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）。

输入样例	输出样例
1 5 10 1 2 3 4 5 5 4 3 2 1	14

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.”。

输入样例	输出样例
3 10 110 2 1 1 30 50 10 110 2 1 1 50 30 1 6 2 10 3 20 4	The minimum amount of money in the piggy-bank is 60. The minimum amount of money in the piggy-bank is 100. 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 $a_1, a_2, a_3, ..., a_n$ 和该面值的数量 $c_1, c_2, c_3, ..., c_n$，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 $a_1, a_2, a_3, ..., a_n$、$c_1, c_2, c_3, ..., c_n$（$1≤a_i≤100000$，$1≤c_i≤1000$）。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。

输入样例	输出样例
3 10 1 2 4 2 1 1 2 5 1 4 2 1 0 0	8 4

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。

输入样例	输出样例
2 2 1 2 1 3 2 2 2 1 2 1 2 3 3 2 1 3 2 1 0 0	3 4 6

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 v_1, v_2, ..., v_N（1≤v_i≤120）。约翰有 c_1 个面值为 v_1 的硬币，c_2 个面值 v_2 的硬币, ..., c_N 个面值 v_N（0≤c_i≤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 v_1, v_2, ..., v_N，表示硬币的面值。第 3 行有 N 个整数 c_1, c_2, ..., c_N，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。

输入样例	输出样例
3 70 5 25 50 5 2 2	3

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？

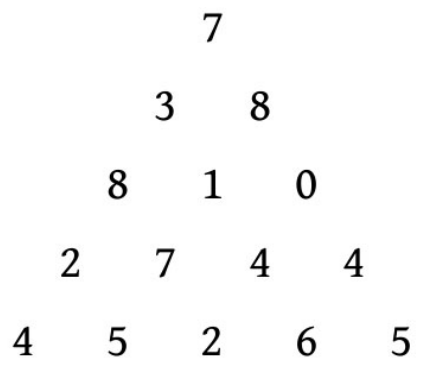
输入：第 1 行包含一个整数 N，表示测试用例的个数。然后是 N 行数据，每行都包含一个整数 M（1≤M≤40），表示楼梯的级数。

输出：对每个测试实例都输出不同走法的数量。

输入样例	输出样例
2 2 3	1 2

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。

输入样例	输出样例
5 7 3 8 8 1 0 2 7 4 4 4 5 2 6 5	30

POJ2533

题目描述（POJ2533）：若一个序列满足 $a_1<a_2<...<a_n$，则该序列是有序（上升）的。设给定数字序列(a_1, a_2, ..., a_n)的子序列为任意序列(a_{i1}, a_{i2}, ..., a_{ik})，其中 $1\le i_1<i_2<...<i_k\le n$，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。

输入样例	输出样例
7 1 7 3 5 9 4 8	4

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 $x=<x_1, x_2, ..., x_m>$及另一个序列 $z=<z_1, z_2, ..., z_k>$，若 x 的索引存在严格递增的序列$<i_1, i_2, ..., i_k>$，则对所有 j=1, 2, ..., k 及 $x_{ij}=z_j$，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。

输入样例	输出样例
abcfbc abfcab programming contest abcd mnp	4 2 0

HDU1003

题目描述（HDU1003）：给定一个序列 $a_1, a_2, a_3, ..., a_n$，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。

输入样例	输出样例
2 5 6 -1 5 4 -7 7 0 6 -1 1 -6 7 -5	Case 1: 14 1 4 Case 2: 7 1 6

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。

输入样例	输出样例
3 4 abcb a 1000 1100 b 350 700 c 200 800	900

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 $a_1a_2...a_n$，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 $a_{i1}a_{i2}...a_{im}$ 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。

输入样例	输出样例
(((())) ()()() ([]]))[)(([][][) end	6 6 4 0 6

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。

输入样例	输出样例
8 5 2 4 7 6 1 3 9	105

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。

输入样例	输出样例
6 10 1 50 50 20 5	3650

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。

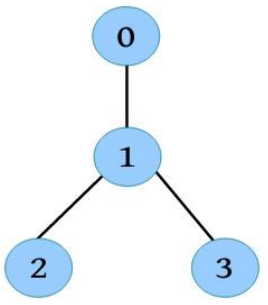
输入：输入包含多个测试用例。每个测试用例都以整数 n（1≤n≤200）开始，表示员工的数量。下一行只包含大老板的名字。在接下来的 n-1 行中，每一行都包含员工的名字及其老板的名字。所有名字都是由至少一个和最多 100 个字母组成的字符串，以空格隔开。每个测试用例的最后一行都包含一个 0。

