

Task 1: EUCLID

The famous Euclidean algorithm is found in Book VII of the *Elements*. The *Elements* was written in 300 B.C. by the Greek mathematician Euclid. It is rumored that King Ptolemy, having looked through the *Elements*, hopefully asked Euclid if there were not a shorter way to geometry, to which Euclid sternly answered: “In geometry there is no royal road!” Probably we should not blame the King for looking for short cuts, for there are thirteen books in the *Elements*! The books consist mainly of the mathematical knowledge amassed by Euclid, and possibly some discoveries of his own. The great achievement of Euclid is the beautifully systematic presentation of material as an organic whole. The *Elements* remained a standard work for over two thousand years.

The original Euclidean algorithm uses subtraction to find the greatest common divisor (gcd) of two positive integers A and B . It is based on the observation that a common divisor of A and B is also a common divisor of the integers $\min(A, B)$ and $\max(A, B) - \min(A, B)$. Thus the gcd of A and B can be found as

1. If $A = B$ then the gcd is B and the algorithm ends.
2. Replace A by $\max(A, B) - \min(A, B)$, B by $\min(A, B)$. Go to Step 1.

With the original Euclidean algorithm or otherwise, find the gcd of two positive integers.

Example

Let $A = 24, B = 15$. The original Euclidean algorithm computes

1. $A = 24 - 15 = 9, B = 15$
2. $A = 15 - 9 = 6, B = 9$
3. $A = 9 - 6 = 3, B = 6$
4. $A = 6 - 3 = 3, B = 3$

That is, $\gcd(24, 15) = 3$.

Input File: EUCLID.IN

The input file contains one line. The line contains two positive integers (each not larger than 32767) separated by spaces. The input file for the given example can be either

24 15

or

15 24

Output File: EUCLID.OUT

The file contains one integer which is the gcd of the two given positive integers. The output file for the given example contains

3

Task 2: CLAWS

Consider a heavy machinery in the form of a binary tree (i.e., every non-leaf node has *at most* two child nodes) as shown in Figure 1. The edges of the tree are metal bars, which are connected by hinges, denoted by nodes (labelled ‘1’ through ‘5’) in the tree. The leaf nodes of the tree in Figure 1(a) are essentially the claws used by the heavy machinery to perform heavy-duty tasks. The claws are also connected to metal bars by hinges.

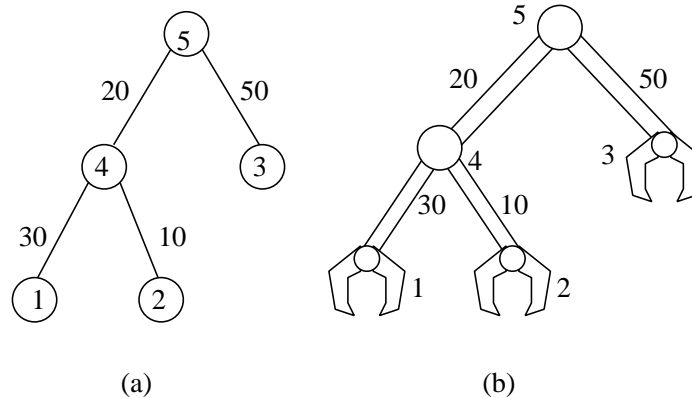


Figure 1: (a) A tree and (b) the corresponding tree of claws.

As the machinery is load-bearing, the hinges used to connect metal bars must be of the correct *grade*. For a particular node, say v , in the tree, the grade of the hinge used at that node is determined by the sum of (i) the total weight of the metal bars in the subtree rooted at v and (ii) the total weight of the metal bars from v to the tree root.

The weights of the metal bars are shown beside the corresponding edges in Figure 1. The claws and hinges used in the machinery are made of superlight materials that they can be considered weightless. Hence, the grades of the hinges used at leaf nodes ‘1’, ‘2’, and ‘3’ in Figure 1 to connect the claws to metal bars are 50, 30, and 50, respectively. The grade of the hinge used at node ‘4’ is 60, and the grade of the hinge used at the root node, ‘5’, is 110.

