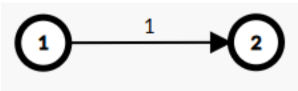
Red serie-paralela

Contribución de Román Castellarin

Descripción del problema

Mediante una secuencia de **N** operaciones, se va a construir una *red* que tendrá en total **N+2** nodos (representados gráficamente mediante círculos). Algunos pares de nodos estarán conectados entre sí mediante flechas **dirigidas**. Los nodos se numeran con enteros positivos, en el orden en que se van creando, y lo mismo ocurre con las flechas.

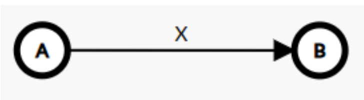
Inicialmente existen solamente dos nodos numerados **1** y **2**, y una única flecha (la número **1**) que va desde el nodo **1** hasta el **2**.



Cada una de las **N** operaciones será de uno de los siguientes dos tipos:

1. Expandir en serie la flecha **X**: Si hasta el momento la red tiene **n** nodos y **s** flechas, y la flecha **X** va desde el nodo **A** hasta el **B**, se agrega un nuevo nodo **n+1** en medio de la flecha **X**, de modo que la flecha **X** pasa a ir desde **n + 1** hasta **B**, y se crea la nueva flecha **s + 1** que va desde **A** hasta **n + 1**.

   Es decir de

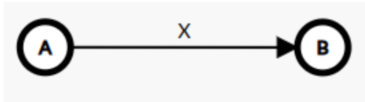
   

   Se pasa a

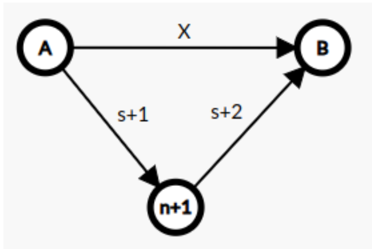
   

2. Expandir en paralelo la flecha **X**: Si hasta el momento la red tiene **n** nodos y **s** flechas, y la flecha **X** va desde el nodo **A** hasta el **B**, se agrega un nuevo nodo **n + 1** y dos nuevas flechas, de modo que la nueva flecha **s + 1** va desde **A** hasta **n + 1**, y la nueva flecha **s + 2** va desde **n + 1** hasta **B**.

   Es decir de

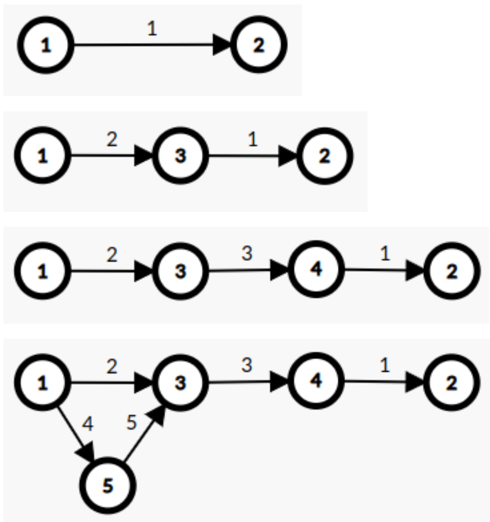
   Se pasa a

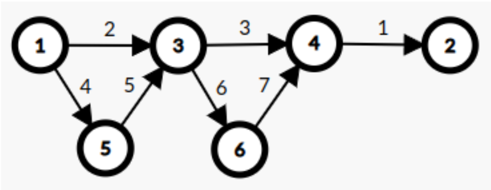
   

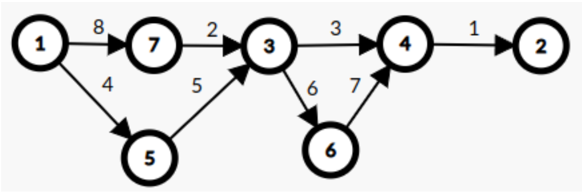
Así, la secuencia de formación de la red es la secuencia de **N** operaciones con la que se construye a partir de la flecha inicial. Por ejemplo, para la siguiente secuencia de formación con **N** = **7**:

