Training Camp Córdoba 2012
Martes 24 de Julio

Inicio de la prueba: 14:00hs
Duración de la prueba: 4 horas

Problema 1 - Tree 2

Consider a tree consisting of n vertices. A distance between two vertices is the minimal number of edges in a path connecting them. Given a vertex v_i and distance d_i find a vertex u_i such that distance between v_i and u_i equals to d_i.

Input: The first line contains the number of vertices $n\,(1 \leq n \leq 20{,}000)$ and the number of queries $q\,(1 \leq q \leq 50{,}000)$. Each of the following $n-1$ lines describes an edge and contains the numbers of vertices connected by this edge. Vertices are numbered from 1 to n. The next q lines describe the queries. Each query is described by a line containing two numbers $v_i\,(1 \leq v_1 \leq n)$ and $d_i\,(1 \leq d_i \leq n)$.

Output: You should output q lines. The i-th line should contain a vertex number u_i, the answer to the i-th query. If there are several possible answers, output any of them. If there are no required vertices, output 0 instead.

Sample Input:
9 10
1 8
1 5
1 4
2 7
2 5
3 6
5 9
6 9
5 4
8 1
4 3
2 4
9 3
1 1
5 2
3 5
6 4
7 3

Sample Output:
0
1
2
3
4
5
6
7
8
9

Problema 2 - Is it a Tree

You are give an unweighted, undirected graph. Write a program to check if it's a tree topology.

Input: The first line of the input file contains two integers N and M — number of nodes and number of edges in the graph ($0 < N <= 10{,}000, 0 <= M <= 20{,}000$). Next M lines contain M edges of that graph — Each line contains a pair (u, v) means there is an edge between node u and node v ($1 <= u, v <= N$).

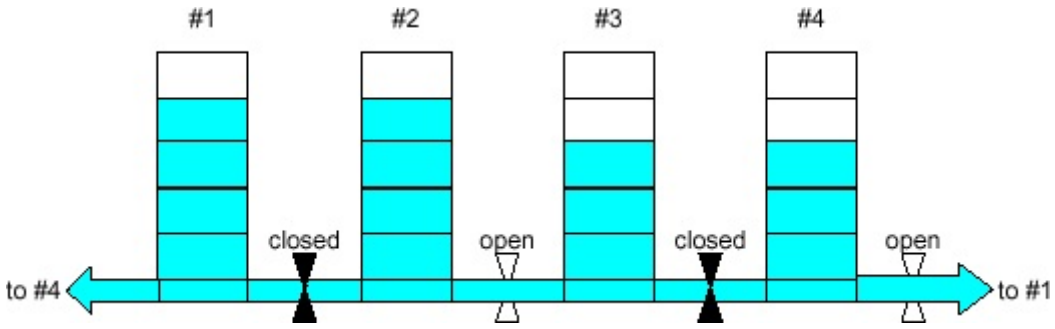
Output: Print YES if the given graph is a tree, otherwise print NO.

Sample Input:
3 2
1 2
2 3

Sample Output:
YES

Problema 3 - Water Tanks

There are n identical large cylindrical tanks for storing water. The tanks are arranged in a circle on level ground. Each tank is connected with its two neighbours by means of pipes situated at its base. There is a valve between each adjacent pair of tanks (tank n is next to tank 1). All valves are initially closed. All the outlets and the pipes are at the same level and are always full of water, even if all the tanks are deemed to be "empty". The volume in any tank is measured by the height of the surface of the water above the level of the top of the outlets. If all valves (or all valves but one) are opened so that water can flow between the tanks, then the levels will eventually equalise. Conversely, if all tanks are initially at the same level, no valves need be opened to equalise the levels. Thus it may be necessary to only open some of the valves to achieve this result. For example, consider $n = 4$ tanks each 5 metres high. Assume that the water level in these tanks is at 4, 4, 3, and 3 meters respectively. Their water level will equalise if we open the valves between tanks #2 and #3 and between #4 and #1, as suggested by the following diagram. Thus for this set we need to open only two valves:



Given a set of initial heights, determine the minimum number of valves to open so that the final water levels in all tanks is equal.

