Edición 20
Concurso
Noviembre 2020

Concurso de Programación
Facultad de Ingeniería
Universidad ORT Uruguay

CARACTERISTICAS Y RECOMENDACIONES GENERALES

Presentarse: 12:45 hs. Hora de comienzo: 13:00 hs. Hora de finalización: 16:30 hs
Nota: Durante la prueba habrá corte para ventilación

- **Cada equipo recibe un sobre que contiene:**
  - identificación del grupo: una hoja con la identificación (ejemplo: grupoXX) y la contraseña (ejemplo: claveXX).
  - 5 propuestas de ejercicios

Ingresar en Windows con el usuario propio y contraseña personal. El dominio es Facultades.

Mapear unidad de red. Para ello, en el Explorador de Windows, seleccionar con botón derecho RED e ir a la opción "Conectar a unidad de red". Indicar:

En carpeta: \\172.16.100.11\ConcursoPGM
Seleccionar conectar con otras credenciales.
Click en "Finalizar"

Aparecerá una ventana "Seguridad de Windows" que solicita usuario y contraseña. Ahí ingresar como

usuario: facultades\grupoXX (siendo XX el número del grupo asignado)
contraseña: YYYY (siendo YYYY la contraseña asignada para el grupo).

Cada grupo solamente podrá acceder a su propio directorio. Este directorio contendrá a su vez varios directorios, uno para cada ejercicio respectivamente. Allí se ubicarán los ejercicios para entregar.

- **Acerca del material de consulta:**

Se puede consultar material impreso propio traído por el equipo. No se permite utilizar CDs, pen drives, celulares, Internet ni ninguna otra fuente de información. Tener presente que, por no estar disponible la conexión a Internet, el "help" puede no funcionar en algunos ambientes.

- **Procedimiento de entrega de una resolución:**

Cuando el equipo tenga pronto un ejercicio, copiará el código fuente del ejercicio a la carpeta respectiva de la unidad de red y avisará al docente o asistente a cargo, firmándose una planilla en la que constará el número de ejercicio y la hora de entrega. Se computará como tiempo de ese ejercicio el tiempo transcurrido desde el comienzo de la prueba hasta ese momento. No se probarán ejercicios que no estén firmados en la planilla. Chequear especialmente que se ubiquen los archivos en el directorio correspondiente al ejercicio. De ubicarse erróneamente, se considerará como entrega fallida y no será probado.
En el respectivo directorio deberá estar si se resolvió en:

Java: todos los archivos .java de las clases necesarias
C,C++: .exe

- **Verificación de una resolución:**

Cada ejercicio entregado será probado lo antes posible y en caso de detectar error en la ejecución o demora excesiva, se avisará al equipo, quien tiene la posibilidad de corregirlo y volverlo a entregar.

En caso de volverlo a entregar, se realizará el mismo procedimiento de entrega detallado, computándose 20 minutos adicionales de multa por cada entrega fallida.

La prueba del ejercicio se realizará corriendo dicho ejercicio asumiéndose como nombres **obligatorios** para los archivos de prueba los siguientes:

Ejercicio	Archivo de Entrada	Archivo de Salida
1	d:\ej1datos.txt	d:\ej1resul.txt
2	d:\ej2datos.txt	d:\ej2resul.txt
3	d:\ej3datos.txt	d:\ej3resul.txt
4	d:\ej4datos.txt	d:\ej4resul.txt
5	d:\ej5datos.txt	d:\ej5resul.txt

- **Premiación:** se entregarán reconocimientos al 1er, 2do y 3er puesto.

1 - Amusement park: Roller coaster

An amusement park includes a new attraction: a roller coaster with magic tunnels that switch the positions of the passengers in the cart. The passengers sit in a single row.

When the cart enters in a magic “simple” tunnel, the first passenger switches its place with the last passenger.
Example:
Passengers in the cart:
Before the “simple” tunnel: “ABCDEFGH”
After the “simple” tunnel: “HBCDEFGA”

When the cart enters in a “full” tunnel, the first passenger exchanges its place with the last, the second with the penultimate, and so on:
Example:
Passengers in the cart:
Before the “full” tunnel: “ABCDEFGH”
After the “full” tunnel: “HGFEDCBA”

Given a roller coaster, which is the minimum amount of tunnels to be added at the end in order to ensure that passengers end the ride in the same order they started at?

Output a line with a number, which represents the minimum number of tunnels to be added at the end of the roller coaster to ensure that the order of the passengers in the cart is the same as the begginning.

