# 最大子段和

给定由n个整数（包含负整数）组成的序列a[1], a[2], ..., a[n]，求该序列子段和的最大值。当所有整数均为负值时定义其最大子段和为0。

例： -2,11,-4,13,-5,-2 MSS=20

# 最长公共子序列（LCS）

问题描述：

求两个字符串的最长公共子串的长度。两个字符串X和Z的公共子串是指X中按序取一些字符构成的子串与在Z中按同样方法得到的一个子串相同。

例如：X=“abcdefghi”

Z=“xyazdwf”

则”a” “d” “f” “ad” “df” “adf”都是它们的公共子串，而最长公共子串的长度为3。

输入样例：

abcfbc abfcab

programming contest

abcd mnp

输出样例：

4

2

0

思考：

（1）如果要求最长公共子序列的个数，该如何处理？

（2）如果要输出一个最长公共子序列，又该如何处理？

（3）如果要输出所有的最长公共子序列，又该如何处理？

# 最长有序子序列(LIS)

找出由n个元素组成的序列的最长有序子序列长度及其中一个最长有序子序列（这里有序指非递减顺序，且不要求子序列连续）。

输入示例：

9

1 4 7 2 5 8 3 6 9

思考：

（1）如何求其中一个最长有序子序列？

（2）如何求所有的最长有序子序列以及最长有序子序列的个数？

（3）对于原序列中的元素，哪些一定出现在最长有序子序列中，哪些可能出现在最长有序子序列中？哪些一定不会出现在最长有序子序列中？

# Mysterious Present

来源：Codeforces 4D

time limit per test 1 second

memory limit per test 64 megabytes

Peter decided to wish happy birthday to his friend from Australia and send him a card. To make his present more mysterious, he decided to make a *chain*. Chain here is such a sequence of envelopes *A* = {*a*1,  *a*2,  ...,  *an*}, where the width and the height of the *i*-th envelope is strictly higher than the width and the height of the (*i*  -  1)-th envelope respectively. Chain size is the number of envelopes in the chain.

Peter wants to make the chain of the maximum size from the envelopes he has, the chain should be such, that he'll be able to put a card into it. The card fits into the chain if its width and height is lower than the width and the height of the smallest envelope in the chain respectively. It's forbidden to turn the card and the envelopes.

Peter has very many envelopes and very little time, this hard task is entrusted to you.

**Input**

The first line contains integers *n*, *w*, *h* (1  ≤ *n* ≤ 5000, 1 ≤ *w*,  *h*  ≤ 106) — amount of envelopes Peter has, the card width and height respectively. Then there follow *n* lines, each of them contains two integer numbers *wi* and *hi* — width and height of the *i*-th envelope (1 ≤ *wi*,  *hi* ≤ 106).

**Output**

In the first line print the maximum chain size. In the second line print the numbers of the envelopes (separated by space), forming the required chain, starting with the number of the smallest envelope. Remember, please, that the card should fit into the smallest envelope. If the chain of maximum size is not unique, print any of the answers.

If the card does not fit into any of the envelopes, print number 0 in the single line.

**Examples**

**Input**

**Copy**

2 1 1  
2 2  
2 2

**Output**

**Copy**

1  
1

**Input**

**Copy**

3 3 3  
5 4  
12 11  
9 8

**Output**

**Copy**

3  
1 3 2

# LCIS

来源：Codeforces 10D

time limit per test 1 second

memory limit per test 256 megabytes

*This problem differs from one which was on the online contest.*

The sequence *a*1, *a*2, ..., *an* is called increasing, if *ai* < *ai*+ 1 for *i* < *n*.

The sequence *s*1, *s*2, ..., *sk* is called the subsequence of the sequence *a*1, *a*2, ..., *an*, if there exist such a set of indexes 1 ≤ *i*1 < *i*2 < ... < *ik* ≤ *n* that *aij* = *sj*. In other words, the sequence *s* can be derived from the sequence *a* by crossing out some elements.

You are given two sequences of integer numbers. You are to find their longest common increasing subsequence, i.e. an increasing sequence of maximum length that is the subsequence of both sequences.

**Input**

The first line contains an integer *n* (1 ≤ *n* ≤ 500) — the length of the first sequence. The second line contains *n* space-separated integers from the range [0, 109] — elements of the first sequence. The third line contains an integer *m* (1 ≤ *m* ≤ 500) — the length of the second sequence. The fourth line contains *m* space-separated integers from the range [0, 109] — elements of the second sequence.

**Output**

In the first line output *k* — the length of the longest common increasing subsequence. In the second line output the subsequence itself. Separate the elements with a space. If there are several solutions, output any.

