Conectando los Supertrees (supertrees)

Los Jardines de la Bahía son un gran parque natural de Singapur. En el parque hay n torres, llamadas "supertrees". estas están indexadas de 0 a $n-1$. Hay que construir **cero o más** puentes. Cada puente conecta un par de torres distintas y puede recorrerse en **ambas** direcciones. No puede haber dos puentes que conecten las mismas dos torres.

Un camino desde la torre x a la torre y es una sucesión de una o más torres tales que:

- la primer torre de la sucesión es x,
- la última torre es y,
- todas las torres de la sucesión son **distintas**, y
- cada par de torres consecutivas en la sucesión están conectadas por un puente.

Nótese que con esta definición hay exactamente un camino entre una torre y sí misma. Además, la cantidad de caminos diferentes desde la torre i a la torre j será igual que la cantidad de caminos diferentes desde la torre j a la torre i.

La arquitecta encargada del diseÒo del parque quiere construir los puentes de tal forma que para todo $0 \leq i, j \leq n-1$ haya exactamente $p[i][j]$ caminos diferentes desde la torre i a la torre j, donde $0 \leq p[i][j] \leq 3$.

Construye un conjunto de puentes que satisfagan los requerimientos de la arquitecta, o determina que es imposible.

Detalles de Implementación

Tienes que implementar la siguiente función:

```
int construct(int[][] p)
```

- p: un arreglo bidimensional de $n \times n$ que representa los requerimientos de la arquitecta.
- Si se pueden satisfacer los requerimientos, esta función debe llamar exactamente una vez a `build`, diciendo la forma de construir los puentes (ver más abajo). Luego debe devolver 1.
- De **no** ser posible, la función debe devolver 0 sin hacer ninguna llamada a `build`.
- Esta función va a ser llamada exactamente una vez.

La función `build` se define así:

```
void build(int[][] b)
```

- b: un arreglo bidimensional de $n \times n$, con $b[i][j] = 1$ si hay un puente que conecta las torres i y j, o $b[i][j] = 0$ si no.
- Nótese que b debe cumplir que $b[i][j] = b[j][i]$ para todo $0 \leq i, j \leq n-1$ y $b[i][i] = 0$ para todo $0 \leq i \leq n-1$.

Ejemplos

Ejemplo 1

Consideremos la llamada siguiente:

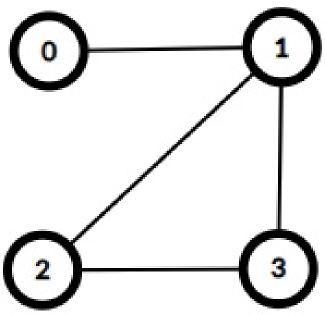
```
construct([[1, 1, 2, 2], [1, 1, 2, 2], [2, 2, 1, 2], [2, 2, 2, 1]])
```

Esto significa que tiene que haber exactamente un camino entre la torre 0 y la 1. Para todo otro par de torres (x, y), tales que $0 \leq x < y \leq 3$, tiene que haber exactamente dos caminos entre la torre x y la y.

Esto se puede satisfacer construyendo 4 puentes, entre los pares de torres $(0, 1)$, $(1, 2)$, $(1, 3)$ y $(2, 3)$.

Para decir la solución, `construct` debe llamar a `build` de la siguiente manera:

- `build([[0, 1, 0, 0], [1, 0, 1, 1], [0, 1, 0, 1], [0, 1, 1, 0]])`



Y luego debe devolver 1.

En este caso, hay más de una solución que satisface los requerimientos. Todas son consideradas correctas.

Ejemplo 2

Consideremos la llamada siguiente:

```
construct([[1, 0], [0, 1]])
```

Esto significa que no debe haber ningún camino entre ambas torres. Esto sólo se puede dar si no hay ningún puente.

Entonces `construct` tiene que hacer la siguiente llamada:

- `build([[0, 0], [0, 0]])`

Y luego debe devolver 1.

Ejemplo 3

Consideremos la llamada siguiente:

```
construct([[1, 3], [3, 1]])
```

Esto significa que tiene que haber exactamente 3 caminos entre la torre 0 y la 1. No hay forma de construir puentes tal que se cumpla esto.

Por eso, `construct` debe devolver 0 sin hacer ninguna llamada a `build`.

Cotas

- $1 \le n \le 1000$
- $p[i][i] = 1$ (para todo $0 \le i \le n-1$)
- $p[i][j] = p[j][i]$ (para todo $0 \le i, j \le n-1$)
- $0 \le p[i][j] \le 3$ (para todo $0 \le i, j \le n-1$)

Subtareas

1. (11 puntos) $p[i][j] = 1$ (para todo $0 \le i, j \le n-1$)
2. (10 puntos) $p[i][j] = 0$ o 1 (para todo $0 \le i, j \le n-1$)
3. (19 puntos) $p[i][j] = 0$ o 2 (para todo $i \neq j$, $0 \le i, j \le n-1$)
4. (35 puntos) $0 \le p[i][j] \le 2$ (para todo $0 \le i, j \le n-1$) y siempre se pueden satisfacer los requerimientos.
5. (21 puntos) $0 \le p[i][j] \le 2$ (para todo $0 \le i, j \le n-1$)
6. (4 puntos) Sin restricciones adicionales.

Evaluador Local

El evaluador local lee la entrada con el siguiente formato:

- línea 1: n
- línea $2+i$ ($0 \le i \le n-1$): $p[i][0]\ \ p[i][1]\ \ \ldots\ \ p[i][n-1]$

La salida del evaluador local es con el siguiente formato:

- línea 1: lo que devuelve la función `construct`.

Si lo que devuelve `construct` es 1, el evaluador local además imprime:

- línea $2 + i$ ($0 \le i \le n-1$): $b[i][0] \;\; b[i][1] \;\; \ldots \;\; b[i][n-1]$