Input
The first line contains the number n of test cases (1<=n<=20).
For each case, the first line contains a string C (uppercase) representing the initial position of the passengers in the cart (1<length(C)<20). Each passenger is represented by a different letter (English alphabet, no spaces). The cart is full.
The second line contains a string S (uppercase) representing the sequence of tunnels (2<length(S)<50). Each tunnel is: “F” (full) or “S” (simple).

Output
For each case, output a line with “Case #i:” with the number i of the case (1<=i<=n) and the minimum number of tunnels needed to restore the passengers to their original positions.

Sample input
2
ABCDEFGH
FSFS
IARBS
FFF

Sample output
Case #1: 0
Case #2: 1

2 - Become Parents

Cameron and Jamie are longtime life partners and have recently become parents! Being in charge of a baby, exciting as it is, is not without challenges. Given that both parents have a scientific mind, they have decided to take a scientific approach to baby care.

Cameron and Jamie are establishing a daily routine and need to decide who will be the main person in charge of the baby at each given time. They have been equal partners their whole relationship, and they do not want to stop now, so they decided that each of them will be in charge for exactly 12 hours (720 minutes) per day.

Cameron and Jamie have other activities that they either need or want to do on their own. Cameron has $\mathbf{A_C}$ of these and Jamie has $\mathbf{A_J}$. These activities always take place at the same times each day. None of Cameron's activities overlap with Jamie's activities, so at least one of the parents will always be free to take care of the baby.

Cameron and Jamie want to come up with a daily baby care schedule such that:

- Scheduled baby time must not interfere with a scheduled activity. That is, during Cameron's activities, Jamie has to be in charge of the baby, and vice versa.
- Each of Cameron and Jamie must have exactly 720 minutes assigned to them.
- The number of exchanges — that is, the number of times the person in charge of the baby changes from one partner to the other — must be as small as possible.

For example, suppose that Jamie and Cameron have a single activity each: Jamie has a morning activity from 9 am to 10 am, and Cameron has an afternoon activity from 2 pm to 3 pm. One possible but suboptimal schedule would be for Jamie to take care of the baby from midnight to 6 am and from noon to 6 pm, and for Cameron to take care of the baby from 6 am to noon and 6 pm to midnight. That fulfills the first two conditions, and requires a total of 4 exchanges, which happen at midnight, 6 am, noon and 6 pm. If there is an exchange happening at midnight, it is counted exactly once, not zero or two times.

A better option would be for Cameron to take care of the baby from midnight to noon, and Jamie to take care of the baby from noon to midnight. This schedule also fulfills the first two conditions, but it uses only 2 exchanges, which is the minimum possible.

Given Cameron's and Jamie's lists of activities, and the restrictions above, what is the minimum possible number of exchanges in a daily schedule?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing two integers $\mathbf{A_C}$ and $\mathbf{A_J}$, the number of activities that Cameron and Jamie have, respectively. Then, $\mathbf{A_C} + \mathbf{A_J}$ lines follow. The first $\mathbf{A_C}$ of these lines contain two integers $\mathbf{C_i}$ and $\mathbf{D_i}$ each. The i-th of Cameron's activities starts exactly $\mathbf{C_i}$ minutes after the start of the day at midnight and ends exactly $\mathbf{D_i}$ minutes after the start of the day at midnight (taking exactly $\mathbf{D_i} - \mathbf{C_i}$ minutes). The last $\mathbf{A_J}$ of these lines contain two integers $\mathbf{J_i}$ and $\mathbf{K_i}$ each, representing the starting and ending time of one of Jamie's activities, in minutes counting from the start of the day at midnight (same format as Cameron's). No activity spans two days, and no two activities overlap (except that one might end exactly as another starts, but an exchange can still occur at that time).

Limits

$1 \le \mathbf{T} \le 100$.
$0 \le \mathbf{C_i} < \mathbf{D_i} \le 24 \times 60$, for all i.
$0 \le \mathbf{J_i} < \mathbf{K_i} \le 24 \times 60$, for all i.
Any two of the intervals of {[$\mathbf{C_i}$, $\mathbf{D_i}$) for all i} union {[$\mathbf{J_i}$, $\mathbf{K_i}$) for all i} have an empty intersection. (The intervals are closed on the left and open on the right, which ensures that two exactly consecutive intervals have nothing in between but do not overlap.)
sum of {$\mathbf{D_i} - \mathbf{C_i}$ for all i} ≤ 720.
sum of {$\mathbf{K_i} - \mathbf{J_i}$ for all i} ≤ 720.

$0 \le \mathbf{A_C} \le 2$.
$0 \le \mathbf{A_J} \le 2$.
$1 \le \mathbf{A_C} + \mathbf{A_J} \le 2$.