The *unladen load* of a claw is the sum of the grades of the hinges along the path from the claw to the root node. The unladen loads of the claws in this example are 220 (at ‘1’), 200 (at ‘2’), and 160 (at ‘3’). In this task, you have to output the maximum unladen load among all the claws of a given heavy machinery.

Input File: CLAWS.IN

Let n ($1 \leq n \leq 12000$) be the total number of nodes in the given binary tree. The input file contains n lines. The first line contains the integer n . Each of the next $(n - 1)$ lines contains three integers separated by spaces:

- **thisnode**, which specifies the node number of the current node.
- **parnode**, which specifies the node number of the parent of the current node.
- **weight**, which specifies the weight, a positive integer ≤ 100 , of the metal bar connecting the current node and its parent node.

The nodes are labelled 1 through n . The root node is the node without a parent.

For the machinery shown in Figure 1, the input file contains

```
5
1 4 30
4 5 20
2 4 10
3 5 50
```

Output File: CLAWS.OUT

For the machinery shown in Figure 1, the output file should contain the following integer:

```
220
```

Task 3: ECASINO

A *pseudo random number generator* (PRNG) is a program that, taking an input (known as *seed*), outputs a sequence of random bits $r[0], r[1], \dots$. (The value of a bit is either 0 or 1.)

Consider a particular PRNG and a seed. A *gambler* is a person (or a computer) who has observed the first L bits of the output $r[0], r[1], \dots, r[L - 1]$. The gambler knows the algorithm of that particular PRNG, and hence is aware of the internal mechanism of random bits generation. However, he does not know the value of the seed. With the knowledge of the PRNG algorithm and the first L bits $r[0], r[1], \dots, r[L - 1]$, the gambler wants to predict the subsequent output bits $r[L], r[L + 1], \dots$, that follow the observed bits.

It is not easy to design a PRNG that is unpredictable. The company E-Casino employs the following method. First, by observing some natural phenomenon, the company created a long sequence of N random bits $S[0], S[1], \dots, S[N - 1]$. This array S is made public and everyone, including gamblers, can access it. The seed is a tuple (k, m) , which consists of an integer k ($0 \leq k < N$) and an M -bit sequence $m = \langle m[0], m[1], \dots, m[M - 1] \rangle$. For $j = 0, 1, 2, \dots$, the output bit $r[j]$ is

$$r[j] = S[(k + j) \bmod N] \text{ xor } m[j \bmod M]. \quad (1)$$

The operator `xor` is the “exclusive or” operation. That is, for any bits a, b ,

$$a \text{ xor } b = (a + b) \bmod 2.$$

The company *always* uses

$$N = 2048, \quad M = 32.$$

But every morning, the managers of E-Casino will collectively choose a secret seed (k, m) , which is to be used in generating a random sequence.

Suppose you are the gambler and for a particular day you have observed the first $2M = 64$ bits of the output sequence: $r[0], r[1], \dots, r[63]$. You also know the values of the array S . However, you do not know the value k and the sequence m . Your ultimate goal is to determine the subsequent bits. In order to do that however, you have to first determine the value of k in the secret seed (k, m) . In this task, you are to find the smallest possible k .

Example

The following is a small example with $N = 10$, $k = 4$, $m[0] = 1$, $m[1] = 0$, $m[2] = 1$, (i.e., $M = 3$), to illustrate the computation of $r[0]$, $r[1]$, \dots , $r[5]$ in (1). The array S is shown in the third row of the table. (For clarity, the value of $r[6]$, $r[7]$, $r[8]$, and $r[9]$ are not shown.)

j	6	7	8	9	0	1	2	3	4	5
$(k + j) \bmod N$	0	1	2	3	4	5	6	7	8	9
$S[(k + j) \bmod N]$	1	1	1	0	0	1	0	1	0	1
$m[j \bmod M]$					1	0	1	1	0	1
$r[j]$					1	1	1	0	0	0

Hence, the first 6 output bits are 1,1,1,0,0,0.

