**HDU2602**

**题目描述（HDU2602）：**有位骨头收藏家喜欢收集各种各样的骨头，不同的骨头有不同的体积和价值。这个收藏家有一个体积为V的背包，请计算他可以收藏的最大价值。

**输入：**第1行包含一个整数T，表示测试用例的数量。每个测试用例都包含3行，第1行包含两个整数N、V（N≤1000，V≤1000），分别表示骨头的数量和背包的体积；第2行包含N个整数，表示每个骨头的价值；第3行包含N个整数，表示每个骨头的体积。

**输出：**对每个测试用例，都单行输出可以得到的最大价值（该数小于231）。

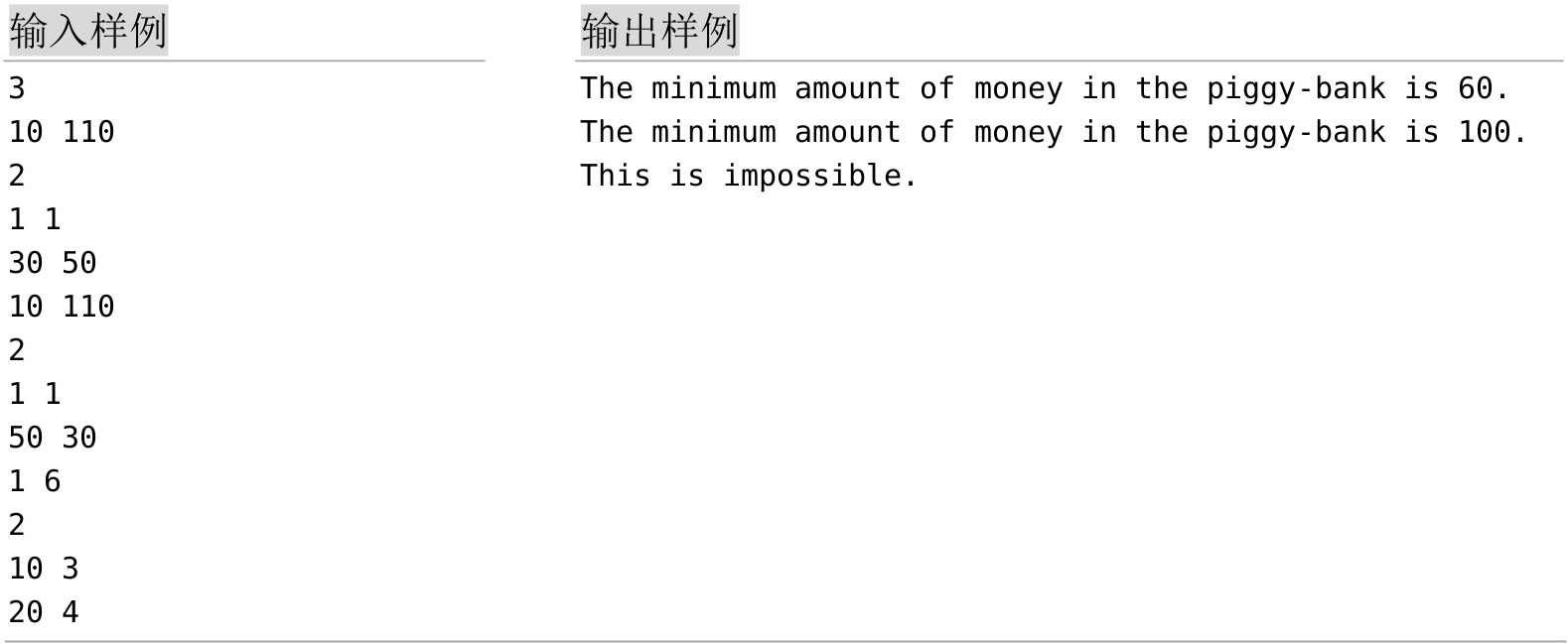


**HDU1114**

**题目描述（HDU1114）：**存钱罐有个大问题，不打碎存钱罐，就无法确定里面有多少钱，所以可能会出现把存钱罐打碎后发现钱不够的情况。唯一的可能是，称一下存钱罐的重量，试着猜里面有多少钱。已知存钱罐的重量和每种面值的硬币重量，请确定存钱罐内的最小金额。

**输入：**输入的第1行包含整数T，表示测试用例的数量。每个测试用例的第1行都包含两个整数e和f（1≤e≤f≤10000），分别表示空存钱罐和装满硬币的存钱罐的重量（以克计）。第2行包含一个整数n（1≤n≤500），表示硬币的总数量。接下来的n行，每行都包含两个整数p和w（1≤p≤50000，1≤w≤10000），分别表示硬币的面值和重量。

**输出：**对每个测试用例，都输出一行，包含“The minimum amount of money in the piggy-bank is x”，其中x是存钱罐内的最小金额。若无法确定，则输出“This is impossible.”。

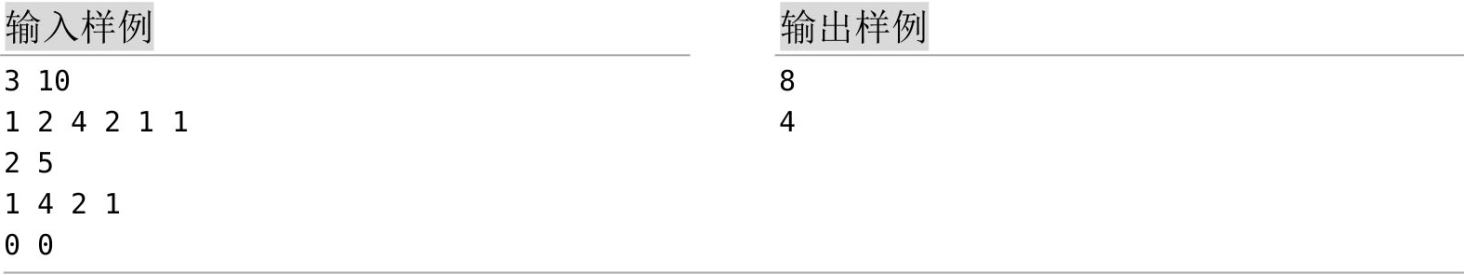


**HDU2844**

**题目描述（HDU2844）：**小明想买一只非常漂亮的手表，他知道价格不会超过m，但不知道手表的确切价格。已知硬币的面值a1, a2, a3, …, an和该面值的数量c1, c2, c3, …, cn，计算可以用这些硬币支付多少种价格（1～m）。

**输入：**输入包含几个测试用例。每个测试用例的第1行都包含两个整数n（1≤n≤100）、m（m≤100000）；第2行包含2n个整数a1, a2, a3, …, an、c1, c2, c3, …, cn （1≤ai≤100000，1≤ci≤1000）。在最后一个测试用例后面包含两个0，表示结束。

**输出：**对每个测试用例，都单行输出答案。

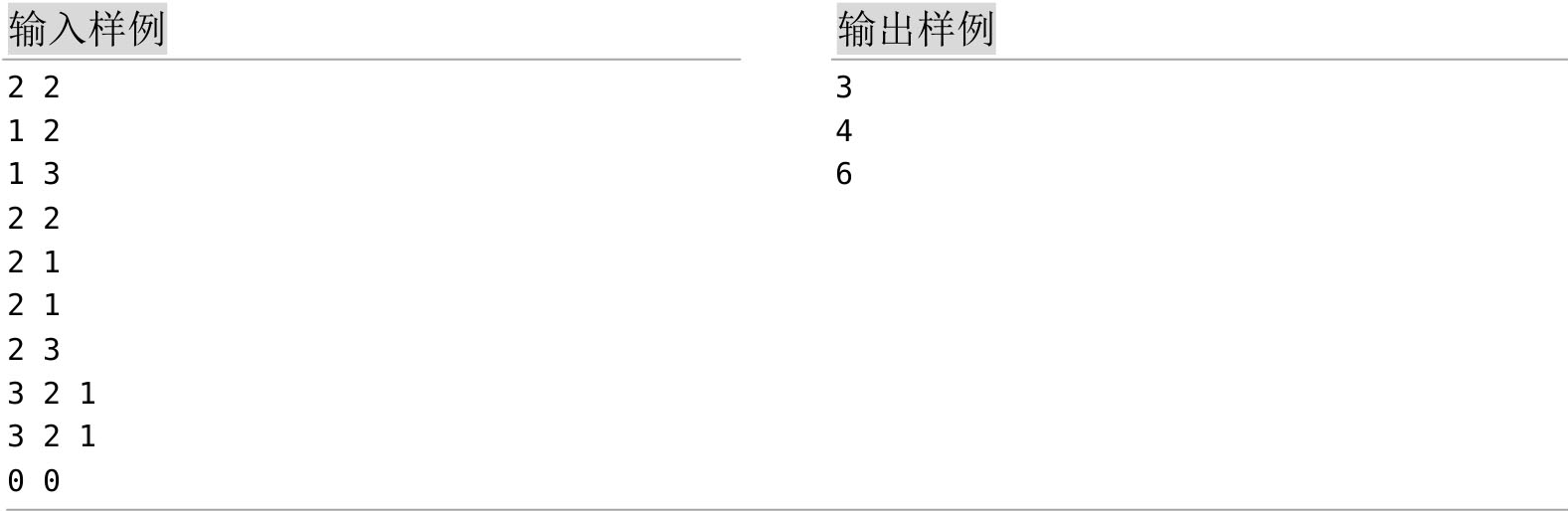


**HDU1712**

**题目描述（HDU1712）：**小明这学期有n门课程，他计划最多花m天学习。根据他在不同课程上花费的天数，他将获得不同的价值，求如何安排n门课程的m天可使价值最大化。

**输入：**输入包含多个测试用例。每个测试用例的第1行都包含两个正整数n和m，分别表示课程数和天数。接下来是矩阵a[i][j]，1≤i≤n≤100，1≤j≤m≤100。a[i][j]表示在第i门课程上花费j天将获得的价值。在n=0、m=0时结束输入。

**输出：**对每个测试用例，都单行输出获得的最大价值。



**POJ3260**

**题目描述（POJ3260）：**约翰进城买农产品时总是以最小数量的硬币来交易，即他用来支付的硬币数量和收到找零的硬币数量之和是最小的。他想购买T（1≤T≤10000）美分的用品，而硬币系统有N（1≤N≤100）种不同的硬币，面值分别为v1, v2, …, vN（1≤vi≤120）。约翰有c1个面值为v1的硬币，c2个面值v2的硬币, …, cN个面值vN（0≤ci≤10000）的硬币。店主拥有无限量的硬币，并且总是以最有效的方式进行交易（约翰必须确保通过其付款方式可以正确交易）。

**输入：**第1行有两个整数N和T。第2行有N个整数v1, v2, …, vN，表示硬币的面值。第3行有N个整数c1, c2, …, cN，表示硬币的数量。

**输出：**单行输出支付和找零的最小硬币数，若不可能支付和找零，则输出-1。



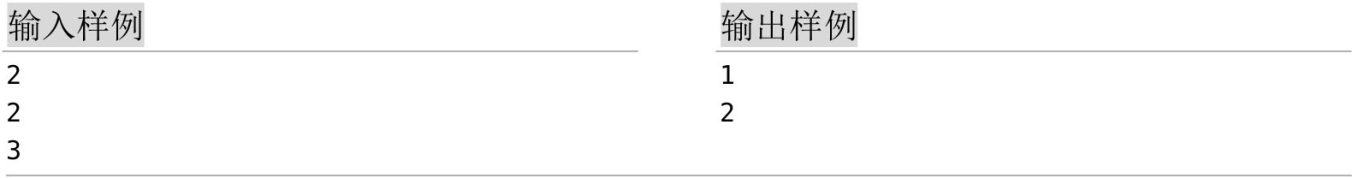
**提示：**约翰用一枚50美分和一枚25美分的硬币支付75美分，并收到一枚5美分的零钱，总共有3枚硬币用于交易。