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。

输入样例	输出样例
6 Jason Jack Jason Joe Jack Jill Jason John Jack Jim Jill 2 Ming Cho Ming 0	4 Yes 1 No

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。

输入样例	输出样例
4 0:(1) 1 1:(2) 2 3 2:(0) 3:(0) 5 3:(3) 1 4 2 1:(1) 0 2:(0) 0:(0) 4:(0)	1 2

对输入样例 1 的数据解释如下所示：

```
4 //节点数为 4
0:(1) 1 //节点 0 连接 1 条道路，道路的另一端节点为 1，即 0-1 有 1 条道路
1:(2) 2 3 //节点 1 连接两条道路，道路的另一端节点分别为 2、3，即 1-2、1-3 分别有 1 条道路
2:(0) //节点 2 连接 0 条道路
3:(0) //节点 3 连接 0 条道路
```

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。

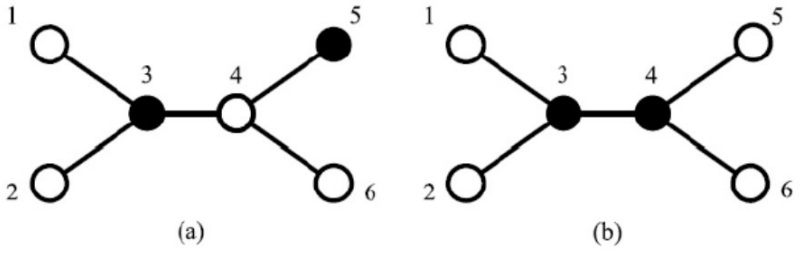
输入：输入包含几个测试用例。每个测试用例都包括两行，第 1 行包含两个整数 n 和 T（1≤n≤105，1≤T≤100），n 表示公司的员工人数（不包括公司大老板），T 是上面描述的参数。每个员工的编号都为 1～n，大老板编号为 0；第 2 行包含整数列表，列表中的位置 i （从 1 开始）为整数 b_i（$0 \le b_i \le i-1$），表示员工 i 的直接上司的编号。在最后一个测试用例后面包含两个 0。

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。

输入样例	输出样例
3 100 0 0 0 3 50 0 0 0 14 60 0 0 1 1 2 2 2 5 7 5 7 5 7 5 0 0	3 2 5

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。

输入样例	输出样例
6 1 3 2 3 3 4 4 5 4 6 0 2 1 2 -1	2 1

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。

输入样例	输出样例
3 2 0 1 0 2 0 3 7 4 2 2 0 1 0 4 2 1 7 1 7 6 2 2 0 0	5 13

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。

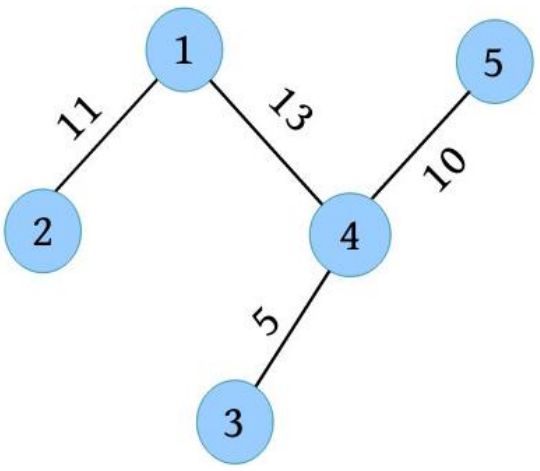
输入：输入包含几个测试用例。每个测试用例都包含 3 部分。第 1 部分包含两个数字 n、k（1≤n≤100，0≤k≤200），分别表示节点数和所走的步数，节点编号为 1～n。第 2 部分包含 n 个整数（所有整数均非负且不大于 1000），第 i 个整数表示节点 i 的苹果数量。第 3 部分包含 n-1 行，每行都包含两个整数 a、b，表示节点 a 和节点 b 是相邻的。

