Conectando los superárboles (supertrees)

Gardens by the Bay es un gran parque natural en Singapur. En el parque hay n torres, conocidas como superárboles. Estas torres están etiquetadas de 0 a $n-1$. Nos gustaría construir un conjunto de **cero o más** puentes. Cada puente conecta un par de torres distintas y puede ser atravesado en **cualquier** dirección. No hay dos puentes que conecten el mismo par de torres.

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

- el primer elemento de la secuencia es x,
- el último elemento de la secuencia es y,
- todos los elementos de la secuencia son **distintos**, y
- cada dos elementos consecutivos (torres) en la secuencia están conectados por un puente.

Note que por definición hay exactamente un camino desde una torre a sí misma y el número de diferentes caminos desde la torre i a la torre j es el mismo que el número de diferentes caminos desde la torre j a la torre i.

El arquitecto principal a cargo del diseño desea que los puentes se construyan de tal manera que para todos los $0 \leq i, j \leq n-1$ haya exactamente $p[i][j]$ caminos diferentes desde la torre i hasta la torre j, donde $0 \leq p[i][j] \leq 3$.

Construir un conjunto de puentes que satisfagan los requerimientos del arquitecto, o determinar que es imposible.

Detalles de la implementación

Debería implementar el siguiente procedimiento:

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

- p: un arreglo de $n \times n$ representando los requisitos del arquitecto.
- Si una construcción es posible, este procedimiento debe hacer exactamente una llamada a `build` (ver abajo) para informar de la construcción, tras lo cual debe retornar 1.
- De lo contrario, el procedimiento debería devolver 0 sin hacer ninguna llamada a `build`.
- Este procedimiento se llama exactamente una vez.

El procedimiento `build` está definido de la siguiente manera:

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

- b: un arreglo $n \times n$, con $b[i][j] = 1$ si hay un puente conectando la torre i y la torre j, o $b[i][j] = 0$ en caso contrario.
- Note que el arreglo debe satisfacer $b[i][j] = b[j][i]$ 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

Considere la siguiente llamada:

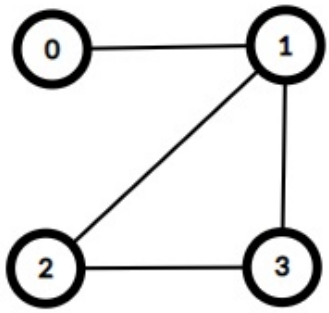
```
construct([[1, 1, 2, 2], [1, 1, 2, 2], [2, 2, 1, 2], [2, 2, 2, 1]])
```

Esto significa que debería haber exactamente un camino desde la torre 0 a la torre de 1. Para todos los otros pares de torres (x, y), tales como $0 \leq x < y \leq 3$, debería haber exactamente dos caminos desde la torre x a la torre y.

Esto se puede lograr con 4 puentes, conectando pares de torres $(0, 1)$, $(1, 2)$, $(1, 3)$ y $(2, 3)$.

Para reportar esta solución, el procedimiento `construct` debe hacer la siguiente llamada:

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



Entonces debería retornar 1.

En este caso, hay múltiples construcciones que se ajustan a los requisitos, todas las cuales se considerarían correctas.

Ejemplo 2

Considere la siguiente llamada:

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

Esto significa que no debería haber forma de viajar entre las dos torres. Esto sólo se puede satisfacer si no hay puentes. Por lo tanto, el procedimiento de `construct` debería hacer la

siguiente llamada:

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

Después de lo cual, el procedimiento `construct` debería retornar 1.

Ejemplo 3

Considere la siguiente llamada:

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

Esto significa que debe haber exactamente 3 caminos desde la torre 0 hasta la torre 1. Este conjunto de requisitos no puede ser satisfecho. Por lo tanto, el procedimiento `construct` debería devolver 0 sin hacer ninguna llamada a `build`.

Restricciones

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

Subtareas

1. (11 puntos) $p[i][j] = 1$ (para todo $0 \leq i, j \leq n-1$)
2. (10 puntos) $p[i][j] = 0$ o 1 (fpara todo $0 \leq i, j \leq n-1$)
3. (19 puntos) $p[i][j] = 0$ o 2 (para todo $i \neq j$, $0 \leq i, j \leq n-1$)
4. (35 puntos) $0 \leq p[i][j] \leq 2$ (para todo $0 \leq i, j \leq n-1$) y hay al menos una construcción que cumple con los requisitos.
5. (21 puntos) $0 \leq p[i][j] \leq 2$ (para todo $0 \leq i, j \leq n-1$)
6. (4 puntos) No hay restricciones adicionales.

Grader de ejemplo

El grader de ejemplo lee la entrada en el siguiente formato:

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

La salida del grader de ejemplo tiene el siguiente formato:

- línea 1: el valor de retorno de `construct`.

Si el valor de retorno de `construct` es 1, el grader de ejemplo imprime adicionalmente:

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