Repartiendo las Atracciones

En Bakú hay n atracciones, identificadas con números desde 0 hasta $n-1$. También hay m rutas bidireccionales, identificadas con números desde 0 hasta $m-1$. Cada ruta conecta dos atracciones diferentes. Es posible viajar desde cualquier atracción hasta cualquier otra utilizando las rutas.

Fátima planea visitar todas las atracciones en tres días. Ella ha decidido que quiere visitar a atracciones el primer día, b atracciones el segundo día, y c atracciones el tercer día. Por lo tanto, ella va a particionar las n atracciones en tres conjuntos A, B, y C de tamaños a, b, y c, respectivamente. Cada atracción pertenecerá a exactamente un conjunto, de modo que $a+b+c=n$.

A Fátima le gustaría encontrar conjuntos A, B, y C, tales que **al menos dos** de los tres conjuntos sean **conexos**. Se dice que un conjunto S de atracciones es conexo si es posible viajar desde cualquier atracción de S hasta cualquier otra atracción de S utilizando las rutas, y sin pasar por ninguna atracción que no forme parte de S. Una partición de las atracciones en conjuntos A, B, y C se dice **válida** si satisface las condiciones anteriormente mencionadas.

Ayuda a Fátima a encontrar una partición válida de las atracciones (dados a, b, y c), o determina que no existe ninguna partición válida. Si hay múltiples particiones válidas, puedes encontrar cualquiera de ellas.

Detalles de implementación

Debes implementar la siguiente función:

```
int[] find_split(int n, int a, int b, int c, int[] p, int[] q)
```

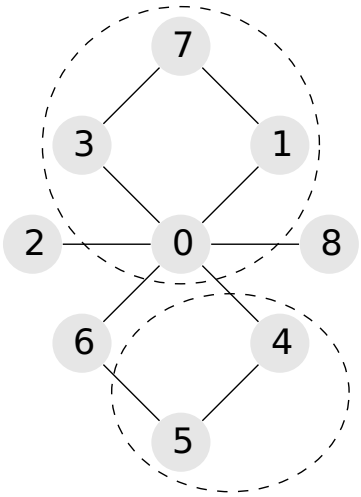
- n: la cantidad de atracciones.
- a, b, y c: los tamaños deseados de los conjuntos A, B, y C.
- p y q: arreglos de longitud m, que contienen los extremos de las rutas. Para cada i ($0 \leq i \leq m-1$), $p[i]$ y $q[i]$ son las dos atracciones que conecta la ruta i.
- Esta función debe retornar un arreglo de longitud n. Llamemos a este arreglo s. Si no existe partición válida, s debe contener n ceros. De lo contrario, para $0 \leq i \leq n-1$, $s[i]$ debe ser 1, 2, o 3 para indicar que la atracción i es asignada al conjunto A, B, o C, respectivamente.

Ejemplos

Ejemplo 1

Considera la siguiente invocación:

```
find_split(9, 4, 2, 3, [0, 0, 0, 0, 0, 0, 1, 3, 4, 5],
                       [1, 2, 3, 4, 6, 8, 7, 7, 5, 6])
```

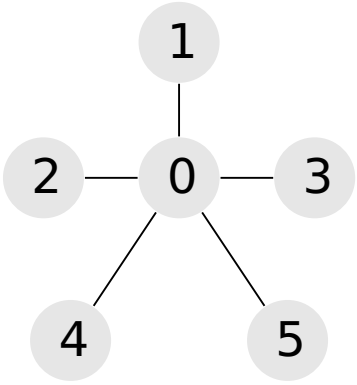


Una solución correcta posible es $[1, 1, 3, 1, 2, 2, 3, 1, 3]$. Esta solución describe la siguiente partición: $A = \{0, 1, 3, 7\}$, $B = \{4, 5\}$, y $C = \{2, 6, 8\}$. Los conjuntos A y B son conexos.

Ejemplo 2

Considere la siguiente invocación:

```
find_split(6, 2, 2, 2, [0, 0, 0, 0, 0], [1, 2, 3, 4, 5])
```



No existe partición válida. Por lo tanto, la única respuesta correcta es $[0, 0, 0, 0, 0, 0]$.

Restricciones

- $3 \leq n \leq 100\,000$
- $2 \leq m \leq 200\,000$
- $1 \leq a, b, c \leq n$
- $a + b + c = n$
- Existe a lo más una ruta entre cada par de atracciones.
- Es posible viajar desde cualquier atracción hasta cualquier otra utilizando las rutas.
- $0 \leq p[i], q[i] \leq n - 1$ y $p[i] \neq q[i]$ para $0 \leq i \leq m - 1$

Subtareas

1. (7 puntos) Cada atracción es extremo de a lo más dos rutas.
2. (11 puntos) $a = 1$
3. (22 puntos) $m = n - 1$
4. (24 puntos) $n \leq 2500, m \leq 5000$
5. (36 puntos) Ninguna restricción adicional.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada con el siguiente formato:

- línea 1: n m
- línea 2: a b c
- línea $3 + i$ (para $0 \leq i \leq m - 1$): $p[i]$ $q[i]$

El evaluador de ejemplo imprime una única línea, que contiene el arreglo retornado por la función `find_split`.