**HDU2041**

**题目描述（HDU2041）：**一个楼梯共有M级台阶，刚开始时我们站在第1级台阶上，若每次只可以走上一级或二级台阶，则要走上第M级台阶共有多少种走法？

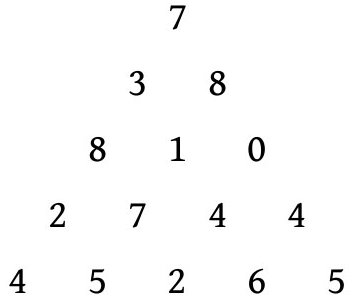
**输入：**第1行包含一个整数N，表示测试用例的个数。然后是N行数据，每行都包含一个整数M（1≤M≤40），表示楼梯的级数。

**输出：**对每个测试实例都输出不同走法的数量。



**POJ1163**

**题目描述（POJ1163）：**下图显示了一个数字三角形。每一步都可以向左斜下方走或向右斜下方走，计算从顶到底某条路线上经过的数字的最大和。



**输入：**第1行包含一个整数n（1<n≤100），表示三角形的行数。下面的n行描述了三角形的数据。三角形中的所有整数都为0～99。

**输出：**输出从顶到底某条路线上经过的数字的最大和。



**POJ2533**

**题目描述（POJ2533）：**若一个序列满足a1<a2<…<an，则该序列是有序（上升）的。设给定数字序列(a1, a2, …, an)的子序列为任意序列(ai1, ai2, …, aik)，其中1≤i1<i2<…<ik≤n，例如序列(1, 7, 3, 5, 9, 4, 8)有上升子序列如(1, 7)、(3, 4, 8)和其他子序列。所有最长的上升子序列的长度都是4，例如(1, 3, 5, 8)。当给定数字序列时，找到其最长上升子序列的长度。

**输入：**第1行包含序列的长度n（1≤n≤1000）；第2行包含序列的n个元素，每个元素都为0～10000的整数。

**输出：**输出给定序列的最长上升子序列的长度。



**POJ1458**

**题目描述（POJ1458）：**序列的子序列指序列中的一些元素被省略。给定一个序列x=<x1, x2, …, xm>及另一个序列z=<z1, z2, …, zk>，若x的索引存在严格递增的序列<i1, i2, …, ik>，则对所有j=1, 2, …, k及xij=zj，z都是x的子序列。例如，z=<a, b, f, c>的索引序列是<1, 2, 4, 6>，它是x=<a, b, c, f, b, c>的子序列。若z既是x的子序列，也是y的子序列，则称z是x和y的公共子序列。给定两个序列x和y，求x和y的最长公共子序列的长度。

**输入：**每个测试用例都包含两个表示给定序列的字符串，序列由任意数量的空格分隔。

**输出：**对每个测试用例，都单行输出最长公共子序列的长度。



**HDU1003**

**题目描述（HDU1003）：**给定一个序列a1, a2, a3, …, an，计算其最大连续字段和。例如，给定(6,-1, 5, 4,-7)，此序列的最大连续字段和为6+(-1)+5+4=14。

**输入：**的第1行包含一个整数t（1≤t≤20），表示测试用例的数量。接下来的t行，每行都以数字n为开头（1≤n≤100000），然后是n个整数（数值范围：-1000～1000）。

**输出：**对每个测试用例，都输出两行。第1行是“Case x:”，x表示测试用例的编号。第2行包含3个整数，为序列的最大连续子段和及该子段的开始位置、结束位置。若有多个结果，则输出第1个结果。在两个测试用例之间输出一个空行。



**POJ3280**

**题目描述（POJ3280）：**约翰在每头牛身上都安装了一个id标签（电子身份标签），当牛通过扫描仪时，系统会读取这个标签。每个id标签都是从n（1≤n≤26）个小写字母的字母表中提取的长度为m（1≤m≤2000）的字符串。牛有时试图通过倒退来欺骗系统。当一头牛的id标签是“abcba”时，不管它朝哪个方向走，都会读到相同的id标签，而一头牛的id标签是“abcb”时，可能会被读到两个不同的id标签（abcb和bcba）。约翰想修改牛的id标签，这样无论牛从哪个方向走过，都可以读到相同的内容。例如，“abcb”可以通过在末尾添加‘a’，形成“abcba”，这样的id标签就是回文（向前和向后读取都是相同的内容）。将id标签更改为回文的其他方法包括将“bcb”添加到开头，产生id标签“bcbabcb”；或删除字符‘a’，产生id标签“bcb”。可以在字符串中的任何位置添加或删除字符，从而生成比原始字符串长或短的字符串。给定牛的id标签及添加、删除每个字符的成本（0≤成本≤10000），求解使id标签满足回文字符串的最小成本。一个空的id标签被认为已满足要求。只有包含相关成本的字母才可以被添加到字符串中。

**输入：**第1行包含两个整数n和m。第2行包含m个字符，表示初始的id标签。第3..n+2行的每一行都包含一个字符和两个整数，分别表示添加和删除该字符的成本。

**输出：**单行输出更改给定标签为回文的最小成本。



**提示：**若在“abcb”末尾添加一个“a”，则得到“abcba”，成本是1000；若把开头的“a”删掉，则得到“bcb”，成本是1100；若在开头插入“bcb”，则得到“bcbabcb”，成本是350+200+350=900，这是最小成本。

**POJ2955**

**题目描述（POJ2955）：**“正则括号”序列的定义如下。

• 空序列是一个正则括号序列。

• 若s是正则括号序列，则(s)和[s]也是正则括号序列。

• 若a和b是正则括号序列，则ab也是正则括号序列。

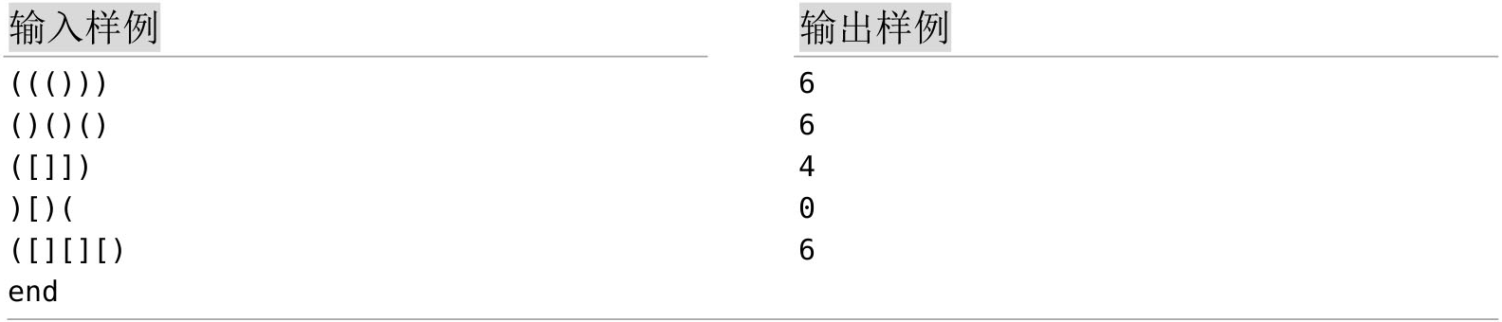
• 没有其他序列是正则括号序列。

例如，( )、[]、( ( ) )、( ) []、( ) [( ) ]都是正则括号序列，而(、]、) (、( [) ]、( [( ]不是正则括号序列。

给定括号序列a1a2…an，求解其最长的正则括号子序列的长度。也就是说，希望找到最大的m，使ai1ai2…aim是一个正则括号序列，其中1≤i1<i2<…<im≤n。例如给定初始序列( [( [] ] ) ]，最长的正则括号子序列是[( [] ) ]，其长度是6。

**输入：**输入包含多个测试用例。每个测试用例都只包含一行由(、)、[、]组成的字符串，其长度为1～100（包括1和100）。输入的结尾由包含“end”的行标记，不应对其进行处理。

**输出：**对每个测试用例，都单行输出最长的正则括号子序列的长度。



**HDU3506**

**题目描述（HDU3506）：**森林之王决定举办一个盛大的派对来庆祝香蕉节，但是小猴子们都不认识对方。有N只猴子坐在一个圈里，每只猴子都有交朋友的时间，而且每只猴子都有两个邻居。介绍它们的规则是：①森林之王每次都可以介绍一只猴子和该猴子的一个邻居；②若森林之王介绍A和B，则A已经认识的每只猴子都将认识B已经认识的每只猴子，介绍的总时间是A和B已经认识的所有猴子交友时间的总和；③每只猴子都认识自己。为了尽快开始聚会和吃香蕉，需求出森林之王需要介绍的时间。

**输入：**输入包含几个测试用例。每个测试用例的第1行都是n（1≤n≤1000），表示猴子的数量。下一行包含n个正整数（小于1000），表示交朋友的时间（第1个和最后1个是邻居）。

**输出：**对每个测试用例，都单行输出需要介绍的时间。



**POJ1651**

**题目描述（POJ1651）：**乘法游戏是用一些牌来玩的，在每张牌上都有一个正整数。玩家从一行牌中取出一张牌，得分的数量等于所取牌上的数字与左右两张牌上的数字的乘积。不允许取出第一张和最后一张牌。经过最后一步后，只剩下两张牌。玩牌的目标是把得分的总数降到最低。例如，若一行牌包含数字10、1、50、20、5，则若玩家先拿出一张1，然后拿出20和50的牌，得分便是10×1×50+50×20×5+10×50×5=500+5000+2500=8000。若他按相反的顺序拿牌，即50、20、1，则得分是1×50×20+1×20×5+10×1×5=1000+100+50=1150。

**输入：**第1行包含牌的数量n（3≤n≤100），第2行包含1～100的n个整数，表示牌上的数字。

**输出：**单行输出玩牌的最小分数。

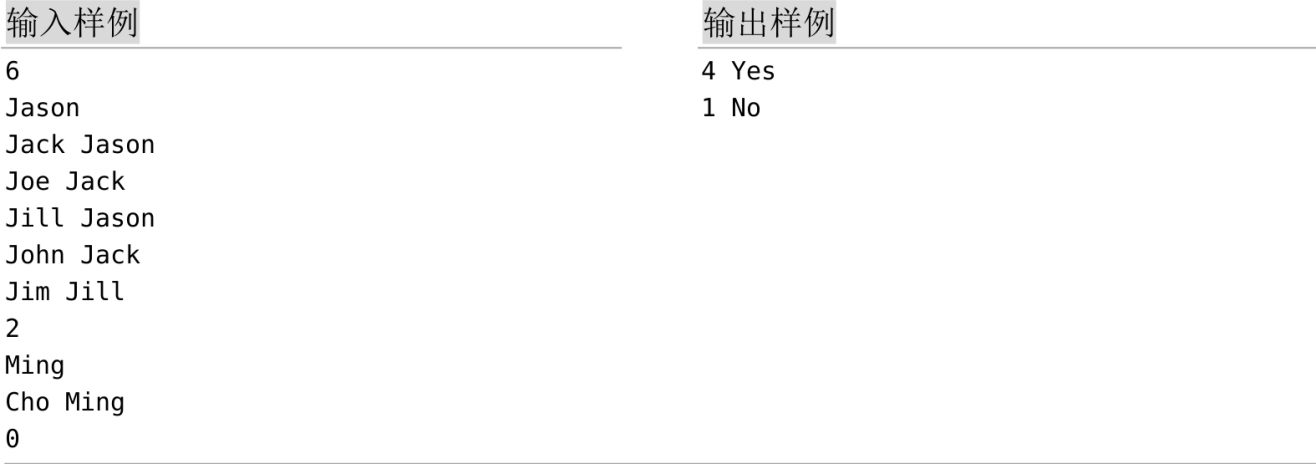


**POJ3342/HDU2412/UVA1220**

**题目描述（POJ3342/HDU2412/UVA1220）：**约翰要在别墅开派对，希望可以邀请所有同事，但不同时邀请员工和老板。公司的组织层级是这样的：除了大老板，每个人都有唯一的老板（直接上司），当一个人被邀请时，他的老板不会被邀请，请确定邀请客人的最大数量。另外，需要表明客人列表中的人是否是唯一确定的。