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。

输入样例	输出样例
2 1 0 11 1 2 3 2 0 1 2 1 2 1 3	11 2

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

路径和流量：1→2　　11
　　1→4→3　　5
　　1→4→5　　8（因为 1→4 的容量是 13）

（2）a(2)=5+6=11

路径和流量：2→1→4→3　　5
　　2→1→4→5　　6

（3）a(3)=5

路径和流量：3→4→5　　5

（4）a(4)=11+5+10=26

路径和流量：4→1→2　　11
　　4→3　　5
　　4→5　　10

（5）a(5)=10

路径和流量：5→4→1→2　　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。

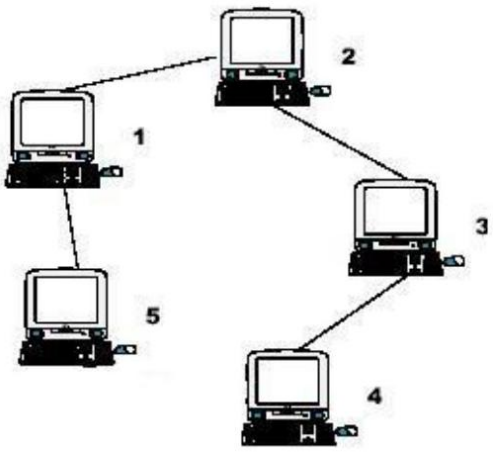
输出：对每个测试用例，都单行输出树的累积度。

输入样例	输出样例
1 5 1 2 11 1 4 13 3 4 5 4 5 10	26

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 s_i（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 s_1=3。计算机 2 距离 5 和 4 最远，最远电缆长度 s_2=2。计算机 3 距离 5 最远，最远电缆长度 s_3=3。同理，得到 s_4=4、s_5=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 10^9。

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。

输入样例	输出样例
5 1 1 2 1 3 1 1 1	3 2 3 4 4

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。

输入样例	输出样例
1 100 0 0	80

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（$1 \le n \le 2^{63}-1$）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。

输入样例	输出样例
3 1 50 500	0 1 15

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（$1 \le start < finish \le 2000000000$）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。

输入样例	输出样例
2 12	6

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。

输入样例	输出样例
1 10 44 497 346 542 1199 1748 1496 1403 1004 503 1714 190 1317 854 1976 494 1001 1960 0 0	1 2 1 1 1 1 1 1 1 1 85 185 185 185 190 96 96 96 95 93 40 40 40 93 136 82 40 40 40 40 115 666 215 215 214 205 205 154 105 106 16 113 19 20 114 20 20 19 19 16 107 105 100 101 101 197 200 200 200 200 413 1133 503 503 503 502 502 417 402 412 196 512 186 104 87 93 97 97 142 196 398 1375 398 398 405 499 499 495 488 471 294 1256 296 296 296 296 287 286 286 247

HDU4734

题目描述（HDU4734）：十进制数 x 包含 n 个数字（a_n，a_{n-1}，a_{n-2}，…，a_2，a_1），它的权值被定义为 $F(x)=a_n\times2^{n-1}+a_{n-1}\times2^{n-2}+\ldots+a_2\times2+a_1\times1$。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<10^9）。

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。

输入样例	输出样例
3 0 100 1 10 5 100	Case #1: 1 Case #2: 2 Case #3: 13

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。

输入样例	输出样例
3 0 1 10 10 1 0 1 2 10 1 0 10 10 2 10 0 0	8

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。

输入样例	输出样例
2 1 1 2 100 3 2 1 2 40 2 3 50 3 3 1 2 3 1 3 4 2 3 10	100 90 7

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。

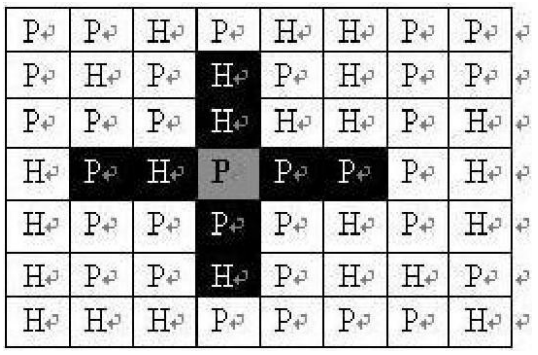
输入样例	输出样例
2 3 1 1 1 0 1 0	9

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。

```
1  2  3
   4
```

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。

输入样例	输出样例
5 4 PHPP PPHH PPPP PHPP PHHP	6

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）。

```
n m p a b
t1 t2 ... tn
x1 y1 z1
x2 y2 z2
...
xp yp zp
```

