \text{如果} i=0 \text{或} j=0\\ c[i-1,j-1]+1 & \text{如果} i,j>0 \text{且} x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{如果} i,j>0 \text{且} x_i\neq y_j\end{cases}$$

c[0,j] = 0

c[i,0] = 0

i = 2, j = 5　　$x_2 = y_5$

c[2,5] = c[1,4] + 1
　　= 1+1
　　= 2

b[2,5] = '↖'

if $x_i == y_j$
　$c[i,j] = c[i-1,j-1]+1$
　$b[i,j] = $ "↖"

构造一个LCS

表b用来构造序列X=<x_1, x_2, . . . , x_m>和Y=<y_1, y_2, . . . , y_n>的一个LCS：

反序，从b[m, n]处开始，沿箭头在表格中向上跟踪。每当在表项b[i, j]中：

- 遇到一个“↖”时，意味着x_i=y_j是LCS的一个元素，下一步继续在b[i-1, j-1]中寻找上一个元素；
- 遇到“←”时，下一步到b[i, j-1]中寻找上一个元素；
- 遇到“↑”时，下一步到b[i-1, j]中寻找上一个元素。

过程PRINT-LCS按照上述规则输出X和Y的LCS

```
PRINT-LCS(b, X, i, j)
if i == 0 or j == 0
    return
if b[i, j] == "↖"
    PRINT-LCS(b, X, i − 1, j − 1)
    print x_i
elseif b[i, j] == "↑"
    PRINT-LCS(b, X, i − 1, j)
else PRINT-LCS(b, X, i, j − 1)
```

由于每一次循环使i或j减1，最终m=0，n=0，算法结束，所以PRINT-LCS的时间复杂度为O(m+n)

```
PRINT-LCS(b, X, i, j)
if i == 0 or j == 0
    return
if b[i, j] == "↖"
    PRINT-LCS(b, X, i − 1, j − 1)
    print x_i
elseif b[i, j] == "↑"
    PRINT-LCS(b, X, i − 1, j)
else PRINT-LCS(b, X, i, j − 1)
```

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

PRINT-LCS(b, X, 7, 6)

PRINT-LCS(b, X, 6, 6)   print A

PRINT-LCS(b, X, 5, 5)

PRINT-LCS(b, X, 4, 5)   print B

PRINT-LCS(b, X, 3, 4)

PRINT-LCS(b, X, 3, 3)   print C

PRINT-LCS(b, X, 2, 2)

PRINT-LCS(b, X, 2, 1)   print B

PRINT-LCS(b, X, 1, 0)   结束

例题9：0-1背包问题

- 小偷有一个可承受W的背包
- 有n件物品: 第i个物品价值v_i 且重w_i
- 目标:
  - 找到x_i使得对于所有的$x_i = \{0, 1\}, i = 1, 2, .., n$

    $\sum w_i x_i \leq W$ 并且$\sum x_i v_i$ 最大

部分背包问题

最优子结构

- 考虑最多重W的物品且价值最高
- 如果我们把j物品从背包中拿出来

$\Rightarrow$ 剩下的装载一定是取自n-1个物品使得不超过载重量$W - w_j$并且所装物品价值最高的装载

最优子结构

首先证明0/1背包问题满足最优性原理。

设(x_1, x_2, …, x_n)是所给0/1背包问题的一个最优解，则(x_2, …, x_n)是下面一个子问题的最优解：

$$\begin{cases} \sum_{i=2}^{n} w_i x_i \le C - w_1 x_1 \\ x_i \in \{0,1\} \quad (2 \le i \le n) \end{cases}$$

如若不然，设(y_2, …, y_n)是上述子问题的一个最优解，则 $\sum_{i=2}^{n} v_i y_i > \sum_{i=2}^{n} v_i x_i$ ，且 $w_1 x_1 + \sum_{i=2}^{n} w_i y_i \le C$ 。因此，

$$v_1 x_1 + \sum_{i=2}^{n} v_i y_i > v_1 x_1 + \sum_{i=2}^{n} v_i x_i = \sum_{i=1}^{n} v_i x_i$$

这说明(x_1, y_2, …, y_n)是所给0/1背包问题比(x_1, x_2, …, x_n)更优的解，从而导致矛盾。

0-1背包问题的动态规划

步骤1

$P(i, w)$ –考虑前i件物品所能获得的最高价值,其中w是背包的承受力

阶段分析：

对于每一个物品i，都有两种情况需要考虑

第1种情况:物品i的重量w_i<=w，小偷对物品i可拿或者不拿

$$P[i,w] = \max\{P[i-1, w], P[i-1,w-w_i] + v_i\}$$

第2种情况:物品i的重量w_i>w,即小偷不拿物品i

$$P(i, w) = P(i - 1, w)$$

步骤2

$$P(i, w) = \begin{cases} P(i-1,w) & \text{当wi> w（不够装不装）} \\ \max\begin{cases} P(i-1,w) & \text{够装但不装} \\ p(i-1,w-wi)+vi & \text{够装而且装} \end{cases} \end{cases}$$

$$F(i,w) = \max\{F(i-1,w-kw[i]) + kv[i]\},\ k = 0,1$$

Knapsack (S,W)

for $w \leftarrow 0$ **to** $w_1 - 1$ **do** $P[1, w] \leftarrow 0$;
for $w \leftarrow w_1$ **to** W **do** $P[1, w] \leftarrow v_1$;
for $i \leftarrow 2$ **to** n **do**
    for $w \leftarrow 0$ **to** W **do**
        if $w_i > w$ **then**
            $P[i,w] \leftarrow P[i-1, w]$;
        else
            $P[i,w] \leftarrow \max\{P[i-1, w], P[i-1,w-w_i] + v_i\}$