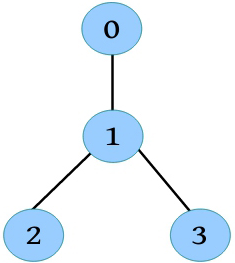
**输入：**输入包含多个测试用例。每个测试用例都以整数n（1≤n≤200）开始，表示员工的数量。下一行只包含大老板的名字。在接下来的n-1行中，每一行都包含员工的名字及其老板的名字。所有名字都是由至少一个和最多100个字母组成的字符串，以空格隔开。每个测试用例的最后一行都包含一个0。

**输出：**对每个测试用例，都单行输出一个数字和一个单词，分别表示邀请客人的最大数量和客人列表是否唯一。



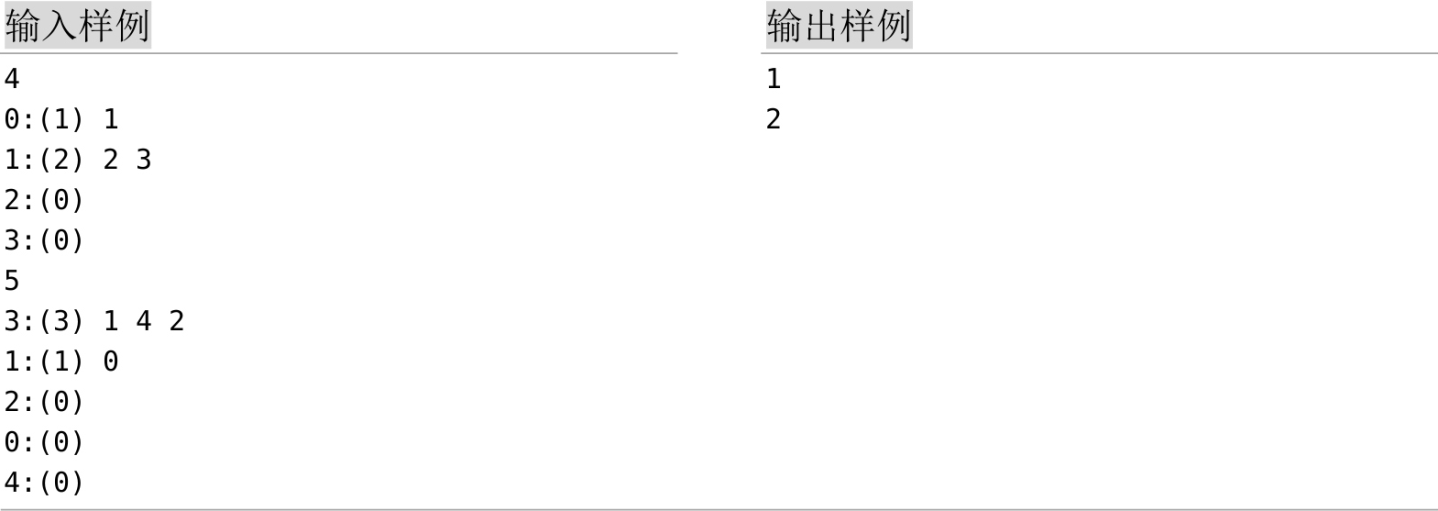
**POJ1463**

**题目描述（POJ1463）：**鲍勃喜欢玩战略游戏，但他有时找不到足够快的解决方案。现在他必须保卫一座中世纪城市，城市的道路形成一棵树。他必须把最小数量的士兵放在节点上，这样才可以观察到所有道路。请帮助鲍勃找到放置的最小士兵数。例如，对如下图所示的树，解决方案是放置1个士兵（放置在节点1处）。

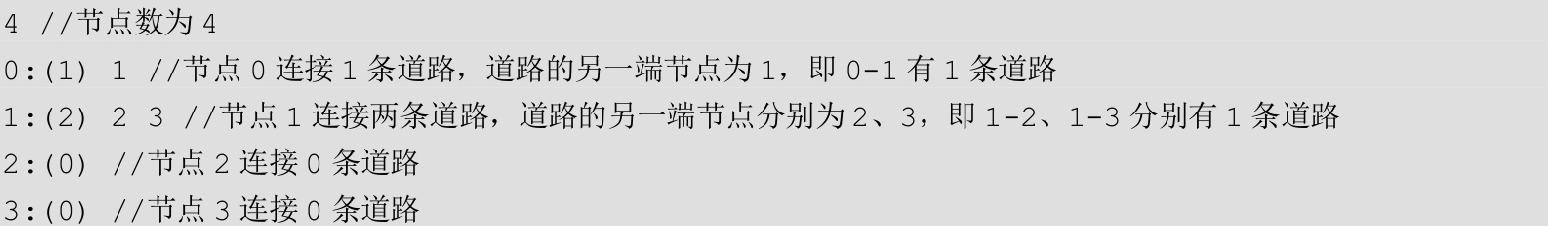


**输入：**输入多个测试用例。每个测试用例的第1行都包含节点数n（0<n≤1500）；接下来的n行，每行的描述格式都为“节点编号：（道路数）节点编号1节点编号2…”或“节点编号：(0)”。节点编号为0～n-1，每个节点连接的道路数都不超过10条。每条道路在输入数据中都只出现一次。

**输出：**对每个测试用例，都单行输出放置的最小士兵数。



对输入样例1的数据解释如下所示：



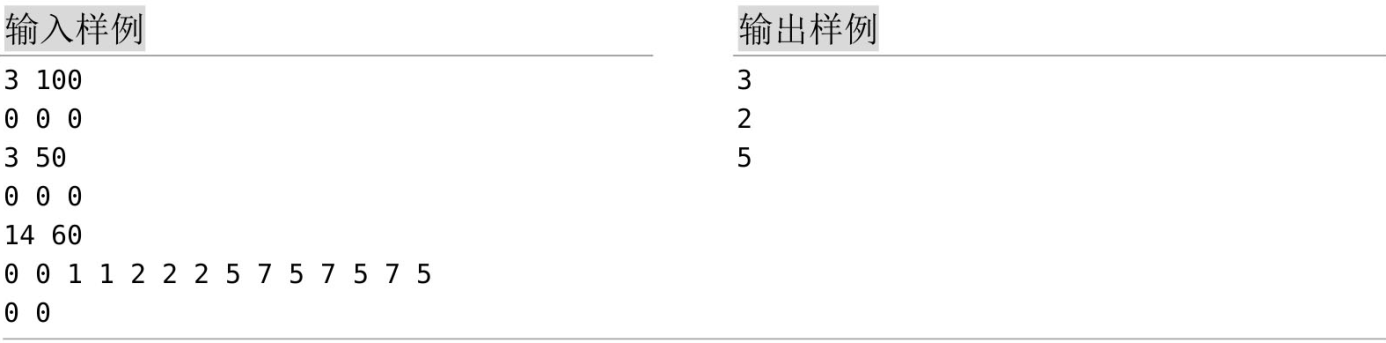
其对应的树形结构如题目描述中的树。

**UVA12186**

**题目描述（UVA12186）：**公司有一个严格的等级制度，除了大老板，每个员工都只有一个老板（直接上司）。不是其他员工老板的员工被称为工人，其余的员工和老板都叫作老板。要求加薪时，工人应向其老板提出请愿书。若至少 T%的直接下属提交请愿书，则该老板会有压力，向自己的老板提交请愿书。每个老板最多向自己的老板提交一份请愿书。老板仅统计他的直接下属的请愿书数量来计算压力百分比。当一份请愿书被提交给公司大老板时，所有人的工资都会增加。请找出为使大老板收到请愿书而必须提交请愿书的最少工人数。

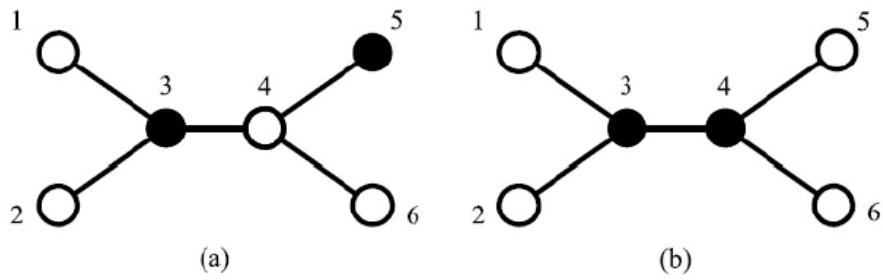
**输入：**输入包含几个测试用例。每个测试用例都包括两行，第1行包含两个整数n和T（1≤n≤105，1≤T≤100），n表示公司的员工人数（不包括公司大老板），T是上面描述的参数。每个员工的编号都为1～n，大老板编号为0；第2行包含整数列表，列表中的位置i （从1开始）为整数bi（0≤bi≤i-1），表示员工i的直接上司的编号。在最后一个测试用例后面包含两个0。

**输出：**对每个测试用例，都单行输出为使大老板收到请愿书而必须提交请愿书的最少工人数。



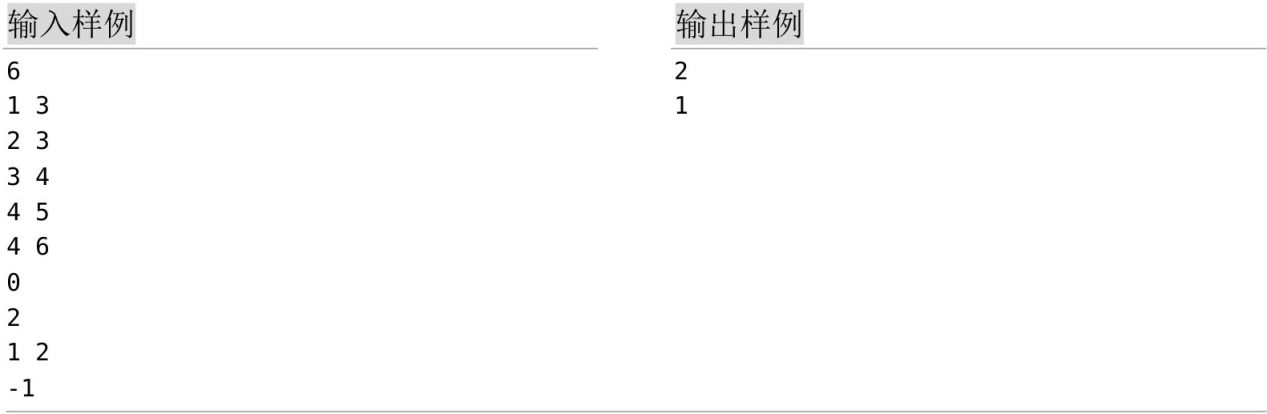
**POJ3398/UVA1218**

**题目描述（POJ3398/UVA1218）：**网络由n台计算机组成，这些计算机通过n-1个通信链路连接，使得任意两台计算机都可以通过唯一的路由进行通信。若两台计算机之间有通信链路，则称它们相邻。计算机的邻居是与其相邻的一组计算机。需要选择一些计算机作为服务器，服务器可以为其所有邻居都提供服务。若每台客户机（非服务器）都只由一台服务器提供服务，则网络中的一组服务器就形成了完美服务，形成完美服务的最小服务器数叫作完美服务数。例如，下图显示了由6台计算机组成的网络，其中黑色节点表示服务器，白色节点表示客户机。图(a)中的服务器3和5不形成完美服务，因为客户机4与服务器3和5相邻，由两台服务器提供服务。图(b)中的服务器3和4形成完美服务，且完美服务数等于2。



**输入：**输入包含多个测试用例。每个测试用例的第1行都包含一个正整数n（n≤10000），表示网络中的计算机数，编号为1～n。接下来的n-1行，每行都包含两个正整数，表示一个通信链路。第n+1行的0表示第1个测试用例的结束，-1表示整个输入的结束。