**Examples**

**Input**

**Copy**

7  
2 3 1 6 5 4 6  
4  
1 3 5 6

**Output**

**Copy**

3  
3 5 6

**Input**

**Copy**

5  
1 2 0 2 1  
3  
1 0 1

**Output**

**Copy**

2  
0 1

# 直线的交点数

来源：HDU 1466

问题描述：

平面上有n条直线，且无三线共点，问这些直线能有多少种不同交点数。

输入：n（n<=20）

输出：每个测试实例对应一行输出，从小到大列出所有相交方案，其中每个数为可能的交点数。

样例输入

4

样例输出

0 3 4 5 6

# 0-1背包

问题描述：

有N件物品和一个容量为V的背包。第i件物品的费用是c[i]，价值是w[i]。求解将哪些物品装入背包可使价值总和最大。

输入样例：

10 5 //V N

2 6 // c[i] w[i]

2 3

6 5

5 4

4 6

样例输出：

15

思考：

获得最大价值应该取哪些物品？

空间优化：

可以将空间复杂度却可以优化到O(V)。

引申：

恰好装满背包的最优解。

# 完全背包

有N种物品和一个容量为V的背包，每种物品都有无限件可用。第i种物品的费用是c[i]，价值是w[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

输入样例：

15 5 //V N

2 3 // c[i] w[i]

3 4

5 7

5 4

4 3

样例输出：

22

方法一：转化为0-1背包

将一件物品变为多件物品；用二进制思想。

方法二：直接动态规划

优化1：可以去掉一些不可能取的物品。

优化2：用类似与0-1背包的方法进行空间优化。

# 多重背包

有N种物品和一个容量为V的背包。第i种物品的费用是c[i]，价值是w[i]，数量为num[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

输入样例：

15 5 // V, N

2 3 2 // c[i] w[i] num[i]

3 4 3

5 7 4

5 4 5

4 3 3

输出样例：

21

输入样例：

15 5

5 12 2

4 3 4

7 10 1

2 3 5

6 6 3

输出样例：

30

# 分组背包

有N件物品和一个容量为V的背包。第i件物品的费用是c[i]，价值是w[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费用总和不超过背包容量，且价值总和最大。

输入样例：

45 4 // V N

10 10 1 // c[i] w[i] 组别，组数<100

10 5 1

5 20 2

50 400 2

输出样例：

30

# Checkout Assistant

来源：Codeforces 19B

time limit per test1 second

memory limit per test256 megabytes

Bob came to a cash & carry store, put n items into his trolley, and went to the checkout counter to pay. Each item is described by its price ci and time ti in seconds that a checkout assistant spends on this item. While the checkout assistant is occupied with some item, Bob can steal some other items from his trolley. To steal one item Bob needs exactly 1 second. What is the minimum amount of money that Bob will have to pay to the checkout assistant? Remember, please, that it is Bob, who determines the order of items for the checkout assistant.

Input

The first input line contains number n (1 ≤ n ≤ 2000). In each of the following n lines each item is described by a pair of numbers ti, ci (0 ≤ ti ≤ 2000, 1 ≤ ci ≤ 109). If ti is 0, Bob won't be able to steal anything, while the checkout assistant is occupied with item i.

Output

Output one number — answer to the problem: what is the minimum amount of money that Bob will have to pay.

Examples

Input

4

2 10

0 20

1 5

1 3

Output

8

Input

3

0 1

0 10

0 100

Output

111

# 最长回文子串

来源：Leetcode 5

给定一个字符串 s，找到 s 中最长的回文子串。你可以假设 s 的最大长度为 1000。

示例 1：

输入: "babad"

输出: "bab"

注意: "aba" 也是一个有效答案。

示例 2：

输入: "cbbd"

输出: "bb"

# 整数划分（插入乘号）

给出两个整数 n , m ,要求在 n 中加入m - 1 个乘号，将n分成m段，求出这m段的最大乘积。

输入：

第一行是一个整数T，表示有T组测试数据

接下来T行，每行有两个正整数 n，m ( 1<= n < 10^19, 0 < m <= n的位数)；

输出：

输出每组测试样例结果为一个整数占一行

样例输入

2

111 2

1111 2

样例输出

11

121

# Brackets

题目来源：poj2955

We give the following inductive definition of a “regular brackets” sequence:

* the empty sequence is a regular brackets sequence,
* if *s* is a regular brackets sequence, then (*s*) and [*s*] are regular brackets sequences, and
* if *a* and *b* are regular brackets sequences, then *ab* is a regular brackets sequence.
* no other sequence is a regular brackets sequence

For instance, all of the following character sequences are regular brackets sequences:

(), [], (()), ()[], ()[()]

while the following character sequences are not:

(, ], )(, ([)], ([(]

Given a brackets sequence of characters *a*1*a*2 … *an*, your goal is to find the length of the longest regular brackets sequence that is a subsequence of *s*. That is, you wish to find the largest *m* such that for indices *i*1, *i*2, …, *im* where 1 ≤ *i*1 < *i*2 < … < *im* ≤ *n*, *ai*1*ai*2 … *aim* is a regular brackets sequence.