If a gambler sees only the first 6 output bits, but does not know the value k and the array m , he can still verify that the value of k must be 4. This can be verified by listing down all the combinations of k and m , and check which combination gives the 6 output bits. But note that for large N and M , this simple exhaustive search is infeasible.

Input File: ECASINO.IN

The file contains only one line. The first $2M$ characters of the file are the observed bits $r[0]$, $r[1]$, \dots , $r[2M - 1]$. Each bit is represented by the character '0' or '1'. The very first character represents $r[0]$, followed by $r[1]$ and so on. Immediately following the $2M$ characters is the character '%', which marks the start of the array S . The array S is represented as a string of '0' and '1'. The first character immediately following the marker '%' is $S[0]$, and the next character is $S[1]$, and so on. In total, the file consists of $2M + 1 + N = 2113$ input characters.

Note that for your task it is always $N = 2048$ and $M = 32$. But for the given example $N = 10$ and $M = 3$ and thus the input file for the given example looks like

```
111000%1110010101
```

Output File: ECASINO.OUT

The output file contains an integer, which is the secret k . For the given example: the output file should contain

4

Task 4: GENOME

In comparative genomics, biologists would like to find a gene sequence that is conserved among a set of species.

Let $\{1, 2, \dots, n\}$ be a set of n integers in which each integer represents a gene. For the m species S_1, S_2, \dots, S_m , each species S_i is identified by a permutation of $\{1, 2, \dots, n\}$. The permutation represents the ordering of genes in S_i .

A subsequence of an integer sequence is obtained by omitting none, one, or more integers from the original sequence. An integer sequence x_1, x_2, \dots, x_k is a *conserved gene sequence* among the m species if x_1, x_2, \dots, x_k is a subsequence of S_i for all $i = 1, 2, \dots, m$. Our aim is to find the length of the longest conserved gene sequences for the m species.

Example

Consider the following 3 species,

- 5, 3, 4, 1, 2;
- 2, 5, 4, 3, 1;
- 5, 2, 3, 1, 4.

The following subsequences

- 5, 1;
- 5, 3;
- 5, 4;
- 3, 1;

are conserved gene sequences among the 3 species but are not longest. The longest conserved gene sequence of the 3 species is 5, 3, 1.

Input File: GENOME.IN

The input file `GENOME.IN` contains $m + 1$ lines. The first line contains two integers n and m separated by spaces, where $1 \leq n \leq 100$ and $1 \leq m \leq 10$. Each of the next m lines contains a permutation of $1, 2, \dots, n$, with spaces between two adjacent integers. For the given example, the input file contains:

```
5 3
5 3 4 1 2
2 5 4 3 1
5 2 3 1 4
```

Output File: GENOME.OUT

The output file `GENOME.OUT` contains an integer that gives the length of the longest conserved gene sequences. For the given example, the output file contains:

```
3
```

Task 5: FLUDTOWN

Fludtown is a small $1 \text{ km} \times 1 \text{ km}$ square-shaped township in Rainee county. The houses in Fludtown are scattered randomly around the township but property ownership follows a simple rule: the land at any point in Fludtown belongs to the house whose straight-line distance to the point is shortest. Of course a house cannot be located at the same spot where another house is.

Last week, the Fenster family moved into one of Fludtown's houses. Mr. Fenster had the property agent put down markers to mark the edges of his property so that he could build a fence around the house. Unfortunately, a big rain storm hit Fludtown yesterday and the markers have all been wiped away. Fortunately, he does have the town map, indicating where the Fenster house and the other properties are.

Fenster is going to buy fencing material from the Fence Depot outside town. Help Mr Fenster figure out how many fenceposts he needs to buy (fenceposts are only needed at the points where the fence changes direction). If the Fenster house property extends to the border of Fludtown, make sure to include the necessary posts for the border as well. Note that some houses may not influence the number of fenceposts required.