**输出：**对每个测试用例，都单行输出完美服务数。

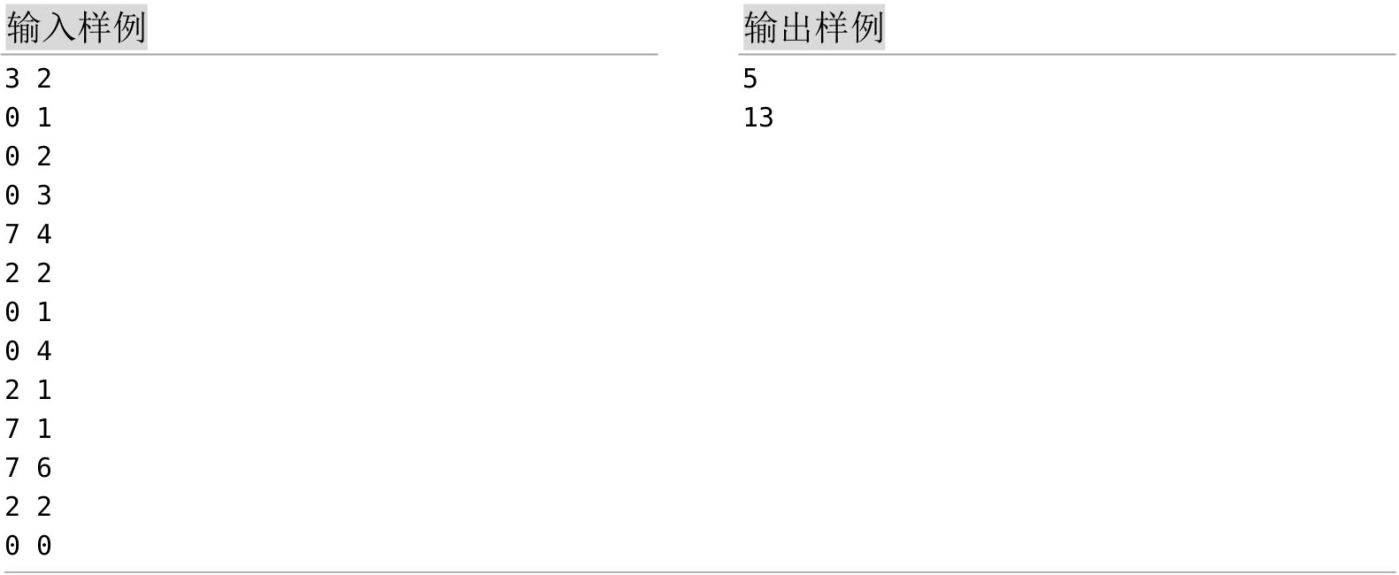


**HDU1561**

**题目描述（HDU1561）：**在一个地图上有N座城堡，每座城堡都有一定的宝物。在每次游戏中都允许攻克M个城堡并获得里面的宝物。但有些城堡不可以直接攻克，要攻克这些城堡，必须先攻克其他某个特定的城堡。计算攻克M个城堡最多可以获得的宝物数量。

**输入：**每个测试实例都首先包括两个整数N和M（1≤M≤N≤200）。接下来的N行，每行都包括两个整数a、b。在第i行中，a表示要攻克第i个城堡，则必须先攻克第a个城堡，若a=0，则表示可以直接攻克第i个城堡；b（b≥0）表示第i个城堡的宝物数量。当N=0、M=0时输入结束。

**输出：**对每个测试实例，都单行输出攻克M个城堡最多可以获得的宝物数量。

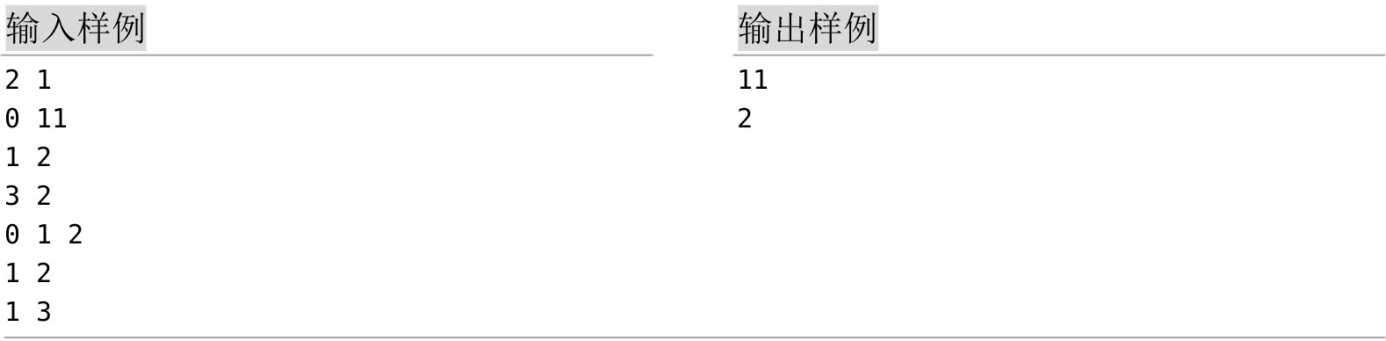


**POJ2486**

**题目描述（POJ2486）：**一棵虚拟的苹果树有n个节点，每个节点都有一定数量的苹果。从节点1出发，可以吃掉到达节点的所有苹果。当从一个节点转到另一个相邻节点时，需要走一步。计算经过k步最多吃多少个苹果。

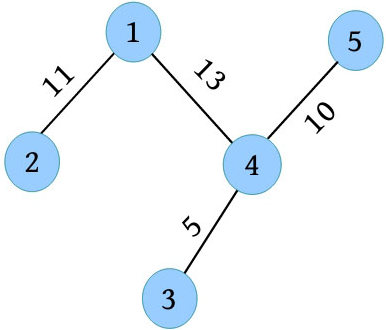
**输入：**输入包含几个测试用例。每个测试用例都包含3部分。第1部分包含两个数字n、k（1≤n≤100，0≤k≤200），分别表示节点数和所走的步数，节点编号为1～n。第2部分包含n个整数（所有整数均非负且不大于1000），第i个整数表示节点i的苹果数量。第3部分包含n-1行，每行都包含两个整数a、b，表示节点a和节点b是相邻的。

**输出：**对每个测试用例，都单行输出经过k步可以吃到的最大苹果数量。

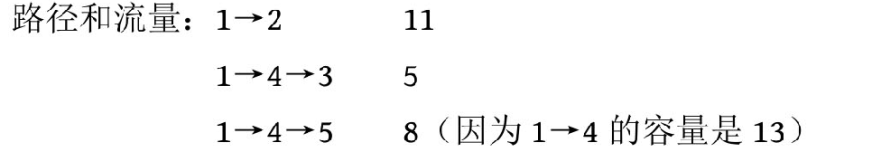


**POJ3585**

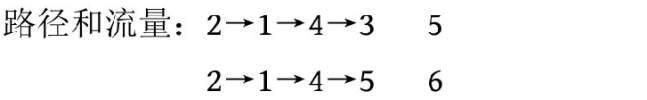
**题目描述（POJ3585）：**a(x)表示树中节点x的累积度，定义如下：①树的每个边都有一个正容量；②树中度为1的节点叫作终端；③每条边的流量都不可以超过其容量；④a(x)是节点x可以流向其他终端节点的最大流量。示例如下图所示。



（1）a(1)=11+5+8=24



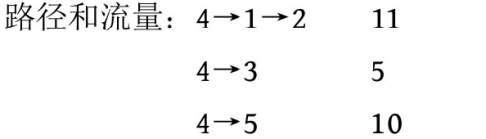
（2）a(2)=5+6=11



（3）a(3)=5



（4）a(4)=11+5+10=26



（5）a(5)=10



树的累积度是树中节点的最大累积度。

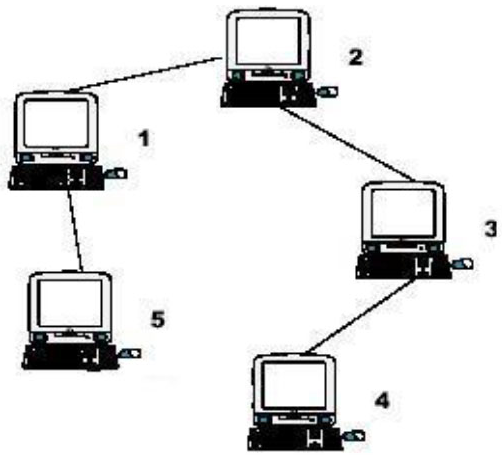
**输入：**第1行是一个整数t，表示测试用例的数量。每个测试用例的第1行都是一个正整数n，表示节点数，节点编号为1～n。下面n-1行中的每一行都包含三个整数x、y、z，表示在节点x和节点y之间有一条边容量为z。所有元素都是不超过200000的非负整数。

**输出：**对每个测试用例，都单行输出树的累积度。



**HDU2196**

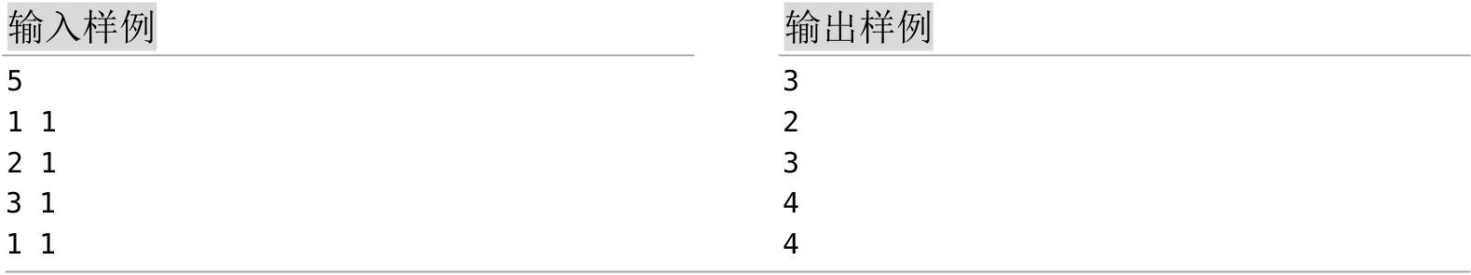
**题目描述（HDU2196）：**学校不久前买了第1台计算机（编号为1）。近年来，学校又买了N-1台新计算机。每台新计算机都被连接到先前安装的一台计算机上。学校管理者担心网络运行缓慢，想知道第i台计算机发送信号的最大距离si（即电缆到最远的计算机的长度）。



**提示：**输入样例对应上图，可以看出，计算机1距离4最远，最远电缆长度为3，所以s1=3。计算机2距离5和4最远，最远电缆长度s2=2。计算机3距离5最远，最远电缆长度s3=3。同理，得到s4=4、s5=4。

**输入：**输入包含多个测试用例。每个测试用例的第1行都为n（n≤10000），后面n-1行为对计算机的描述。第i行包含两个自然数，分别表示连接第i台计算机的计算机编号和用于连接的电缆长度。电缆总长度不超过109。

**输出：**对每个测试用例，都输出n行，第i行表示第i台计算机到其他计算机的最远距离。



**HDU2089**

**题目描述（HDU2089）：**很多人都不喜欢在车牌中有不吉利的号码，不吉利的号码为所有包含4或62的号码，例如62315、73418、88914都属于不吉利的号码。但是，61152虽然含有6和2，但不是62连号，所以不属于不吉利的号码。

**输入：**输入整数对n、m（0<n≤m<1000000），遇到都是0的整数对时，输入结束。

**输出：**对每个整数对都单行输出[n, m]区间不包4或62的号码个数。



**HDU3555**

**题目描述（HDU3555）：**反恐怖分子在尘土中发现了一枚定时炸弹，但这次恐怖分子改进了定时炸弹。定时炸弹的数字序列从1到n。若当前的数字序列包括子序列“49”，则爆炸的力量会增加一个点。现在反恐人员知道了数字n，他们想知道最后的力量点。