Given the initial sequence ([([]])], the longest regular brackets subsequence is [([])].。

Input

The input test file will contain multiple test cases. Each input test case consists of a single line containing only the characters (, ), [, and ]; each input test will have length between 1 and 100, inclusive. The end-of-file is marked by a line containing the word “end” and should not be processed.

Output

For each input case, the program should print the length of the longest possible regular brackets subsequence on a single line.

Sample Input

((()))

()()()

([]])

)[)(

([][][)

end

Sample Output

6

6

4

0

6

引申：

给你一个字符串，里面只包含"(",")","[","]"四种符号，请问你需要至少添加多少个括号才能使这些括号匹配起来。

# 矩阵连乘

来源：FZU - 1061

给定n个矩阵{A1,A2,...,An}，考察这n个矩阵的连乘积A1A2...An。由于矩阵乘法满足结合律，故计算矩阵的连乘积可以有许多不同的计算次序，这种计算次序可以用加括号的方式来确定。

矩阵连乘积的计算次序与其计算量有密切关系。例如，考察计算3个矩阵{A1,A2,A3}连乘积的例子。设这3个矩阵的维数分别为10\*100，100\*5，和5\*50。若按(A1A2)A3计算，3个矩阵连乘积需要的数乘次数为10\*100\*5+10\*5\*50 = 7500。若按A1(A2A3)计算，则总共需要100\*5\*50+10\*100\*50 = 75000次数乘。

现在你的任务是对于一个确定的矩阵连乘方案，计算其需要的数乘次数。

输入：

输入数据由多组数据组成。每组数据格式如下：

第一行是一个整数n (1≤n≤26)，表示矩阵的个数。

接下来n行，每行有一个大写字母，表示矩阵的名字，后面有两个整数a,b，分别表示该矩阵的行数和列数，其中1<a,b<100。

第n+1行是一个矩阵连乘的表达式，由括号与大写字母组成，没有乘号与多余的空格。如果表达式中没有括号则按照从左到右的顺序计算，输入的括号保证能够配对。

输出：

对于每组数据，输出仅一行包含一个整数，即将该矩阵连乘方案需要的数乘次数。如果运算过程中出现不满足矩阵乘法法则的情况（即左矩阵列数与右矩阵的行数不同），则输出“error”。

输入样例：

3

A 10 100

B 100 5

C 5 50

A(BC)

输出样例：

75000

# 矩阵连乘最少计算次数

给定n个矩阵{A1,A2,…,An},其中，Ai与Ai+1是可乘的，i=1，2，….n-1。由于矩阵乘法满足结合律，故计算矩阵的连乘积可以有不同的计算次序。矩阵A和B可乘的条件是矩阵A的列数等于矩阵B的行数。若A是一个p×q矩阵，B是一个q×r矩阵，则其乘积C=AB是一个p×r矩阵。在上述计算C的标准算法中，主要计算量在3重循环，总共需要p×q×r次数乘。

# 石子合并

来源：洛谷P1880，hdu3506 Monkey Party

在一个圆形操场的四周摆放N堆石子,现要将石子有次序地合并成一堆.规定每次只能选相邻的2堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。

