Ligando Super-Árvores (supertrees)

Os Jardins da Baía são um grande parque natural em Singapura. No parque existem n torres, conhecidas como super-árvores. Estas torres são numeradas de 0 a $n-1$. Nós queremos construir um conjunto de **zero ou mais** pontes. Cada ponte liga um par de torres distintas e pode ser atravessada em **ambas** as direções. Não podem existir duas pontes a ligar o mesmo par de torres.

Um caminho da torre x para a torre y é uma sequência de uma ou mais torres tal que:

- o primeiro elemento da sequência é x,
- o último elemento da sequência é y,
- todos os elementos da sequência são **distintos**, e
- cada dois elementos consecutivos (torres) na sequência estão ligados por uma ponte.

Nota que, por definição, existe exatamente um caminho de uma torre para si própria e a quantidade de caminhos diferentes da torre i para a torre j é a mesma que a quantidade de caminhos diferentes da torre j para a torre i.

O arquiteto principal que tem a seu cargo esta obra deseja que as pontes sejam construídas de tal modo que para todos os $0 \leq i, j \leq n-1$ existam exatamente $p[i][j]$ caminhos diferentes da torre i para a torre j, onde $0 \leq p[i][j] \leq 3$.

O teu objetivo é construir um conjunto de pontes que satisfaça os requisitos do arquiteto, ou determinar que tal é impossível.

Detalhes de implementação

Deves implementar a seguinte função:

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

- p: uma matriz $n \times n$ representando os requisitos do arquiteto.
- Se a construção for possível, esta função deve fazer exatamente uma chamada `build` (ver em baixo) para reportar a construção, e de seguida deve devolver 1 como valor de retorno.
- Caso contrário, a função deve devolver 0 sem fazer nenhuma chamada a `build`.
- Esta função é chamada exatamente uma vez.

A função `build` é definida da seguinte maneira:

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

- b: uma matriz $n \times n$, com $b[i][j] = 1$ se existir uma ponte ligando a torre i à torre j, ou $b[i][j] = 0$ caso contrário.
- Nota que a matriz deve satisfazer $b[i][j] = b[j][i]$ para todos os $0 \leq i, j \leq n - 1$ e $b[i][i] = 0$ para todo o $0 \leq i \leq n - 1$.

Exemplos

Exemplo 1

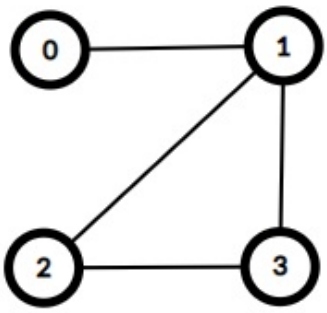
Considera a seguinte chamada:

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

Isto significa que deve existir exatamente um caminho da torre 0 para a torre 1. Para todos os outros pares de torres (x, y), tais que $0 \leq x < y \leq 3$, devem existir exactamente dois caminhos da torre x para a torre y. Isto pode ser obtido com 4 pontes, ligando os pares de torres $(0, 1)$, $(1, 2)$, $(1, 3)$ e $(2, 3)$.

Para reportar esta solução, a função `construct` deve fazer a seguinte chamada:

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



No final desta chamada deve também devolver 1.

Neste caso existem múltiplas construções que obedecem aos requisitos, e qualquer uma delas seria considerada correta.

Exemplo 2

Considera a seguinte chamada:

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

Isto significa que não deve existir nenhuma maneira de viajar entre as duas torres. Isto apenas pode ser satisfeito se não existir nenhuma ponte.

Desse modo, a função `construct` deve fazer a seguinte chamada:

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

Depois disto, a função `construct` deve devolver 1.

Exemplo 3

Considera a seguinte chamada:

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

Isto significa que devem existir exatamente 3 caminhos da torre 0 para a torre 1. Este conjunto de requisitos não pode ser satisfeito. Como tal, a função `construct` deve devolver 0 sem fazer nenhuma chamada a `build`.

Restrições

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

Subtarefas

1. (11 pontos) $p[i][j] = 1$ (para todos os $0 \leq i, j \leq n-1$)
2. (10 pontos) $p[i][j] = 0$ ou 1 (para todos os $0 \leq i, j \leq n-1$)
3. (19 pontos) $p[i][j] = 0$ ou 2 (para todos os $i \neq j, 0 \leq i, j \leq n-1$)
4. (35 pontos) $0 \leq p[i][j] \leq 2$ (para todos os $0 \leq i, j \leq n-1$) e existe pelo menos uma construção que satisfaz os requisitos.
5. (21 pontos) $0 \leq p[i][j] \leq 2$ (para todos os $0 \leq i, j \leq n-1$)
6. (4 pontos) Nenhuma restrição adicional.

Avaliador exemplo

O avaliador exemplo lê o input no seguinte formato:

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

O avaliador exemplo escreve o output no seguinte formato:

- linha 1: o valor de retorno de `construct`.

Se o valor de retorno de `construct` for 1, o avaliador exemplo imprime também:

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