Input File: FLUDTOWN.IN

The location of a Fludtown house is specified with (x, y) coordinates. The origin $(0, 0)$ of the coordinate system is located at the south-west corner of Fludtown. The x axis runs from west to east, the y axis runs from south to north, and 1 coordinate unit corresponds to 1 meter.

Let n ($1 \leq n \leq 10$) be the number of houses in Fludtown. The input file contains $n + 1$ lines. The first line contains the integer n . Each of the next n lines contains two integers x, y separated by spaces, giving the (x, y) coordinates of the location of a house. The location of Fenster family house is given before the rest of the houses, i.e., in the second input line. All houses in the 1 km^2 Fludtown are located in or on the edges of the square township, i.e., $0 \leq x \leq 1000, 0 \leq y \leq 1000$.

Example 1

Fenster house is at $(500, 500)$ and his fence forms a pentagon:

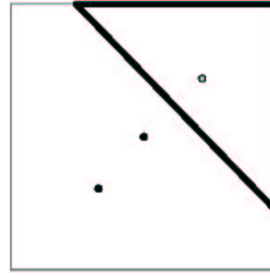
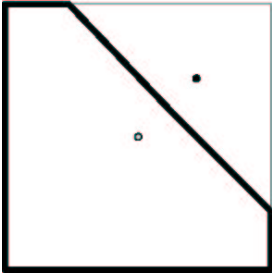
```
2
500 500
700 700
```

Example 2

Fenster house is at $(700, 700)$ and his fence forms a triangle:

```
3
700 700
500 500
200 200
```

The diagram below depicts the two data sets.



Output File: FLUDTOWN.OUT

The output file contains an integer giving the number of fenceposts required to build the fence for Fenster's house.

Example 1

5

Example 2

3

Some Formulas

- The line equation for the perpendicular bisector of the line segment connecting the two points (x_0, y_0) and (x_1, y_1) is

$$(y_1 - y_0) \left(y - \frac{y_0 + y_1}{2} \right) + (x_1 - x_0) \left(x - \frac{x_0 + x_1}{2} \right) = 0.$$

- If the two lines

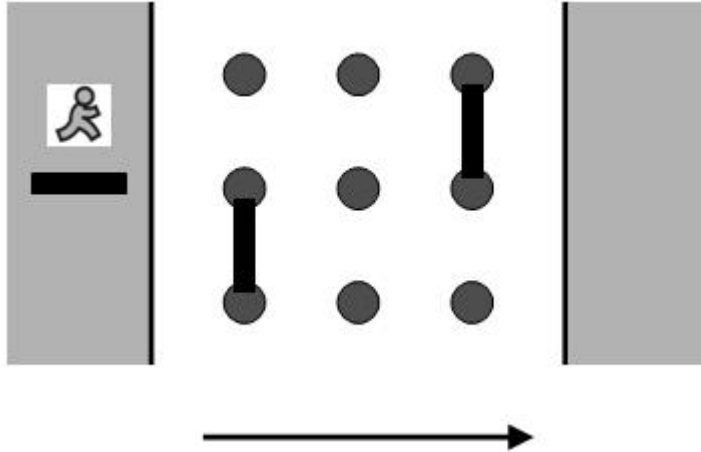
$$\begin{aligned} ax + by + c &= 0, \\ dx + ey + f &= 0 \end{aligned}$$