Input: The input will consist of one or more scenarios, each scenario consisting of two lines. The first line contains a descriptive title, which is a string of letters or spaces no more than 200 characters long, containing at least 1 letter. The second line starts with the number of basins $n\,(3 \leq n \leq 200)$, a space, and then n integers in the range 0 to 99, separated by single spaces, representing the water levels in the tanks. The scenarios sequence is terminated by a single '#' character on a line by itself.

Note: The input data for this program may contain lines up to 600 characters long

Output: Output one line for each input scenario. The line consists of the first letter of each word in the descriptive title in upper case, followed by a colon (':'), a space, and then the minimum number of valves that need to be open to achieve equal heights in all tanks.

Sample Input:
High four dude basins
4 4 4 3 3
The Australasian eight
8 2 1 1 2 2 1 1 6
#

Sample Output:
HFDB: 2
TAE: 5

Problema 4 - Subset Sums

Given a sequence of $N\,(1 \leq N \leq 34)$ numbers $S_1, \ldots, S_N\,(-20{,}000{,}000 \leq Si \leq 20{,}000{,}000)$, determine how many subsets of S (including the empty one) have a sum between A and $B\,(-500{,}000{,}000 \leq A \leq B \leq 500{,}000{,}000)$, inclusive.

Input: The first line of standard input contains the three integers N, A, and B. The following N lines contain S_1 through S_N, in order.

Output: Print a single integer to standard output representing the number of subsets satisfying the above property. Note that the answer may overflow a 32-bit integer.

Sample Input:
3 -1 2
1
-2
3

Sample Output:
5

The following 5 subsets have a sum between -1 and 2:
0 = 0 (the empty subset)
1 = 1
1 + (-2) = -1
-2 + 3 = 1
1 + (-2) + 3 = 2

Problema 5 - Team building

There are n programmers in the software development company. Each of them thinks that he is the greatest or the second greatest programmer in the company. In the latter case he can name the greatest programmer in his opinion.
The administration decided to divide all programmers into development teams using the following algorithm:

1. If there are programmers, who are not assigned to a development team, choose any of them and mark him as current one.
2. Create a new development team and assign it to the current programmer.
3. If the current programmer thinks that one of his colleagues is the greatest programmer and this colleague is not assigned to a development team, then this colleague is assigned to the same development team and is marked as current programmer. Then the step 3 is repeated. Otherwise, the team is formed, and the administration returns to the step 1.

What is the minimal and maximal number of teams which can be formed in this company according to the algorithm?

Input: The first line contains an integer $n\,(1 \leq n \leq 10^5)$. The i-th of the following n lines contains the number of the programmer, who is the greatest, according to the i-th programmer's opinion.

Output: Output the minimal and maximal number of development teams, separated with space.

Sample Input:
4
2
3
4
2

Sample Output:
1 2

Problema 6 - Perfect Election

In a country (my memory fails to say which), the candidates $\{1, 2, \ldots, N\}$ are running in the parliamentary election. An opinion poll asks the question "For any two candidates of your own choice, which election result would make you happy?". The accepted answers are shown in the table below, where the candidates i and j are not necessarily different, i.e. it may happen that $i = j$. There are M poll answers, some of which may be similar or identical. The problem is to decide whether there can be an election outcome (It may happen that all candidates fail to be elected, or all are elected, or only a part of them are elected. All these are acceptable election outcomes.) that conforms to all M answers. We say that such an election outcome is perfect. The result of the problem is '1' if a perfect election outcome does exist and 0 otherwise.

Input: Write a program that reads sets of data from an input text file. Each data set corresponds to an instance of the problem and starts with two integral numbers: $1 \leq N \leq 1{,}000$ and $1 \leq M \leq 1{,}000{,}000$. The data set continues with M pairs $\pm i \pm j$ of signed numbers, $1 \leq i, j \leq N$. Each pair encodes a poll answer as follows:

Accepted answers to the poll question	Encoding
I would be happy if at least one from i and j is elected.	$+i\ +j$
I would be happy if at least one from i and j is not elected.	$-i\ -j$
I would be happy if i is elected or j is not elected or both events happen.	$+i\ -j$
I would be happy if i is not elected or j is elected or both events happen.	$-i\ +j$

The input data are separated by white spaces, terminate with an end of file, and are correct.

Output: For each data set the program prints the result of the encoded election problem. The result, 1 or 0, is printed on the standard output from the beginning of a line. There must be no empty lines on output. An example of input/output is shown below.