**输入：**输入的第1行包含一个整数T（1≤T≤10000），表示测试用例的数量。对每个测试用例，都有一个整数n（1≤n≤263-1）作为描述。

**输出：**对每个测试用例，都输出一个整数，表示最终的力量点。



**提示：**[1,500]区间包括“49”的数是49、149、249、349、449、490、491、492、493、494、495、496、497、498、499，所以答案是15。

**POJ3252**

**题目描述（POJ3252）：**若正整数n的二进制形式0的个数大于或等于1的个数，则称其为Round Numbers。例如，整数9的二进制形式是1001，1001有两个0和两个1，所以9是一个Round Numbers。整数26在二进制中是11010，因为它有两个0和三个1，所以它不是一个Round Numbers。计算在输入范围（1≤start<finish≤2000000000）内出现了多少个Round Numbers。

**输入：**以两个空格分隔的整数，分别是start和finish。

**输出：**单行输出[start, finish]区间Round Numbers的个数。

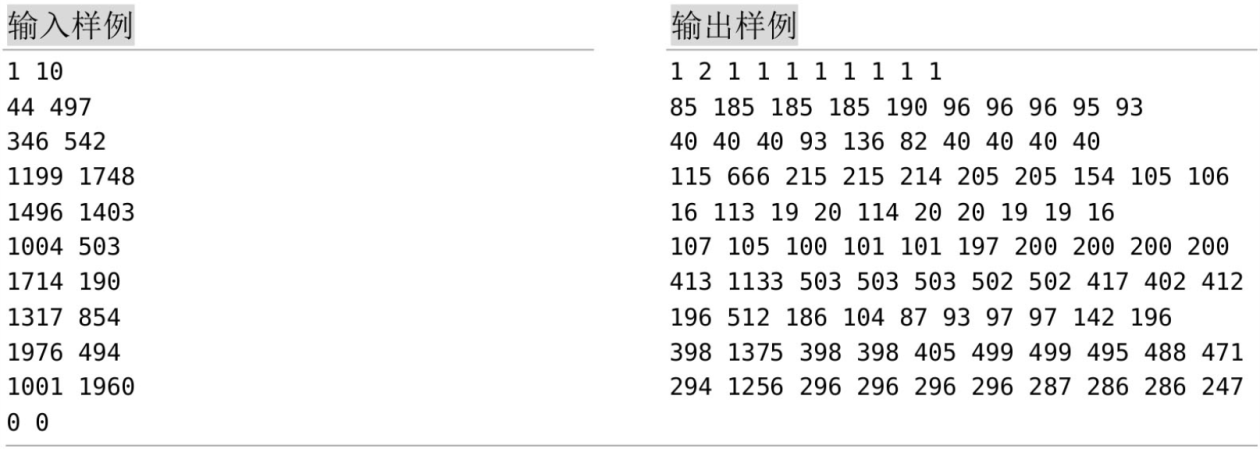


**POJ2282**

**题目描述（POJ2282）：**给定两个整数a和b，将[a, b]区间的数写在一个列表中。计算每个数字（0～9）的出现次数。例如，若a=1024和b=1032，则列表是1024 1025 1026 1027 1028 1029 1030 1031 1032，列表中包含10个0、10个1、7个2、3个3，等等。

**输入：**输入最多由500行组成，每行都包含两个数字a和b，其中0<a，b<100000000。输入由一行“0 0”终止，对该行不做处理。

**输出：**对每对输入都输出一行，包含由单个空格分隔的10个数。第1个数是数字0的出现次数，第2个数是数字1的出现次数，等等。



**HDU4734**

**题目描述（HDU4734）：**十进制数x包含n个数字（an, an-1, an-2, …, a2, a1），它的权值被定义为F(x)=an×2n-1+an-1×2n-2+…+a2×2+a1×1。给定两个数字A和B，请计算[0, B]区间有多少个数字的权值不超过F(A)。

**输入：**第1行包含一个数字T（T≤10000），表示测试用例的数量。每个测试用例都有两个数字A和B（0≤A，B<109）。

**输出：**对每个测试用例，都先输出“Case #t:”（t是从1开始的测试用例号），然后输出答案。

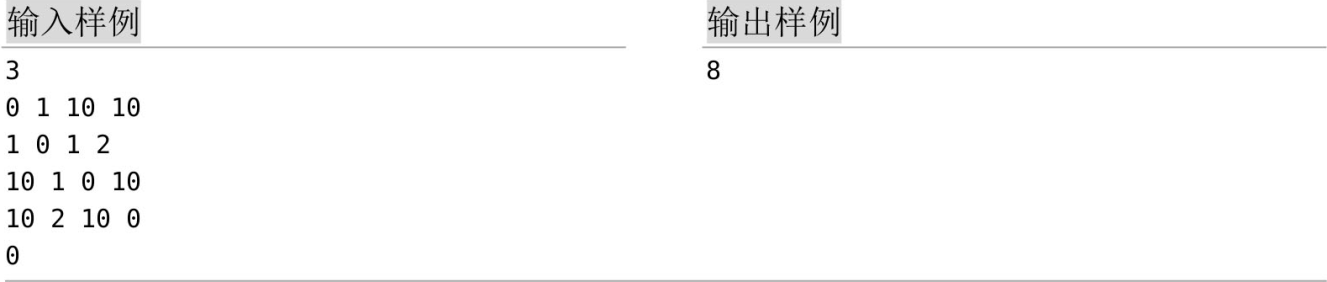


**POJ3311**

**题目描述（POJ3311）：**披萨店以尽可能快地向顾客提供披萨而自豪。司机将等待一个或多个（最多10个）订单被处理，然后开始送货。他愿意走最短的路线运送这些货物，然后返回比萨店，即使这意味着途中要经过相同的地点或披萨店不止一次。

**输入：**输入包含多个测试用例。每个测试用例的第1行都包含一个整数n（1≤n≤10），表示要交付的订单数量。之后n+1行中的每一行都包含n+1个整数，表示披萨店（编号0）和n个位置（编号为1～n）之间的行程时间。第i行上的第j个值表示从位置i直接到位置j的时间，时间值可能不对称，即从位置i直接到位置j的时间可能与从位置j直接到位置i的时间不同。n=0时将终止输入。

**输出：**对每个测试用例，都单行输出交付所有披萨并返回披萨店的最短时间。



**HDU3001**

**题目描述（HDU3001）：**阿克默决定参观n个城市，他要参观所有城市，不介意哪座城市是他的起点。有m条道路照常收费，但他不想去一座城市超过两次，想把总费用降到最低。

**输入：**输入包含几个测试用例，每个测试用例的第1行都包含两个整数n（1≤n≤10）和m，表示n个城市、m条道路。接下来的m行，每行都包含三个整数a、b和c（1≤a，b≤n），表示在a和b之间有一条道路，费用是c。

**输出：**对每个测试用例，都单行输出应支付的最低费用，若找不到这样的路线，则输出-1。

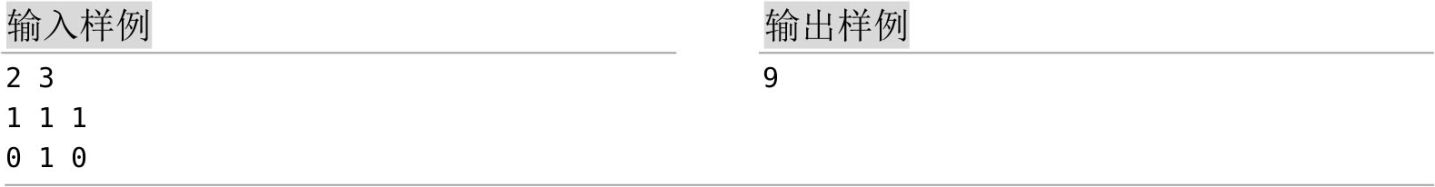


**POJ3254**

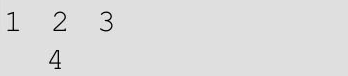
**题目描述（POJ3254）：**约翰购买了由m×n（1≤m，n≤12）的方格组成的矩形牧场，想在一些方格上种玉米。遗憾的是，有些方格土壤贫瘠，无法种植。约翰在选择种植哪些方格时，会避免选择相邻的方格，没有两个选定的方格共享一条边。约翰考虑了所有可能的选择，他认为没有选择方格也是一种有效的选择！帮助他选择种植方格的方案数。

**输入：**第1行包含以两个空格分隔的整数m和n。后面有m行，每行都包含n个整数，表示一个方格是否肥沃（1表示肥沃，0表示贫瘠）。

**输出：**单行输出选择种植方格的方案数模100000000。

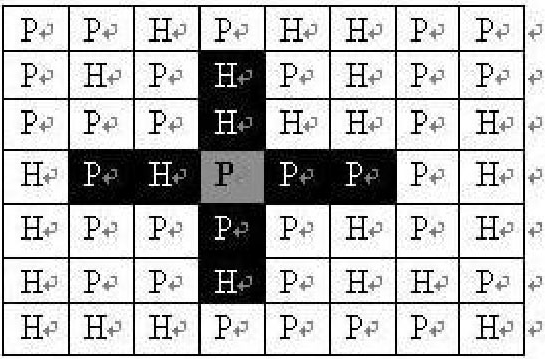


**提示：**按如下方式对肥沃的方格进行编号，仅在一个方格上种植有4种方案（1、2、3或4），在两个方格上种植有3种方案（13、14或34），在三个正方形上种植有1种方案（134），还有1种方案是所有方格都不种植。所以一共有9种方案。



**POJ1185**

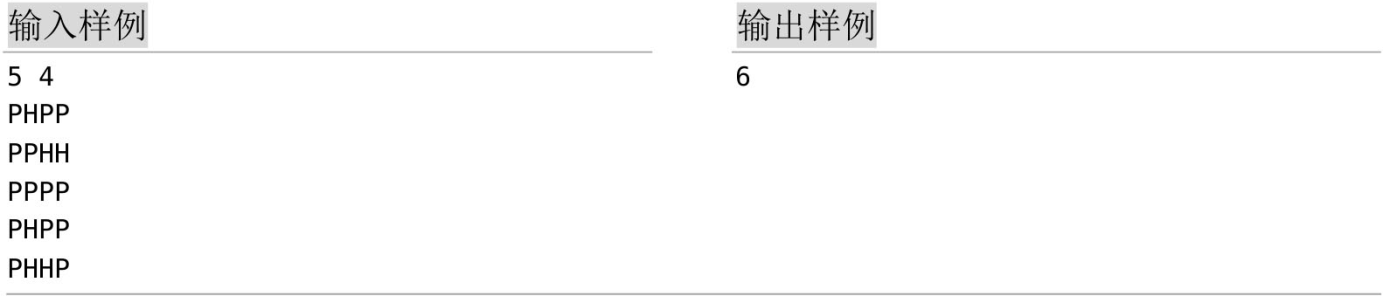
**题目描述（POJ1185）：**将军打算在地图上部署炮兵部队。地图由N行M列组成，地图的每一格都可能是山地（用H表示），也可能是平原（用P表示）。在每一格平原上最多可以部署一支炮兵部队（在山地上不可以部署炮兵部队）。一支炮兵部队在地图上的攻击范围如下图中黑色区域所示。



若在地图中灰色所标识的平原上部署一支炮兵部队，则图中的黑色网格表示它可以攻击到的区域：沿横向左右各两格，沿纵向上下各两格。不能攻击图上的其他白色网格。从图上可见炮兵的攻击范围不受地形的影响。将军们将规划部署炮兵部队，在防止误伤的前提下（任何一支炮兵部队都不在其他炮兵部队的攻击范围内），求整个地图区域内最多可以部署多少炮兵部队。