intersect, the intersection is the point

$$\left(\frac{bf - ce}{ae - bd}, \frac{cd - af}{ae - bd} \right).$$

Task 6: HENG

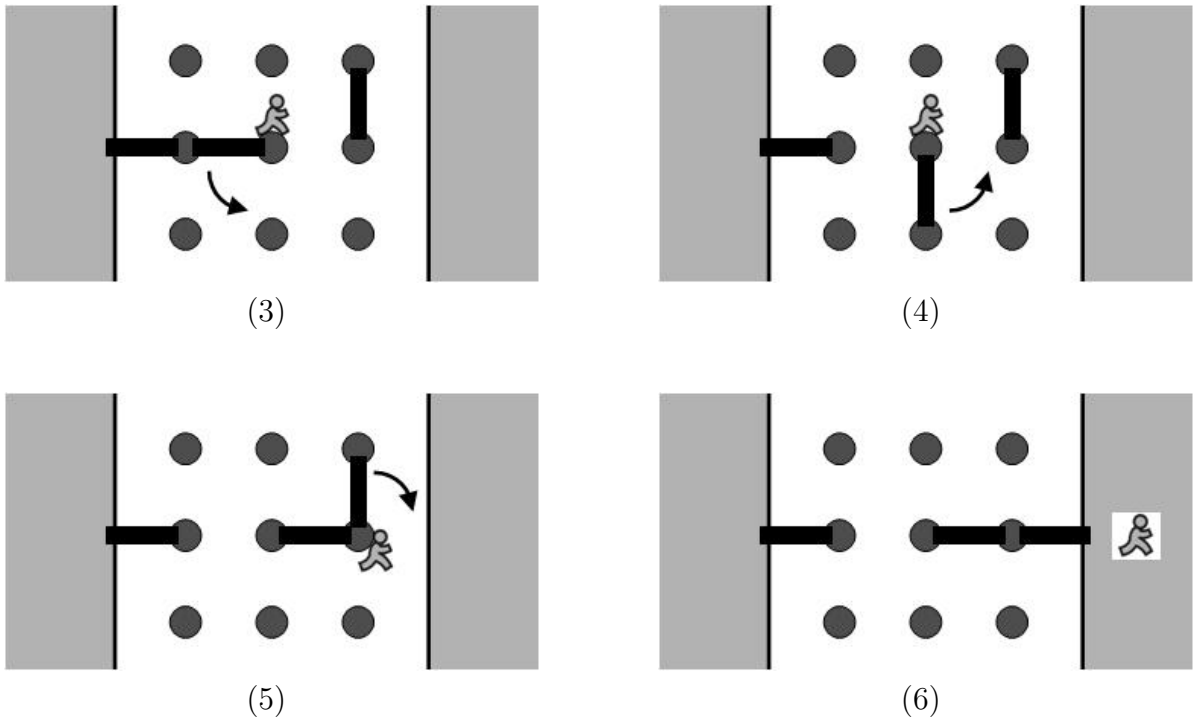
Heng wants to cross a river. The river is dotted with islands that he can reach with boards serving as bridges. On his side of the river, Heng has a board that points to the other side. Some of the islands are already connected with boards such that the boards point in the direction of the river flow. The islands are arranged in an $N \times N$ grid. For $N = 3$, the situation may look like this.



When Heng is on an island, he can turn any board that touches the island by 90 degrees such that after the turn it still touches the island. Since turning a board is quite tiresome, he wants to cross the river in such a way that he needs to turn boards by 90 degrees as few times as possible. Of course, Heng can turn a board by 180 degrees by turning it by 90 degrees twice.

In the beginning, Heng has a board on his side of the river, which points to the other side. Thus, initially, he can reach any island in the first column of islands without turning a board. So in the situation above, he can cross the river with 4 board turns as follows:





Input File: HENG.IN

The input file `HENG.IN` contains N lines, $2 \leq N \leq 40$. The first line contains the integer N . Each of the next $N - 1$ lines contains N characters which are 0 or 1, and these N characters describe the bridge configuration between two adjacent rows of islands. The character 1 means there is a bridge and the character 0 means there is no bridge. Thus the second line of the input file describes the bridge configuration between the top two rows of islands, and so on, and the last input line describes the bridge configuration between the bottom two rows of islands. The first of the N characters describes the leftmost bridge configuration between the leftmost two islands of two adjacent rows of islands, and so on, and the rightmost character describes the rightmost bridge configuration between the rightmost two islands of the same two rows of islands. The input file for the starting configuration of the given example contains:

```
3
001
100
```

Output File: HENG.OUT

The output file `HENG.OUT` contains an integer, indicating the smallest number of board turns required to cross the river. The output file for the given example contains:

```
4
```