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，t_i 是第 i 张车票的马数（1≤i≤n，1≤t_i≤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 x_i 和 y_i 连接起来，并有距离 z_i（1≤i≤p，1≤z_i≤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。

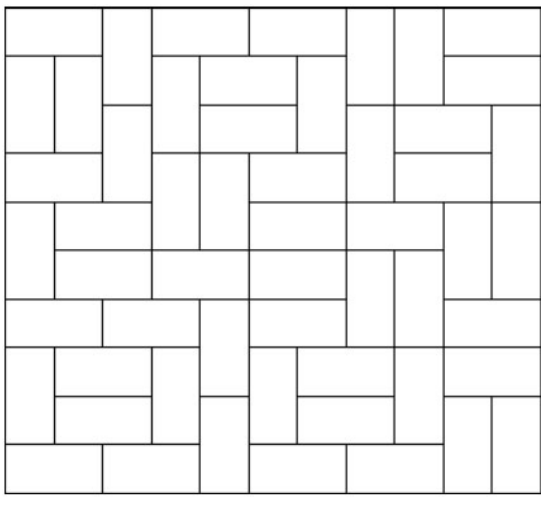
输出：若旅行者可以到达目的地，则输出出发点到目的地的最短时间。答案的误差不应大于 0.001。在满足上述精度条件的前提下，可以输出小数点后的任意位数。若无法到达目的地，则输出“Impossible”。

输入样例	输出样例
3 4 3 1 4 3 1 2 1 2 10 2 3 30 3 4 20 2 4 4 2 1 3 1 2 3 3 1 3 3 4 1 2 4 2 5 2 4 3 4 1 5 5 1 2 10 2 3 10 3 4 10 1 2 0 1 2 1 8 5 10 1 5 2 7 1 8 4 5 6 3 1 2 5 2 3 4 3 4 7 4 5 3 1 3 25 2 4 23 3 5 22 1 4 45 2 5 51 1 5 99 0 0 0 0 0	30.000 3.667 Impossible Impossible 2.856

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。

输入样例	输出样例
1 2 1 3 1 4 2 2 2 3 2 4 2 11 4 11 0 0	1 0 1 2 3 5 144 51205

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。

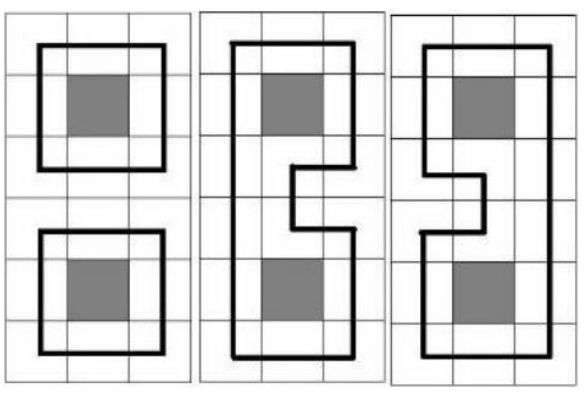
输入：包含多个测试实例，每个测试实例都包含一个整数 n 和 n×n 个非负数（n≤20）。

输出：对每个测试实例都单行输出可能取得的最大和值。

输入样例	输出样例
3 75 15 21 75 15 28 34 70 5	188

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）。



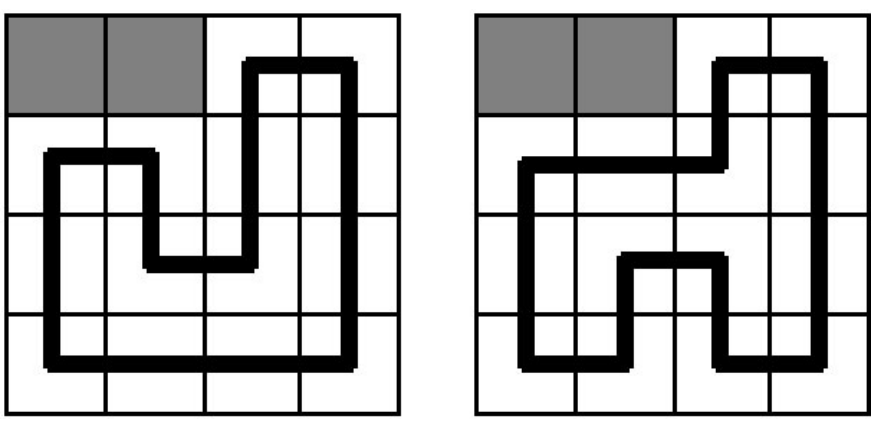
输入：输入的第 1 行是测试用例数 T（T≤10）。每个测试用例的第 1 行都包含整数 n 和 m（1≤n, m≤11）。在接下来的 n 行中，每行都包含 m 个数字（0 或 1），0 表示没有树的单元格，1 表示只有一棵树的单元格。