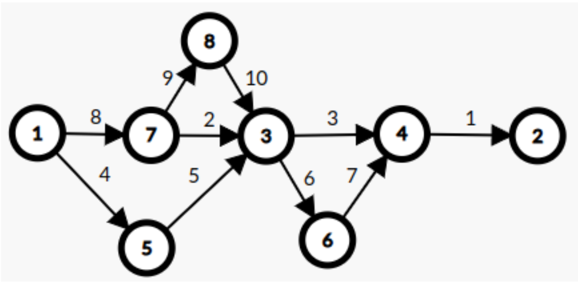
1. Expandir en serie la flecha **1**
2. Expandir en serie la flecha **1**
3. Expandir en paralelo la flecha **2**
4. Expandir en paralelo la flecha **3**
5. Expandir en serie la flecha **2**
6. Expandir en paralelo la flecha **2**
7. Expandir en paralelo la flecha **2**

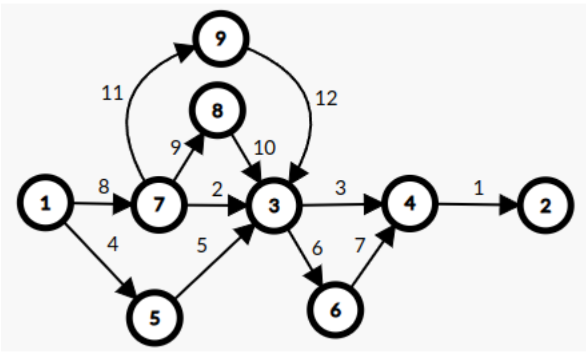
El proceso detallado hasta llegar a la red completa sería el siguiente:











Un *ordenamiento válido* para la red final es un ordenamiento de los enteros desde **1** hasta $\mathbf{N+2}$ tal que para cada flecha de la red, si la flecha va desde **A** hasta **B**, entonces **A** está antes que **B** en el ordenamiento.

Ejemplos de ordenamientos válidos para la red del ejemplo anterior serían $\mathbf{1,5,7,8,9,3,6,4,2}$ y $\mathbf{1,7,8,9,5,3,6,4,2}$, entre otros posibles.

No sería un ordenamiento válido $\mathbf{1,7,8,9,3,5,6,4,2}$, ya que en la red hay una flecha $\mathbf{5 \to 3}$ pero el **3** está antes que el **5** en el ordenamiento, en contra de la flecha.

Dadas las **N** operaciones de la secuencia de formación de la red en orden, debes escribir una función que calcule la cantidad de ordenamientos válidos de los $\mathbf{N+2}$ nodos. Como esta cantidad puede ser muy grande, la función debe retornar el resto de dividir esta cantidad por el entero $\mathbf{10^9+7=1.000.000.007}$.

Descripción de la función

Debes implementar la función `serieparalelo(t,e)`. Sus parámetros son:

- `t`: Un arreglo de **N** enteros. **t[i]** es **1** si la **i**-ésima operación es expandir en serie, y **2** si la **i**-ésima operación es expandir en paralelo.
- `e`: Un arreglo de **N** enteros. **e[i]** es el número de flecha a expandir en la **i**-ésima operación.

La función debe retornar un único entero: el resto resultante de dividir a la cantidad de ordenamientos válidos por $\mathbf{10^9+7=1.000.000.007}$.

Evaluador

El evaluador local lee de la entrada estándar con el siguiente formato:

- El entero **N**
- **N** pares de enteros **t[i]**, **e[i]**

El evaluador local escribe a la salida estándar la respuesta retornada por la función.

Restricciones

- $\mathbf{1 \le N \le 250.000}$
- $\mathbf{1 \le t_i \le 2}$

Ejemplo

Si se invoca al evaluador con la siguiente entrada:

```
7
1 1
1 1
2 2
2 3
1 2
2 2
2 2
```

Para un programa correcto, la salida será:

```
8
```

Si en cambio la entrada es:

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

Para un programa correcto, la salida será:

```
1
```

y si es:

```
4
2 1
2 1
2 1
2 1
```

la salida será:

```
24
```

Subtareas

1. **N** = **1** (3 puntos)
2. **N** = **2** (5 puntos)
3. **N** ≤ **8** (7 puntos)
4. **N** ≤ **16** (12 puntos)
5. **N** ≤ **24** (15 puntos)
6. **N** ≤ **1000** (33 puntos)
7. Sin más restricción (25 puntos)