**输入：**第1行包含两个正整数N和M（N≤100，M≤10），表示N行M列。接下来的N行，每一行都包含M个字符（H或P），表示地图上的山地或平原。

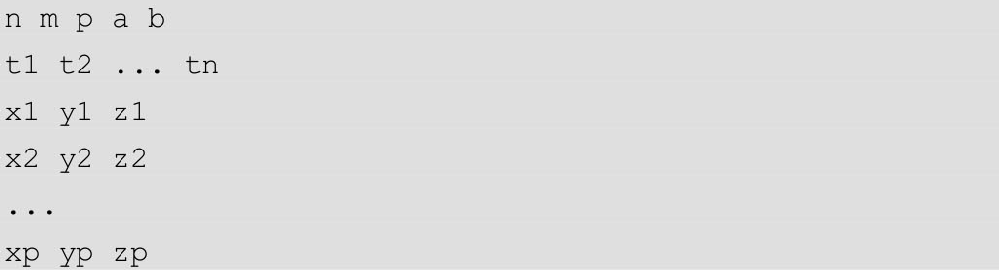
**输出：**单行输出最多可以部署的炮兵部队的数量。



**POJ2686**

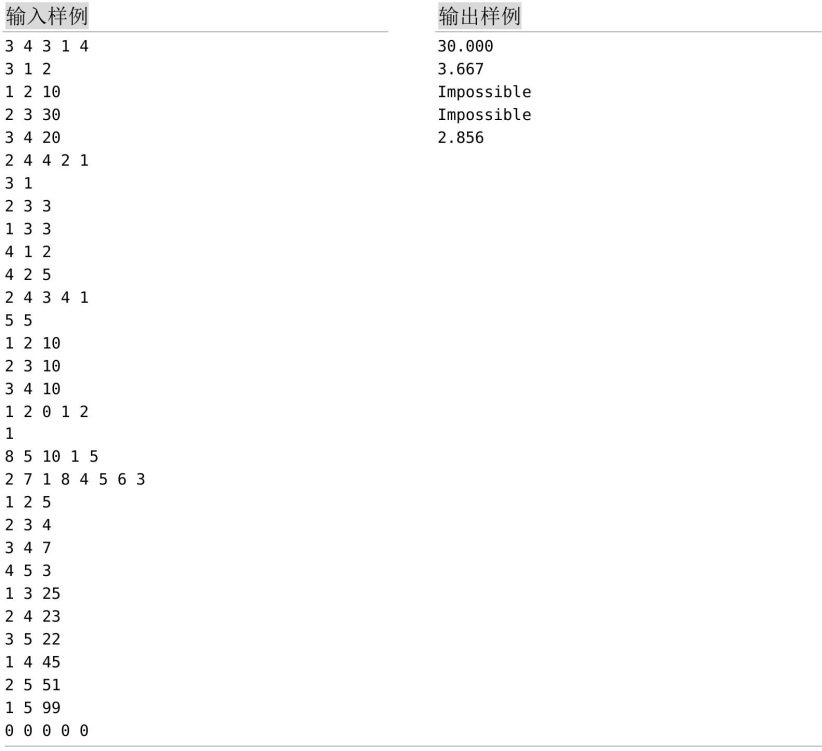
**题目描述（POJ2686）：**有一个旅行者计划乘马车旅行，他的出发点和目的地是固定的，但不能确定路线。全国的城市有一个公路网，若在两个城市之间有一条路，则可以坐公共马车从一个城市到另一个城市。乘马车需要一张票，在每张票上都注明了马的数量。当然，马越多，马车跑得越快。在出发点，旅行者有许多车票。通过考虑这些车票和道路网络上的信息，我们应该能找到在最短时间内把他带到目的地的最佳路线。应考虑怎样使用车票，假设以下条件：①乘马车可以把旅行者从一个城市直接带到另一个通过公路相连的城市。换而言之，每到一个城市，他都必须换车；②在通过公路直接连接的两个城市之间只可以使用一张车票；③每张车票都只可以使用一次；④乘马车所需的时间是两个城市之间的距离除以马的数量；⑤应忽略换乘所需的时间。

**输入：**输入由多个数据集组成，每个数据集的格式如下。在最后一个数据集后面是一行，包含5个0（用空格分隔）。



数据集中的每个输入项都是非负整数。n是长途汽车票的数量，1≤n≤8；m是城市数，2≤m≤30；p是道路数，可能为0；a是起始城市的编号，b是目的地城市的编号，a≠b。所有城市的编号都为1～m。第2行给出了车票信息，ti是第i张车票的马数（1≤i≤n，1≤ti≤10）。以下p行给出城市之间的道路信息。第i条道路将两个城市xi和yi连接起来，并有距离zi（1≤i≤p，1≤zi≤100）。两个城市之间最多一条道路，一条路不会连接城市自己，每条路都可双向行驶。

**输出：**若旅行者可以到达目的地，则输出出发点到目的地的最短时间。答案的误差不应大于0.001。在满足上述精度条件的前提下，可以输出小数点后的任意位数。若无法到达目的地，则输出“Impossible”。

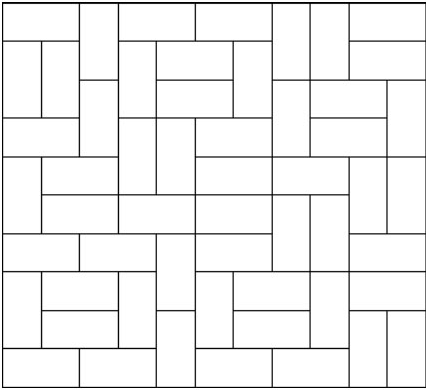


**POJ2411**

**题目描述（POJ2411）：**荷兰著名画家蒙德里安着迷于正方形和长方形，梦想着用不同的方式将高1宽2的小长方形填满一个大长方形。



计算填充大长方形（其大小也是整数值）的方案数。



**输入：**包含几个测试用例，每个测试用例都由大长方形的高度h和宽度w两个整数组成（1≤h, w≤11）。输入以0 0结束。

**输出：**对每个测试用例，都输出用1×2的小长方形填充给定长方形的方案数，假设给定的大长方形是定向的，即多次计算对称的瓷砖。



**HDU1565**

**题目描述（HDU1565）：**一个n×n的格子棋盘，在每个格子里面都有一个非负数，从中取出若干数，所取的数不可以相邻并且取出的数之和最大。

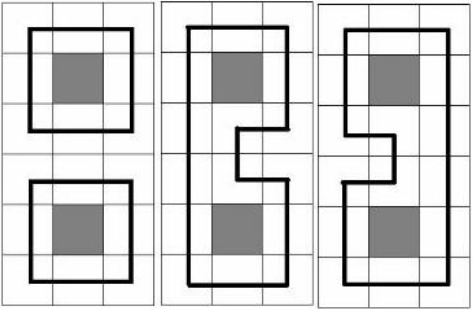
**输入：**包含多个测试实例，每个测试实例都包含一个整数n和n×n个非负数（n≤20）。

**输出：**对每个测试实例都单行输出可能取得的最大和值。



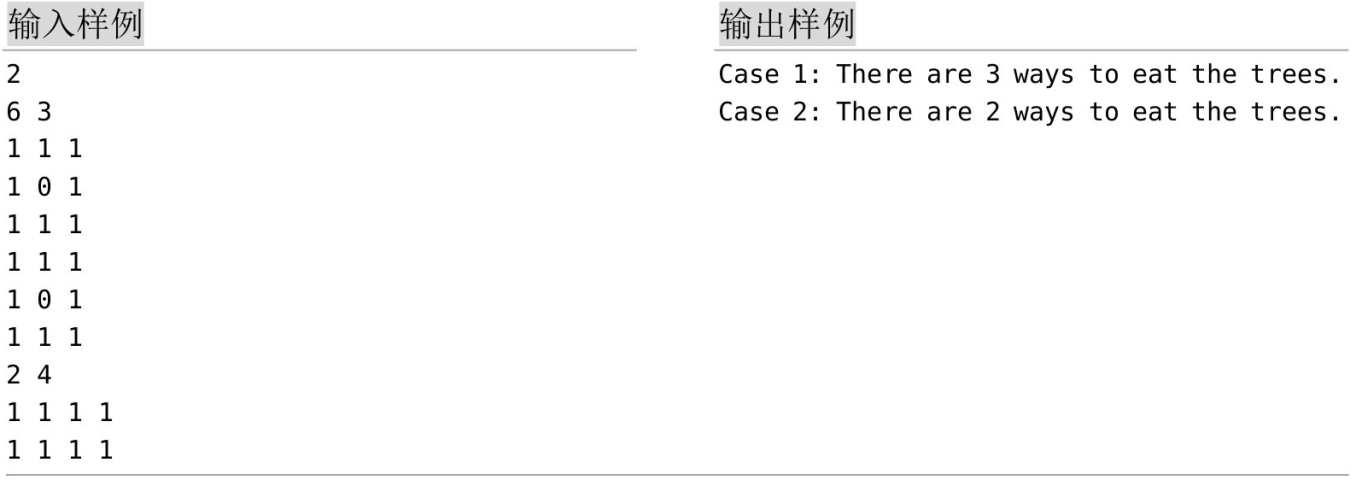
**HDU1693**

**题目描述（HDU1693）：**在Dota（古代防御）游戏中，普吉的队友给了他一个新的任务“吃树”。这些树都是大小为n×m的矩形单元格，每个单元格要么只有一棵树，要么什么都没有。普吉需要做的是“吃掉”单元格里的所有树。他必须遵守几条规则：①必须通过选择一条回路来吃掉这些树，然后吃掉所选回路中的所有树；②不包含树的单元格是不可被访问的，例如，选择的回路通过的每个单元格都必须包含树，当选择回路时，回路上单元格中的树将消失；③可以选择一个或多个回路来吃这些树。有多少方法可以吃这些树？在下图中为n=6和m=3给出了三个样本（灰色方块表示在单元格中没有树，粗体黑线表示所选的回路）。



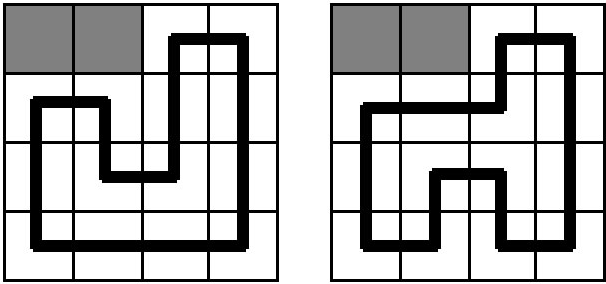
**输入：**输入的第1行是测试用例数T（T≤10）。每个测试用例的第1行都包含整数n和m（1≤n, m≤11）。在接下来的n行中，每行都包含m个数字（0或1），0表示没有树的单元格，1表示只有一棵树的单元格。

**输出：**对每个测试用例，都单行输出有多少种方法可以吃这些树，保证不超过263-1。



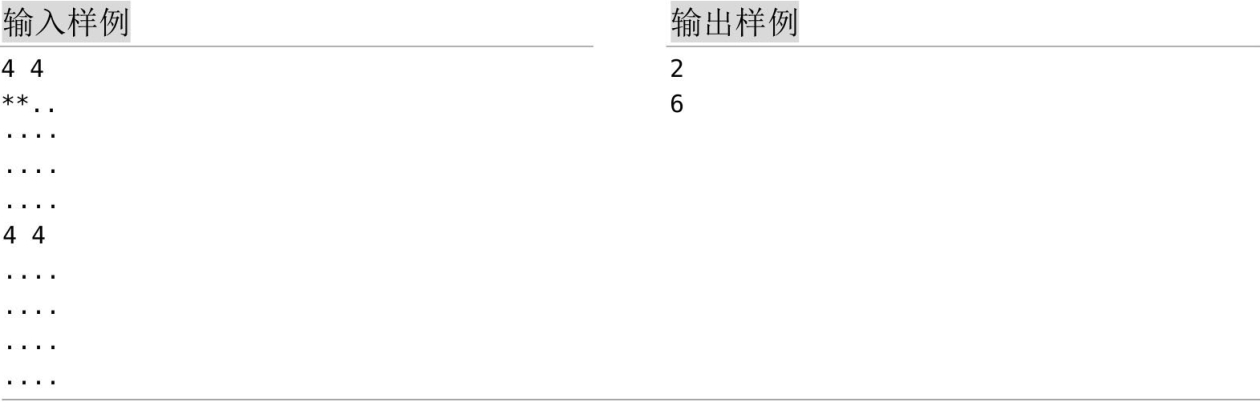
**URAL1519**

**题目描述（URAL1519）：**W市将举办一级方程式赛事，需要建立一个新的赛道。但是在未来的赛道上有许多地鼠生活在洞里，不允许在洞上建造赛道。赛道是一个n×m的矩形单元，每个单元都有一个单独的路段，每个路段都应该与矩形的一条边平行，所以赛道只可以有90º转弯。在下图中给出了n=m=4的两个样例（灰色方块表示地鼠洞，粗体黑线表示赛道）。求解有多少种方法可以建立赛道。