试设计出1个算法,计算出将N堆石子合并成1堆的最小得分和最大得分.。

【输入文件】

输入第一行为n(n<100)，表示有n堆石子，第二行为n个用空格隔开的整数，依次表示这n堆石子的石子数量（<=100）

【输出文件】

输出将n堆石子合并成一堆的最小得分和将n堆石子合并成一堆的最大得分。

【输入样例】

3

1 2 3

【输出样例】

9 11

# 最大子矩阵和

求一个M\*N的矩阵的最大子矩阵和。2 <= M,N <= 500。

输入：

第1行：M和N，中间用空格隔开（2 <= M,N <= 500)。

第2 - N + 1行：矩阵中的元素，每行M个数，中间用空格隔开。-10^9 <= a(i,j) <= 10^9。a(i,j)为矩阵的元素

输出：

输出和的最大值。如果所有数都是负数，就输出0。

输入样例：

3 3

-1 3 -1

2 -1 3

-3 1 2

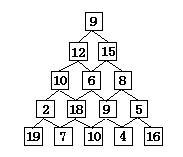
输出样例：

7

# 数塔问题

来源：HDU 2084

有形如下图所示的数塔，从顶部出发，在每一结点可以选择向左走或是向右走，一直走到底层，要求找出一条路径，使路径上的值的和最大。



输入格式：

输入数据首先包括一个整数C,表示测试实例的个数，每个测试实例的第一行是一个整数N(1 <= N <= 100)，表示数塔的高度，接下来用N行数字表示数塔，其中第i行有个i个整数，且所有的整数均在区间[0,99]内。

输出格式：

对于每个测试实例，输出可能得到的最大和，每个实例的输出占一行。

输入样例：

1

5

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

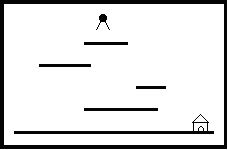
输出样例：

30

# Help Jimmy

来源：POJ 1661

场景中包括多个长度和高度各不相同的平台。地面是最低的平台，高度为零，长度无限。



Jimmy老鼠在时刻0从高于所有平台的某处开始下落，它的下落速度始终为1米/秒。当Jimmy落到某个平台上时，游戏者选择让它向左还是向右跑，它跑动的速度也是1米/秒。当Jimmy跑到平台的边缘时，开始继续下落。Jimmy每次下落的高度不能超过MAX米，不然就会摔死，游戏也会结束。

设计一个程序，计算Jimmy到底地面时可能的最早时间。

Input

第一行是测试数据的组数t（0 <= t <= 20）。每组测试数据的第一行是四个整数N，X，Y，MAX，用空格分隔。N是平台的数目（不包括地面），X和Y是Jimmy开始下落的位置的横竖坐标，MAX是一次下落的最大高度。接下来的N行每行描述一个平台，包括三个整数，X1[i]，X2[i]和H[i]。H[i]表示平台的高度，X1[i]和X2[i]表示平台左右端点的横坐标。1 <= N <= 1000，-20000 <= X, X1[i], X2[i] <= 20000，0 < H[i] < Y <= 20000（i = 1..N）。所有坐标的单位都是米。

Jimmy的大小和平台的厚度均忽略不计。如果Jimmy恰好落在某个平台的边缘，被视为落在平台上。所有的平台均不重叠或相连。测试数据保证问题一定有解。

Output

对输入的每组测试数据，输出一个整数，Jimmy到底地面时可能的最早时间。

Sample Input

1

3 8 17 20

0 10 8

0 10 13

4 14 3

Sample Output

23

# Humble Numbers

来源：HDU 1058, POJ 2247

A number whose only prime factors are 2,3,5 or 7 is called a humble number. The sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 24, 25, 27, ... shows the first 20 humble numbers.

Write a program to find and print the nth element in this sequence

Sample Input

1

2

11

21

100

1000

5842

0

Sample Output

The 1st humble number is 1.

The 2nd humble number is 2.

The 11th humble number is 12.

The 21st humble number is 28.

The 100th humble number is 450.

The 1000th humble number is 385875.

The 5842nd humble number is 2000000000.

# 厉害了我的杯

来源：腾讯面试题

有一种玻璃杯质量确定但未知，需要检测。

有一栋100层的大楼，该种玻璃杯从某一层楼扔下，刚好会碎。

现给你两个杯子，问怎样检测出这个杯子的质量，即找到在哪一层楼刚好会碎？

# 数字游戏

小W发明了一个游戏，他在黑板上写出了一行数字a1，a2，a3，……，an，然后给你M个回合的机会，每会回你可以从中选择一个数字擦去它，接着剩下来的每个数字ai都要递减一个值bi。如此重复m个回合，所有你擦去的数字之和就是你所得的分数。

小W和他的好朋友小Y玩了这个游戏，可是他发现，对于每个给出的a和b序列，小Y的得分总比他高，所以他就很不服气。于是他想让你帮他算算，对于每个a和b序列，可以得到的最大得分是多少。

输入文件：

输入文件的第一行是一个整数n（1<=n<=2000），表示数字个数；第二行一个整数m（1<=m<=n），表示回合数，接下来一行有n个不超过10000的正整数，a1，a2，a3，……，an表示原始序列，最后一行有n个不超过500的正整数，b1，b2，b3，……，bn，表示每回合每个数字递减的值。

输出文件：

输出文件只有一个整数，表示最大的可能得分。

输入样例：

3

3

10 20 30

4 5 6

输出样例：

47

# Phone Number(\*\*\*)

来源：Codeforces 44H

Alas, finding one's true love is not easy. Masha has been unsuccessful in that yet. Her friend Dasha told Masha about a way to determine the phone number of one's Prince Charming through arithmancy.