Output

For each test case, output one line containing **Case #x: y,** where x is the test case number (starting from 1) and y the minimum possible number of exchanges, as described in the statement.

Sample Input

```
3
1 1
540 600
840 900
2 0
900 1260
180 540
1 1
1439 1440
0 1
```

Sample Output

```
Case #1: 2
Case #2: 4
Case #3: 2
```

3 - Cats

Mark decided to celebrate his first big salary by going to the restaurant. He lives by an unusual park. The park is a rooted tree consisting of *n* vertices with the root at vertex 1. Vertex 1 also contains Mark's house. Unfortunaely for our hero, the park also contains cats. Mark has already found out what are the vertices with cats in them.

The leaf vertices of the park contain restaurants. Mark wants to choose a restaurant where he will go, but unfortunately he is very afraid of cats, so there is no way he will go to the restaurant if the path from the restaurant to his house contains more than *m* **consecutive** vertices with cats.

Your task is to help Mark count the number of restaurants where he can go.

Note: A *tree* is a connected graph on n vertices and $n - 1$ edges. A *rooted* tree is a tree with a special vertex called *root*. In a rooted tree among any two vertices connected by an edge, one vertex is a parent (the one closer to the root), and the other one is a child. A vertex is called a *leaf*, if it has no children.

Input
The first line of the input contains an integer T (1<=T<=20) denoting the number of test cases. The description of the test cases follows. The first line contains two integers, n and m ($2 \le n \le 10^5$, $1 \le m \le n$) — the number of vertices of the tree and the maximum number of consecutive vertices with cats that is still ok for Mark.
The second line contains n integers $a_1, a_2, ..., a_n$, where each a_i either equals to 0 (then vertex i has no cat), or equals to 1 (then vertex i has a cat).
Next $n - 1$ lines contains the edges of the tree in the format "x_i y_i" (without the quotes) ($1 \le x_i, y_i \le n$, $x_i \ne y_i$), where x_i and y_i are the vertices of the tree, connected by an edge.
It is guaranteed that the given set of edges specifies a tree.

Output
For each test case, output one line containing "**Case #x:" ,** where x is the test case number (starting from 1) and a single integer: the number of distinct leaves of a tree the path to which from Mark's home contains at most *m* consecutive vertices with cats.

Sample input
```
2
4 1
1 1 0 0
1 2
1 3
1 4
7 1
1 0 1 1 0 0 0
1 2
1 3
2 4
2 5
3 6
3 7
```

Sample output
```
Case #1: 2
Case #2: 2
```

4 - Permutations

For this problem, you will write a program that takes a (possibly long) string of decimal digits, and outputs the permutation of those decimal digits that has the next larger value (as a decimal number) than the input number. For example:

123 ------> 132
279134399742 ------> 279134423799

It is possible that no permutation of the input digits has a larger value. For example, 987.

Input
The first line of input contains a single integer P, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by up to 80 decimal digits which is the input value.

Output
For each data set there is one line of output. If there is no larger permutation of the input digits, the output should be the data set number followed by a single space, followed by the string BIGGEST. If there is a solution, the output should be the data set number, a single space and the next larger permutation of the input digits.

Sample input
3
1 123
2 279134399742
3 987

Sample output
1 132
2 279134423799
3 BIGGEST

5- Sources of Light

You are given an axis-aligned rectangle in a 2D Cartesian plane. The bottom left corner of this rectangle has coordinates (0,0) and the top right corner has coordinates (N−1,N−1). You are also given K light sources; each light source is a point inside or on the perimeter of the rectangle.

For each light source, let's divide the plane into four quadrants by a horizontal and a vertical line passing through this light source. The light source can only illuminate one of these quadrants (including its border, i.e. the point containing the light source and two half-lines), but the quadrants illuminated by different light sources may be different.

You want to assign a quadrant to each light source in such a way that when they illuminate their respective quadrants, the entire rectangle (including its perimeter) is illuminated. Find out whether it is possible to assign quadrants to light sources in such a way.

Input
The first line of the input contains an integer T (1≤T≤5000) denoting the number of test cases. The description of the test cases follows.
The first line of each test case contains two space-separated integers K (1≤K≤100) and N ($1 \le N \le 10^9$).
Each of the next K lines contains two space separated integers x and y denoting a light source with coordinates (x,y). (0≤x,y≤N−1, no two light sources coincide)

Output
For each test case, output one line containing "**Case #x:" ,** where x is the test case number (starting from 1) and the string "yes" (lowercase) if it is possible to illuminate the whole rectangle or "no" if it is impossible.

Sample Input
```
2
2 10
0 0
1 0
2 10
1 2
1 1
```

Sample Output
```
Case #1: yes
Case #2: no
```