**输入：**第1行包含整数n和m（2≤n，m≤12），表示行数和列数。接下来的n行中，每一行都包含m个字符，字符“。”表示可以在该单元建立赛道，字符“\*”表示该单元有地鼠洞。至少有4个单元格没有地鼠洞。

**输出：**输出建立赛道的方法数，保证不超过263-1。

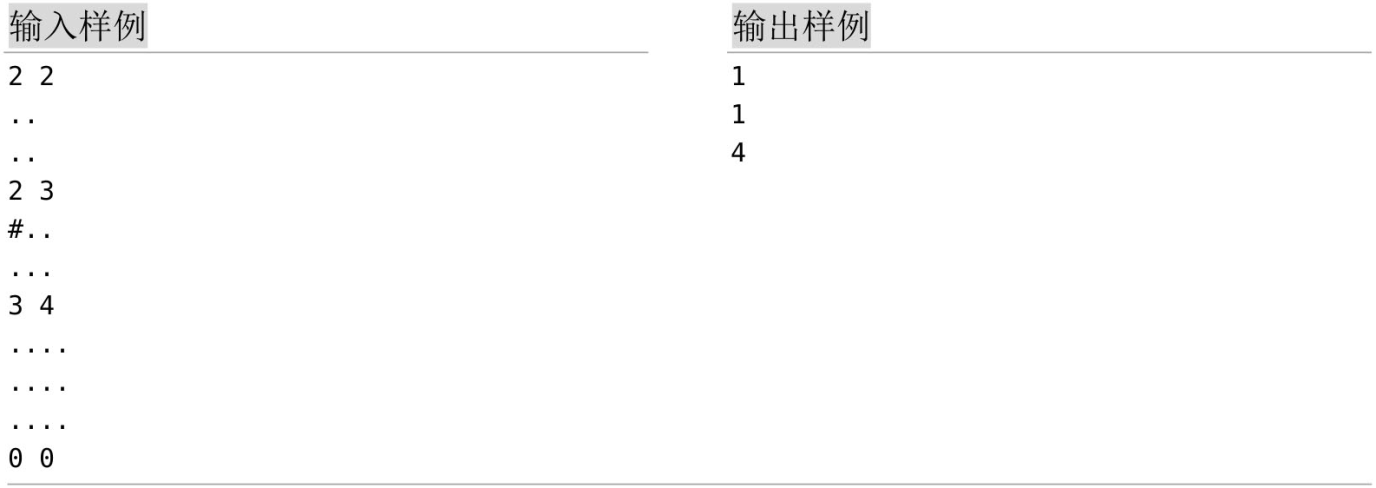


**POJ1739**

**题目描述（POJ1739）：**一个方形乡镇被划分为n×m个方块（1≤n, m≤8），有的封锁，有的畅通。农场位于左下方，市场位于右下方。托尼打算从农场到市场进行一次乡间旅行，对每一块没有封锁的土地都要走一次。计算从农场到市场有多少种不重复的旅游方案。

**输入：**包含几个测试用例。每个测试用例的第1行都包含两个整数n、m，表示行数和列数。下面的n行，每行都包含m个字符。“#”表示封锁的方格，“.”表示未封锁的方格。在最后一个测试用例后面跟着两个0。

**输出：**对每个测试用例，都单行输出从农场到市场不重复的旅游方案数。



**HDU1423**

**题目描述（HDU1423）：**若存在1≤i1<i2<…<iN≤M，1≤j<N，使Sj=Aij且Sj<Sj+1，则称序列S1, S2, …, SN为A1, A2, …, AM的上升子序列。若z既是x的上升子序列，也是y的上升子序列，则称z是x和y的公共上升子序列。给定两个整数序列，求两者的最长公共上升子序列的长度。

**输入：**第1行包含测试用例数量T。每个测试用例都包含两个序列，对每个序列都用长度m（1≤m≤500）和m个整数ai（-231≤ai<231）描述。

**输出：**输出两个序列最长公共上升子序列的长度。



**HDU4991**

**题目描述（HDU4991）：**给定数字序列(A1, A2, …, An)的子序列是任意序列(Ai1, Ai2, …, Aik)，其中1≤i1<i2<ik≤n，若子序列是严格递增的，则称之为有序子序列。例如，序列(1, 7, 3, 5, 9, 4, 8)的有序子序列为(1, 7)、(3, 4, 8)等。给定数字序列，求解其长度为m的有序子序列的个数。

**输入：**输入包含多个测试用例，每个测试用例都包含两行。第1行包含两个整数n（1≤n≤10000）和m（1≤m≤100），n表示序列的长度，m表示需要查找的有序子序列的长度；第2行包含序列的n个整数元素，每个元素的范围都为0～987654321。

**输出：**对每个测试用例，都输出答案 % 123456789 。

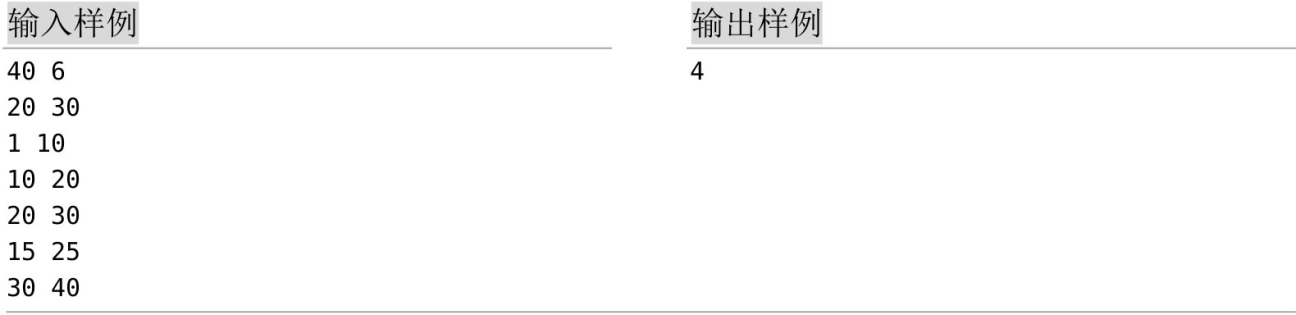


**POJ1769**

**题目描述（POJ1769）：**公司正在准备一个新的分拣硬件，称之为最大化器。最大化器的n个输入都从1到n，每个输入都代表一个整数。最大化器有一个输出，代表输入的最大值。最大化器的实现为排序器(i1, j1), …, 排序器(ik, jk)的流水线。每台排序器都有n个输入和n个输出。排序器(i, j)对输入i, i+1, …, j以非递减顺序输出，对其他输入原样输出。最后一个排序器的第n个输出是最大化器的输出。经过观察，去掉一些排序器之后，最大化器仍然可以产生正确的结果。给定排序器序列，求可以产生正确结果的最少排序器数量。

**输入：**输入的第1行包含两个整数n和m（2≤n≤50000，1≤m≤500000），分别表示输入的数量和流水线中的排序器数量。接下来的m行描述排序器的初始顺序，第k行包含第k个排序器的参数，即两个整数s和t（1≤s<t≤n），表示排序器排序的范围。

**输出：**单行输出可以产生正确结果的最少排序器数量。



**POJ2373**

**题目描述（POJ2373）：**约翰在山脊上安装了洒水装置。每个洒水器都必须沿着山脊安装，山脊的长度为L（1≤L≤1000000，L是偶数）。每个洒水器都沿山脊在两个方向上浇灌地面一段距离。每个洒水器的喷洒半径均为[a, b]（1≤a≤b≤1000）内的整数。约翰需要用一些洒水器来浇灌整个山脊，且浇灌范围不会超过山脊的末端。

约翰的n（1≤n≤1000）头牛都有一个特别喜欢的范围[s, e]（这些范围可能重叠）。对每头牛喜欢的范围都必须用一个洒水器，洒水器可能会（或不会）喷到指定的范围之外。找到浇灌整个山脊而不重叠所需的洒水器最小数量。

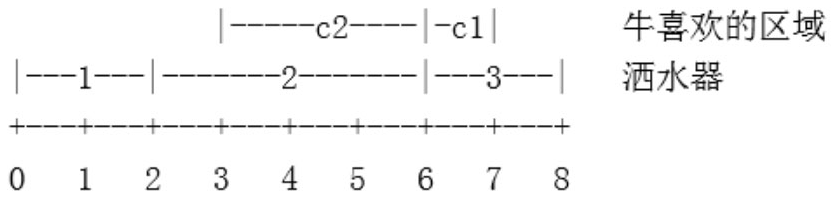
**输入：**第1行包含两个整数n和L。第2行包含两个整数a和b。第3..n+2行中的每一行都包含两个整数s和e（0≤s<e≤L），分别表示一头牛喜欢的范围的开始位置和结束位置。位置以到山脊起点的距离表示，所以在0..L范围内。

**输出：**单行输出洒水器最小数量。若无法设计洒水装置，则输出-1。



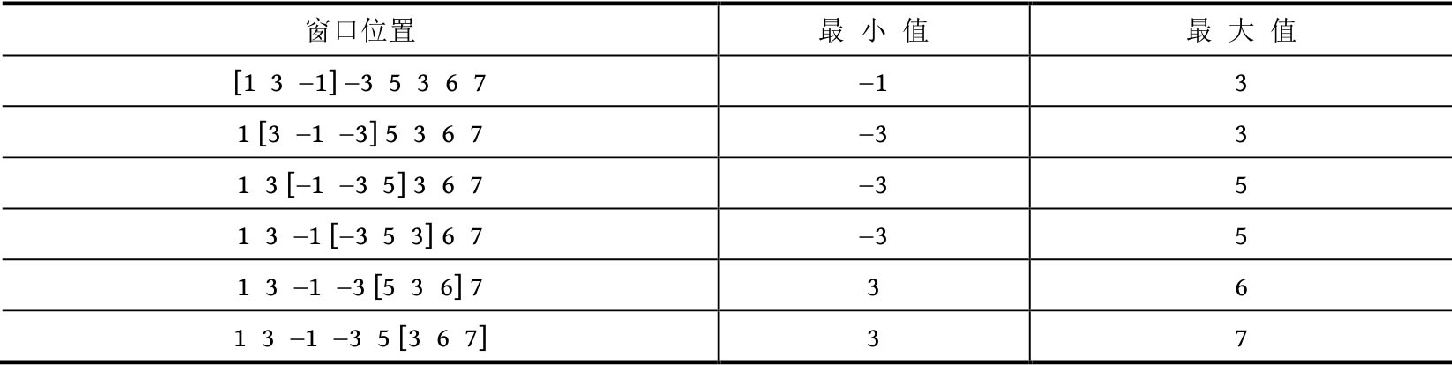
**提示：**根据输入样例，一共有两头牛，山脊的长度为8。洒水器的喷洒半径为[1,2]（即1或2）。一头牛喜欢3-6区域，另一头牛喜欢6-7区域。

我们需要3个洒水器：一个在1处，喷洒半径为1；一个在4处，喷洒半径为2；一个在7处，喷洒半径为1。第2个洒水器浇灌了第2头牛喜欢的三叶草。最后一个洒水器浇灌了第1头牛喜欢的三叶草，如下图所示。喷水器在2和6处不被视为重叠。