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 2^{63}-1。

输入样例	输出样例
2 6 3 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 2 4 1 1 1 1 1 1 1 1	Case 1: There are 3 ways to eat the trees. Case 2: There are 2 ways to eat the trees.

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 2^{63}-1。

输入样例	输出样例
4 4 **.. .... 4 4	2 6

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。

输入样例	输出样例
2 2 .. .. 2 3 #.. ... 3 4 .... 0 0	1 1 4

HDU1423

题目描述（HDU1423）：若存在 $1\le i_1<i_2<\ldots<i_N\le M$，$1\le j<N$，使 $S_j=A_{ij}$ 且 $S_j<S_{j+1}$，则称序列 $S_1, S_2, \ldots, S_N$ 为 $A_1, A_2, \ldots, A_M$ 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（$1\le m\le 500$）和 m 个整数 ai（$-2^{31}\le a_i<2^{31}$）描述。

输出：输出两个序列最长公共上升子序列的长度。

输入样例	输出样例
1 5 1 4 2 5 -12 4 -12 1 2 4	2

HDU4991

题目描述（HDU4991）：给定数字序列$(A_1, A_2, \ldots, A_n)$的子序列是任意序列$(A_{i1}, A_{i2}, \ldots, A_{ik})$，其中 $1\le i_1<i_2<i_k\le n$，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（$1\le n\le 10000$）和 m（$1\le m\le 100$），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 。

输入样例	输出样例
3 2 1 1 2 7 3 1 7 3 5 9 4 8	2 12

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$(i_1, j_1), \ldots,$ 排序器(i_k, j_k)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（$2\le n\le 50000$，$1\le m\le 500000$），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（$1\le s<t\le n$），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。

输入样例	输出样例
40 6 20 30 1 10 10 20 20 30 15 25 30 40	4

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。

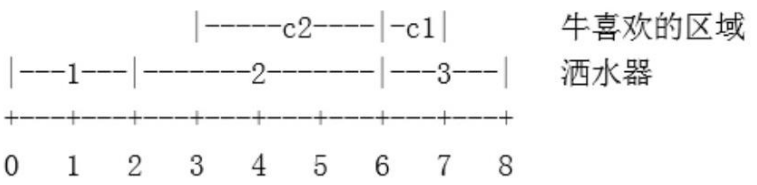
输入：第 1 行包含两个整数 n 和 L。第 2 行包含两个整数 a 和 b。第 3..n+2 行中的每一行都包含两个整数 s 和 e（0≤s<e≤L），分别表示一头牛喜欢的范围的开始位置和结束位置。位置以到山脊起点的距离表示，所以在 0..L 范围内。

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。

输入样例	输出样例
2 8 1 2 6 7 3 6	3

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（$n \le 10^6$）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。

窗口位置	最小值	最大值
[1 3 −1] −3 5 3 6 7	−1	3
1 [3 −1 −3] 5 3 6 7	−3	3
1 3 [−1 −3 5] 3 6 7	−3	5
1 3 −1 [−3 5 3] 6 7	−3	5
1 3 −1 −3 [5 3 6] 7	3	6
1 3 −1 −3 5 [3 6 7]	3	7

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。

输入样例	输出样例
8 3 1 3 -1 -3 5 3 6 7	-1 -3 -3 -3 3 3 3 3 5 5 6 7

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。

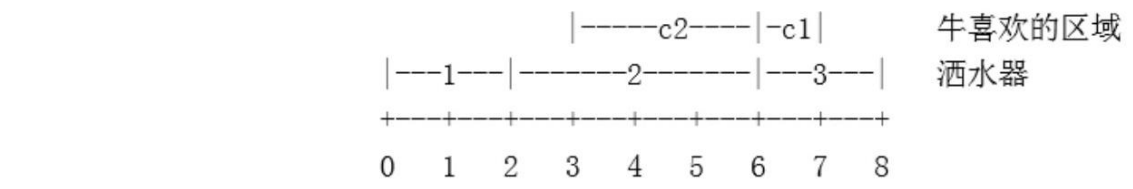
输入：第 1 行包含两个整数 n 和 L。第 2 行包含两个整数 a 和 b。第 3..n+2 行中的每一行都包含两个整数 s 和 e（0≤s<e≤L），分别表示一头牛喜欢的范围的开始位置和结束位置。位置以到山脊起点的距离表示，所以在 0..L 范围内。