Note for the Sample:

For the first data set the result of the problem is 1; there are several perfect election outcomes, e.g. 1 is not elected, 2 is elected, 3 is not elected. The result for the second data set is justified by the perfect election outcome: 1 is not elected, 2 is not elected. The result for the third data set is 0. According to the answers $-1\ +2$ and $-1\ -2$ the candidate 1 must not be elected, whereas the answers $+1\ -2$ and $+1\ +2$ say that candidate 1 must be elected. There is no perfect election outcome. For the fourth data set notice that there are similar or identical poll answers and that some answers mention a single candidate. The result is 1.

Sample Input:

```
3 3  +1 +2  −1 +2  −1 −3
2 3  −1 +2  −1 −2  +1 −2
2 4  −1 +2  −1 −2  +1 −2  +1 +2
2 8  +1 +2  +2 +1  +1 −2  +1 −2  −2 +1  −1 +1  −2 −2  +1 −1
```

Sample Output:

```
1
1
0
1
```

Problema 7 - Balance Trip

You know that May Day is coming and all the students will have a 7 days' vacation. So Robby has began to plan his trip to the famous Big Cow City now...
There are N cities and M roads. All the cities are numbered from 1 to N. Robby doesn't care how long he will travel, but he wants his trip to be as balance as possible. To be more precise, he defines the balance value of a path as the difference between the longest road and the shortest road of the path. Robby wants to find a path with the minimum banlance value. Can you help him?

Input: The first line of each test case has two numbers N and $M\,(2 \leq N \leq 200, 1 \leq M \leq 1000)$. Each of the following M lines describes a road and will contain three numbers A, B and $D\,(1 \leq A, B \leq N, A \neq B, 1 \leq D \leq 10^6)$. A and B are the two ends of the road, D is the length of the road. The last line of each case contains two numbers S and $T\,(1 \leq S, T \leq N, S \neq T)$, indicating Robby's home and the Big Cow City.
The input is terminated by a test case starting with N = M = 0. This test case should not be processed.

Output: Your program should output one line for each test case, which is the minimum balance value. The value should be "-1" if Robby can not get to Big Cow City.

Sample Input:
5 5
1 2 100
2 3 101
3 4 102
1 5 20
5 4 80
1 4
0 0

Sample Output:
2

Hint: Considering two paths from 1 to 4, that is, $1 \to 2 \to 3 \to 4$ and $1 \to 5 \to 4$, the balance value of the first path is $102 - 100 = 2$ and that of the second path is $80 - 20 = 60$, so the first one is better.

Problema 8 - Maze Stretching

Usually the path in a maze is calculated as the sum of steps taken from the starting point until the ending point, assuming that the distance of one step is exactly 1. Lets assume that we could "stretch" (shorten or extend) the maze in vertical dimension (north-south). By stretching, we are just changing the passed distance between two cells. (it becomes X instead of one). We have a two dimensional maze which has '#' for walls, 'S' in the starting cell and 'E' at the ending cell.
Due to outside conditions, we need to make the shortest path to be exactly L in size.
We are not allowed to change the maze configuration, nor to make any changes in the horizontal dimension. We are only allowed to stretch the vertical dimension, and that can be done by any percentage.
Find the percentage of the stretch P, for which the shortest path of the maze will be exactly L.

Input: First line of the input contains the number of test cases. For each test case, firstly two numbers L and N are given, where L is the required length of the shortest path and N is the number of lines that are given describing the maze. The following N lines describes the maze such as that each line describes a row of the maze. (Each row length is the horizontal dimension of the maze).

Constraints:

- The height and width of the maze are maximum 100 cells.
- All the lines describing one maze are the same size.
- There will always be a solution.
- The result will be between 0.000 and 1000.000 inclusive
- There will be no direct horizontal only path connecting 'S' and 'E'. (the result is always unique).

Output: For each test case output the percentage of the stretch in the following format:

Case #K: P %

P should have leading zero if the number is between 0 and 1.
P should be rounded up on 3 decimals, and always formatted on 3 decimals (with trailing zeros if needed).

Explanation of the first test case in the example: On the original maze, the length of the shortest path is 3 because there are two horizontal steps and a vertical one. Our goal is to make the length of the shortest path to be 2.5. That's why we have to "stretch"the vertical dimension of the maze by a percentage value less than 100. In this case it is 50 % which actually changes the vertical distance between two cells to 0.5.