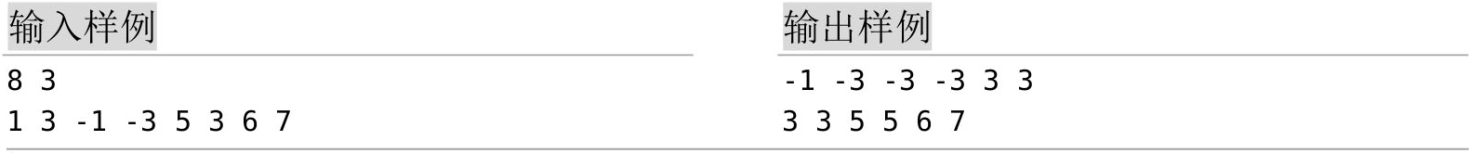
**POJ2823**

**题目描述（POJ2823）：**有n（n≤106）个元素的数组，以及一个大小为k的滑动窗口，将滑动窗口从数组的最左边移动到最右边，只可以在该窗口中看到k个数字，滑动窗口每次都向右移动一个位置，请确定滑动窗口在每个位置的最大值和最小值。下面是一个例子，数组是[1 3-1-3 5 3 6 7]，k是3。



**输入：**第1行包含整数n和k，表示元素个数和滑动窗口的长度；第2行包含n个整数。

**输出：**第1行从左到右分别输出每个窗口中的最小值，第2行输出最大值。



**POJ2373**

**题目描述（POJ2373）：**约翰在山脊上安装了洒水装置。每个洒水器都必须沿着山脊安装，山脊的长度为L（1≤L≤1000000，L是偶数）。每个洒水器都沿山脊在两个方向上浇灌地面一段距离。每个洒水器的喷洒半径均为[a, b]（1≤a≤b≤1000）内的整数。约翰需要用一些洒水器来浇灌整个山脊，且浇灌范围不会超过山脊的末端。

约翰的n（1≤n≤1000）头牛都有一个特别喜欢的范围[s, e]（这些范围可能重叠）。对每头牛喜欢的范围都必须用一个洒水器，洒水器可能会（或不会）喷到指定的范围之外。找到浇灌整个山脊而不重叠所需的洒水器最小数量。

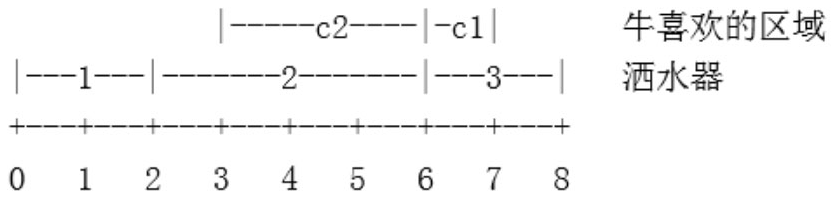
**输入：**第1行包含两个整数n和L。第2行包含两个整数a和b。第3..n+2行中的每一行都包含两个整数s和e（0≤s<e≤L），分别表示一头牛喜欢的范围的开始位置和结束位置。位置以到山脊起点的距离表示，所以在0..L范围内。

**输出：**单行输出洒水器最小数量。若无法设计洒水装置，则输出-1。



**提示：**根据输入样例，一共有两头牛，山脊的长度为8。洒水器的喷洒半径为[1,2]（即1或2）。一头牛喜欢3-6区域，另一头牛喜欢6-7区域。

我们需要3个洒水器：一个在1处，喷洒半径为1；一个在4处，喷洒半径为2；一个在7处，喷洒半径为1。第2个洒水器浇灌了第2头牛喜欢的三叶草。最后一个洒水器浇灌了第1头牛喜欢的三叶草，如下图所示。喷水器在2和6处不被视为重叠。



**HDU3401**

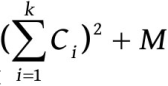
**题目描述（HDU3401）：**预测未来T天的股市。在第i天可以以APi的价格购买一只股票，或者以BPi的价格卖出一只股票。在第i天最多买ASi只股票，最多卖BSi只股票。两个交易日的间隔应大于W天。也就是说，假设在第i天交易（任何买卖股票都被视为交易），则下一个交易日必须是第（i+W+1）天或更晚。在任何时候都不可以拥有超过maxP只股票。第1天之前，小明已经有了无限多的钱，但没有股票，他想从股票市场尽可能多地赚钱。

**输入：**第1行是一个整数t，表示测试用例的数量。每个测试用例的第1行都是三个整数T、maxP、W，0≤W<T≤2000，1≤maxP≤2000。接下来的T行各有4个整数APi、BPi、ASi、BSi，1≤BPi≤APi≤1000，1≤ASi，BSi≤maxP。

**输出：**单行输出小明赚得最多的钱数。



**HDU3507**

**题目描述（HDU3507）：**小明要打印一篇有N个单词的文章。每个单词i都有一个打印成k本Ci。在一行中打印k个单词要花费的成本为 ，其中M是常量。他想知道打印文章的最小成本。

**输入：**输入包含多个测试用例。每个测试用例的第1行都包含两个数字N和M（0≤N≤500000，0≤M≤1000）。在接下来的2～N+1行中有N个数字，表示N个单词的打印成本。

**输出：**单行输出打印文章的最小成本。



**HDU4258**

**题目描述（HDU4258）：**准备建一条新的走道，走道上的某些点必须被覆盖，其他点是否被覆盖并不重要。有一个有趣的定价方案：为了覆盖从x点到y点的走道，将收费c+(x-y)2，其中c是常数。注意：x与y可能相等。给定走道沿线的点和常数c，覆盖走道的最低成本是多少？

**输入：**输入包含几个测试用例。每个测试用例都以两个整数n（1≤n≤106）和c（1≤c≤109）为开头，其中n是必须覆盖的点数，c是常数。以下n行中的每一行都包含一个整数p（1≤p≤109），表示走道上必须覆盖的一个点，这些点从小到大排列。以一行两个0结尾。

**输出：**对每个测试用例都单行输出覆盖所有指定点的最小成本，答案为64位有符号整数。



**POJ1180**

**题目描述（POJ1180）：**有N个作业要在一台机器上处理，编号为1～N。作业序列不得改变，可以划分为一个或多个批次，其中每个批次都由序列中的连续作业组成。处理从时间0和第1批作业开始，一批一批地处理。批次中的作业在机器上依次处理，处理完一个批次中的所有作业后，机器立即输出该批次中所有作业的结果。作业j的输出时间是包含j的批处理完成的时间。

在每个批次启动机器都需要S时间。对每个作业i，其处理时间都为Ti，费用系数都为Fi。若批处理包含作业x, x+1, …, x+k，从时间t开始，则该批次中每个作业的输出时间都为t+S+(Tx+Tx+1+…+Tx+k)。若作业i的输出时间为Oi，则其成本为Oi×Fi。

假设有5个作业，启动时间S=1，(T1, T2, T3, T4, T5)=(1, 3, 4, 2, 1)，(F1, F2, F3, F4, F5)=(3, 2, 3, 3, 4)。若将作业分成三批{1, 2}、{3}、{4, 5}，则输出时间为(5, 5, 10, 14, 14)，成本为(15, 10, 30, 42, 56)，总成本是所有作业成本的总和153。

**输入：**第1行包含作业数N（1≤N≤10000），第2行包含批次启动时间整数S（0≤S≤50）。以下N行，每行都包含两个整数，即作业的处理时间Ti和费用系数Fi（1≤Ti，Fi≤100）。

**输出：**单行输出批处理作业的最小总成本。



**HDU3480**

**题目描述（HDU3480）：**S是一个整数集合。若MIN是S中的最小整数，MAX是S中的最大整数，则将S集合的价值定义为(MAX-MIN)2。给定整数集合S，找出S的M个子集S1, S2, …, SM，满足S1∪S2∪…∪SM=S，且每个子集的总价值都是最小的。

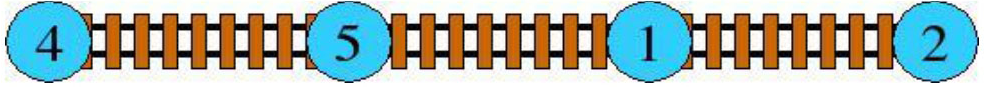
**输入：**输入包含多个测试用例。第1行包含整数T，表示测试用例的数量。每个测试用例的第1行都包含两个整数N（N≤10000）和M（M≤5000）。N是S中的元素个数（可以重复），M是子集数量。在下一行中包含集合S中的N个整数。

**输出：**对每个测试用例，都单行输出最小的总价值。



**HDU2829**

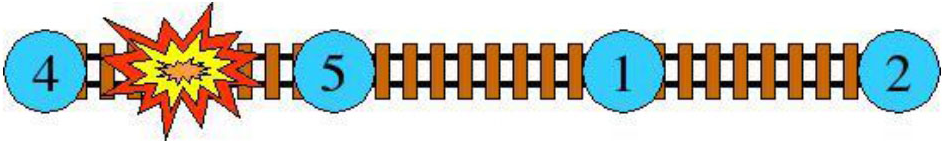
**题目描述（HDU2829）**：某国情报部门给每个车站都分配了一个战略重要性：从1到100的整数。单个车站自身没有价值，与其他车站相连才有价值。整个铁路的战略价值是将铁路线直接或间接连接的每对车站的战略价值的乘积相加来计算的。假设铁路如下图所示，则其战略价值为4×5+4×1+4×2+5×1+5×2+1×2=49。



假设只有一次攻击，不可以攻击车站，只能攻击两个车站之间的铁路线。若攻击了中间的这条铁路线，则剩余铁路的战略价值为4×5+1×2=22。



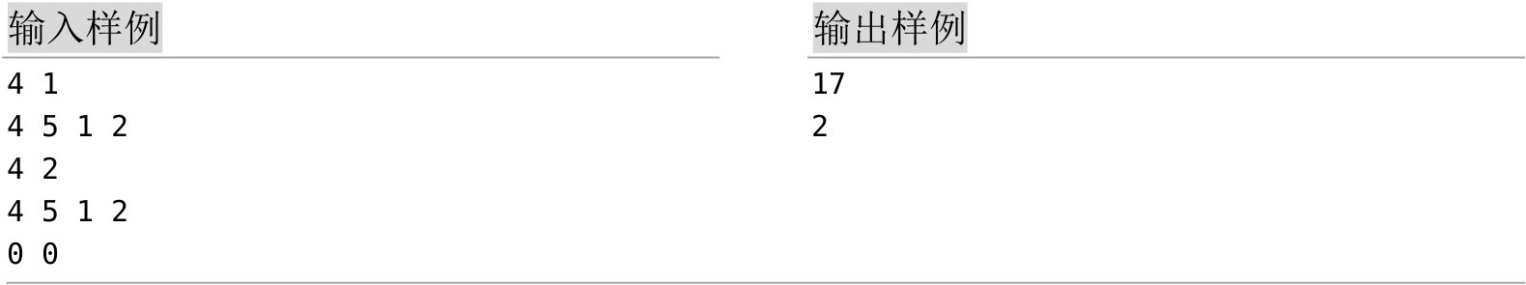
但是，假设攻击了4和5之间的铁路线，则剩余铁路的战略价值为5×1+5×2+1×2=17。



给出一条铁路的描述和可以执行的攻击次数，找出可以实现的最小战略价值。

**输入：**输入包含几个测试用例。每个测试用例都以两个整数n（1≤n≤1000）和m（0≤m<n）为开头。n是铁路上的站点数量，m是攻击数量。下一行是n个整数，范围为1～100，依次表示每个站点的战略价值。输入以两个0结束。

**输出：**对每个测试用例，都单行输出通过攻击可以实现的铁路的最小战略值。



**HDU3480**

**题目描述（HDU3480）：**S是一个整数集合。若MIN是S中的最小整数，MAX是S中的最大整数，则将S集合的价值定义为(MAX-MIN)2。给定整数集合S，找出S的M个子集S1, S2, …, SM，满足S1∪S2∪…∪SM=S，且每个子集的总价值都是最小的。

**输入：**输入包含多个测试用例。第1行包含整数T，表示测试用例的数量。每个测试用例的第1行都包含两个整数N（N≤10000）和M（M≤5000）。N是S中的元素个数（可以重复），M是子集数量。在下一行中包含集合S中的N个整数。

**输出：**对每个测试用例，都单行输出最小的总价值。