The phone number is divined like that. First one needs to write down one's own phone numbers. For example, let's suppose that Masha's phone number is 12345. After that one should write her favorite digit from 0 to 9 under the first digit of her number. That will be the first digit of the needed number. For example, Masha's favorite digit is 9. The second digit is determined as a half sum of the second digit of Masha's number and the already written down first digit from her beloved one's number. In this case the arithmetic average equals to (2 + 9) / 2 = 5.5. Masha can round the number up or down, depending on her wishes. For example, she chooses the digit 5. Having written down the resulting digit under the second digit of her number, Masha moves to finding the third digit in the same way, i.e. finding the half sum the the third digit of her number and the second digit of the new number. The result is (5 + 3) / 2 = 4. In this case the answer is unique. Thus, every *i*-th digit is determined as an arithmetic average of the *i*-th digit of Masha's number and the *i* - 1-th digit of her true love's number. If needed, the digit can be rounded up or down. For example, Masha can get:

12345

95444

Unfortunately, when Masha tried dialing the number, she got disappointed: as it turned out, the number was unavailable or outside the coverage area. But Masha won't give up. Perhaps, she rounded to a wrong digit or chose the first digit badly. That's why she keeps finding more and more new numbers and calling them. Count the number of numbers Masha calls. Masha calls all the possible numbers that can be found by the described means of arithmancy, except for, perhaps, her own one.

**Input**

The first line contains nonempty sequence consisting of digits from 0 to 9 — Masha's phone number. The sequence length does not exceed 50.

**Output**

Output the single number — the number of phone numbers Masha will dial.

**Examples**

**input**

12345

**output**

48

**input**

09

**output**

15

# 切棒子问题

给你一根长n英尺的棒子和一份关于该棒子的价目表如下（其中 i = 1,2,3,…,n），请问如何将这根棒子卖出最高的价格，可以对棒子进行切割。

http://www.roading.org/images/2012-03/image_thumb.png

# Long Path

来源：Codeforces 407B

time limit per test 1 second

memory limit per test 256 megabytes

One day, little Vasya found himself in a maze consisting of (*n* + 1) rooms, numbered from 1 to (*n* + 1). Initially, Vasya is at the first room and to get out of the maze, he needs to get to the (*n* + 1)-th one.

The maze is organized as follows. Each room of the maze has two one-way portals. Let's consider room number *i* (1 ≤ *i* ≤ *n*), someone can use the first portal to move from it to room number (*i* + 1), also someone can use the second portal to move from it to room number *pi*, where 1 ≤ *pi* ≤ *i*.

In order not to get lost, Vasya decided to act as follows.

* Each time Vasya enters some room, he paints a cross on its ceiling. Initially, Vasya paints a cross at the ceiling of room 1.
* Let's assume that Vasya is in room *i* and has already painted a cross on its ceiling. Then, if the ceiling now contains an odd number of crosses, Vasya uses the second portal (it leads to room *pi*), otherwise Vasya uses the first portal.

Help Vasya determine the number of times he needs to use portals to get to room (*n* + 1) in the end.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 103) — the number of rooms. The second line contains *n* integers *pi* (1 ≤ *pi* ≤ *i*). Each *pi* denotes the number of the room, that someone can reach, if he will use the second portal in the *i*-th room.

**Output**

Print a single number — the number of portal moves the boy needs to go out of the maze. As the number can be rather large, print it modulo 1000000007 (109 + 7).

**Examples**

**Input**

2  
1 2

**Output**

4

**Input**

4  
1 1 2 3

**Output**

20

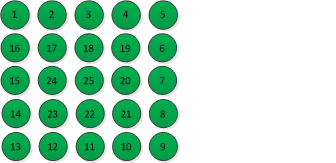
**Input**

5  
1 1 1 1 1

**Output**

62

# 滑雪问题



上图显示为R\*C的雪场，R是行数，C是列数。圆圈内的数字表示的是雪场的海拔高度h，根据常识我们知道，滑雪只有从上往下滑行才可能滑的动，现在我们想要求出能够滑行的最长距离，上面的例子我们可以很直接判断出25-24-......-1这个顺时针方向螺旋的滑雪方式可以滑的最远。