运行时间: $\Theta(nW)$

拓展1：装箱问题 （vijos 1133）

有一个箱子容量为v(正整数，o≤v≤20000)，同时有n个物品(o≤n≤30)，每个物品有一个体积 (正整数)。要求从 n 个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

输入格式 Input Format

第一行，一个整数，表示箱子容量；
第二行，一个整数，表示有n个物品；
接下来n行，分别表示这n个物品的各自体积。

输出格式 Output Format

一个整数，表示箱子剩余空间。

样例输入 Sample Input

24
6
8
3
12
7
9
7

样例输出 Sample Output

0

拓展2：采药（vijos1104)

辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

如果你是辰辰，你能完成这个任务吗？

输入格式 Input Format

输入的第一行有两个整数T（1 <= T <= 1000）和M（1 <= M <= 100），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到100之间（包括1和100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

输出格式 Output Format

输出包括一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

样例输入 Sample Input

```
70 3
71 100
69 1
1 2
```

样例输出 Sample Output

```
3
```

拓展3：开心的金明 （vijos　1317）

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N 元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N 元。于是，他把每件物品规定了一个重要度，分为5 等：用整数1~5 表示，第5 等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N 元（可以等于N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。设第j 件物品的价格为v[j]，重要度为w[j]，共选中了k 件物品，编号依次为j1...jk，则所求的总和为：v[j1]*w[j1]+..+v[jk]*w[jk]请你帮助金明设计一个满足要求的购物单.

输入格式 Input Format

输入的第1 行，为两个正整数，用一个空格隔开：
N m
（其中N（<30000）表示总钱数，m(<25)为希望购买物品的个数。）
从第2 行到第m+1 行，第j 行给出了编号为j-1
的物品的基本数据，每行有2 个非负整数
v p
（其中v 表示该物品的价格（v≤10000），p 表示该物品的重要度（1~5））

输出格式 Output Format

输出只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<100000000）

样例输入 Sample Input

1000 5
800 2
400 5
300 5
400 3
200 2

样例输出 Sample Output

3900

例题10：完全背包

描述：有N种物品和一个体积为W的背包，每种物品都有无限件可用。第i种物品的体积是w[i]，价值是v[i]。求解将哪些物品装入背包可使这些物品的体积总和不超过背包容量，且价值总和最大。

$F(i,w)$表示前i种物品放入容量为w的背包中的最大价值，

$$F(i,w) = \max\{F(i-1, w-kw[i]) + kv[i]\}，kw[i] \le w$$

例题：洛谷P1616，疯狂采药

可转化为01背包问题

例题11：多重背包

描述：有N种物品和一个容量为W的背包。第i种物品最多有p[i]件可用，每件体积是w[i]，价值是v[i]。求解将哪些物品装入背包可使这些物品的体积总和不超过背包容量，且价值总和最大。

$F(i,w)$表示前i种物品放入容量为w的背包中的最大价值，

$F(i,w)=\max\{F(i-1,w-kw[i])+kv[i]\}$，$0\le k\le p[i]$

例题：Luogu P1776 宝物筛选

可转化为01背包问题

例题12：混合背包

描述：01和完全背包混合，或者再加上多重背包

全部转化为01背包问题

例题：[Luogu P1833 樱花]

例题13：分组背包

描述：有N件物品和一个容量为W的背包。第i件物品的体积是w[i]，价值是v[i] 。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的体积总和不超过背包容量，且价值总和最大。

$F(i,w)$表示前i组物品放入容量为w的背包中的最大价值，

$$F(i,w)=\max(F[i-1][w],f[i-1][w-w[k]]+v[k]\mid 物品k属于组i)$$

例题Luogu 1757 通天之分组背包

一组最多选一件

例题14：有依赖的背包

金明的预算方案

描述：金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N元钱就行”。今天一早，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：

主件	附件
电脑	打印机，扫描仪
书柜	图书
书桌	台灯，文具
工作椅	无

一组可以选多件

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有0个、1个或2个附件。附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是10元的整数倍）。他希望在不超过N元（可以等于N元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为j1，j2，……，jk，则所求的总和为：v[j1]*w[j1]+v[j2]*w[j2]+ …+v[jk]*w[jk]。（其中*为乘号）请你帮助金明设计一个满足要求的购物单。

输入格式 Input Format

输入文件的第1行，为两个正整数，用一个空格隔开：
N m
其中N（<32000）表示总钱数，m（<60）为希望购买物品的个数。）
从第2行到第m+1行，第j行给出了编号为j-1的物品的基本数据，每行有3个非负整数
v p q
（其中v表示该物品的价格（v<10000），p表示该物品的重要度（1~5），q表示该物品是主件还是附件。如果q=0，表示该物品为主件，如果q>0，表示该物品为附件，q是所属主件的编号）

输出格式 Output Format

输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值
（<200000）。

样例输入 Sample Input

```
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
```

样例输出 Sample Output

```
2200
```

例题15：二维费用背包

描述：对于每件物品，具有两种不同的费用；选择这件物品必须同时付出这两种代价；对于每种代价都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。设第i件物品物品的价值为v[i]，所需的两种代价分别为w[i] 和g[i]。两种代价可付出的最大值（两种背包容量）分别为V和T。

设$F(i,w,g)$表示前i件物品付出两种代价分别为w和g时可获得的最大价值。

$$F(i,w,g)=max(f(i-1,w,g),f(i-1,w-w[i],w-g[i])+v[i])$$

例题Luogu 1507 NASA的食物计划

继续题目类型整理