Sample Input:

```
2
2.5 4
#####
#S  #
#  E#
#####
21 13
############
#S##      #E#
# ##  #  # #
#   # #  # #
### # #  # #
#   # #  # #
#  ## #  # #
##  # #  # #
### # #  # #
##  # #  # #
#  ## #    #
#     #    #
############
```

Sample Output:
Case #1: 50.000 % Case #2: 21.053 %

Problema 9 - Moving to Nuremberg

One of the most important inventions for modern-day city life is the public transportation. However, most people probably do not think of it that way - even though it makes travel in the city a lot easier, we generally want to spend as little time as possible on the subway. After your experience in NWERC 2009, Nuremberg holds a special place in your heart, and some years later you decide to move here. Your only problem is to figure out which part of Nuremberg to move to. Naturally, you want to move to a nice neighborhood, but since most parts of the city are nice there are still a lot of choices. Being naturally averse to spending hours each day on commuting, you instead decide to choose a place based on the amount of time you will have to spend on the subway.
Now, if you were only going to go to one place, it would be easy to find the best place to live. But of course, there are several places where you anticipate that you will go regularly, such as work, friends, and the occasional Christkindlesmarkt. In order to be able to work this out, you have written a list of all the places which you want to visit regularly, along with estimates of how often you want to go there. For simplicity, you assume that you will always go somewhere and then back home, e.g., if you are going to Christkindlesmarkt after work you will drop by your house on the way from work before going to the markt, rather than going to the markt directly from work. Now, you have to find the place to live for which the total travel time is minimal.
Because Nuremberg has an extensive public transportation system, you will be using the subway for traveling. The subway net is quite big, but is still fairly easily maneuvered because it is shaped like a tree. In other words, there is always a unique path between any pair of subway stations. (This is not quite true for the Nuremberg subway of today, but when you move here in a few years, we anticipate that it will be true.)

Input: The input consists of several test cases. The first line of input contains an integer $c\,(1 \leq c \leq 200)$, giving the number of test cases. Then, each test case starts with an integer $n\,(1 \leq n \leq 50{,}000)$, giving the number of subway stations in Nuremberg. Then follow $n-1$ lines, describing the subway net. Each of these lines contains three integers a, b, and $t\,(1 \leq a, b \leq n, 1 \leq t \leq 300)$, indicating that stations a and b are adjacent and that it takes t seconds to travel between them. This is followed by a line containing an integer $m\,(0 \leq m \leq n)$, denoting the number of stations which you want to go to regularly. Then follow m lines. Each of these lines contains two integers a and $f\,(1 \leq a \leq n, 1 \leq f \leq 500)$, where a is the station you want to visit and f is the number of times you want to visit this station in a year. No station will occur in this list more than once.

Output: For each test case, first output a line containing the number of seconds spent in traffic during a year, provided you choose an optimal place to live. Following this line, output a line giving all optimal choices of subway stations, separated by single spaces and in increasing order.

Sample Input:
2
2
1 2 17
2
1 5
2 10
5
1 3 10
2 3 20
3 4 30
4 5 30
3
1 10
2 10
5 20

Sample Output:
170
2
3000
3 4 5

Problema 10 - Faulty Odometer

You are given a car odometer which displays the miles traveled as an integer. The odometer has a defect, however: it proceeds from the digit 3 to the digit 5, always skipping over the digit 4. This defect shows up in all positions (the one's, the ten's, the hundred's, etc.). For example, if the odometer displays 15339 and the car travels one mile, odometer reading changes to 15350 (instead of 15340).

Input: Each line of input contains a positive integer in the range 1..999999999 which represents an odometer reading. (Leading zeros will not appear in the input.) The end of input is indicated by a line containing a single 0. You may assume that no odometer reading will contain the digit 4.

Output: Each line of input will produce exactly one line of output, which will contain: the odometer reading from the input, a colon, one blank space, and the actual number of miles traveled by the car.

Sample Input:
13
15
2003
2005
239
250
1399
1500
999999
0

Sample Output:
13: 12
15: 13
2003: 1461
2005: 1462
239: 197
250: 198
1399: 1052
1500: 1053
999999: 531440