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。

输入样例	输出样例
2 8 1 2 6 7 3 6	3

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 AP_i 的价格购买一只股票，或者以 BP_i 的价格卖出一只股票。在第 i 天最多买 AS_i 只股票，最多卖 BS_i 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 AP_i、BP_i、AS_i、BS_i，$1≤BP_i≤AP_i≤1000$，$1≤AS_i$，$BS_i≤maxP$。

输出：单行输出小明赚得最多的钱数。

输入样例	输出样例
1 5 2 0 2 1 1 1 2 1 1 1 3 2 1 1 4 3 1 1 5 4 1 1	3

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 C_i。在一行中打印 k 个单词要花费的成本为 $(\sum_{i=1}^{k} C_i)^2 + M$，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。

输入样例	输出样例
5 5 5 9 5 7 5	230

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（$1 \le n \le 10^6$）和 c（$1 \le c \le 10^9$）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（$1 \le p \le 10^9$），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。

输入样例	输出样例
10 5000 1 23 45 67 101 124 560 789 990 1019 0 0	30726

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 T_i，费用系数都为 F_i。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 $t+S+(T_x+T_{x+1}+…+T_{x+k})$。若作业 i 的输出时间为 O_i，则其成本为 $O_i×F_i$。

假设有 5 个作业，启动时间 S=1，$(T_1, T_2, T_3, T_4, T_5)=(1, 3, 4, 2, 1)$，$(F_1, F_2, F_3, F_4, F_5)=(3, 2, 3, 3, 4)$。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 T_i 和费用系数 F_i（$1≤T_i, F_i≤100$）。

输出：单行输出批处理作业的最小总成本。

输入样例	输出样例
5 1 1 3 3 2 4 3 2 3 1 4	153

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 $S_1, S_2, …, S_M$，满足 $S_1∪S_2∪…∪S_M=S$，且每个子集的总价值都是最小的。

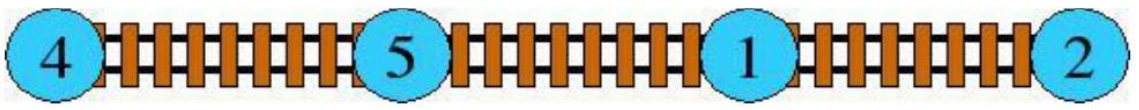
输入：输入包含多个测试用例。第 1 行包含整数 T，表示测试用例的数量。每个测试用例的第 1 行都包含两个整数 N（N≤10000）和 M（M≤5000）。N 是 S 中的元素个数（可以重复），M 是子集数量。在下一行中包含集合 S 中的 N 个整数。

输出：对每个测试用例，都单行输出最小的总价值。

输入样例	输出样例
2 3 2 1 2 4 4 2 4 7 10 1	Case 1: 1 Case 2: 18

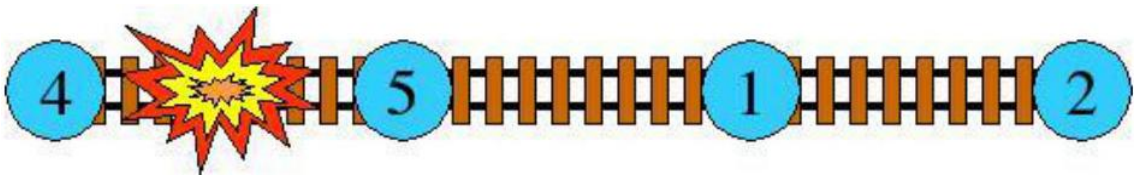
HDU2829

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。

输入样例	输出样例
4 1 4 5 1 2 4 2 4 5 1 2 0 0	17 2

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 $S_1, S_2, \ldots, S_M$，满足 $S_1 \cup S_2 \cup \ldots \cup S_M = S$，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。

输入样例	输出样例
2 3 2 1 2 4 4 2 4 7 10 1	Case 1: 1 Case 2: 18