# Supertrees Verbinden (supertrees)

Gardens by the Bay ist ein großer Naturpark in Singapur. Im Park stehen $n$ Türme, bekannt als Supertrees. Diese Türme sind nummeriert von $0$ bis $n-1$. Das Ziel ist es, **keine oder mehrere** Brücken zu bauen. Eine Brücke verbindet zwei verschiedene Türme und kann in **beiden** Richtungen durchquert werden. Es ist nicht erlaubt, dass zwei Brücken die gleichen zwei Türme verbinden.

Ein Pfad von Turm $x$ nach Turm $y$ ist definiert als Folge von einem oder mehreren Türmen, sodass:

- das erste Element der Folge $x$ ist,
- das letzte Element der Folge $y$ ist,
- alle Elemente der Folge **verschieden** sind, sowie
- zwei aufeinanderfolgende Elemente (Türme) in der Folge durch eine Brücke verbunden sind.

Beachte, dass es nach dieser Definition genau einen Pfad von einem Turm zu sich selbst gibt und es gleich viele Pfade von Turm $i$ zu Turm $j$ gibt wie von Turm $j$ zu Turm $i$.

Der Hauptarchitekt des Parks möchte, dass die Brücken so gebaut werden, dass es für alle $0 \leq i, j \leq n-1$ genau $p[i][j]$ verschiedene Pfade von Turm $i$ zu Turm $j$ gibt, wobei $0 \leq p[i][j] \leq 3$.

Konstruiere eine Menge von Brücken, welche die Bedingungen des Architekten erfüllt oder stelle fest, dass es keine solche gibt.

## Implementierungsdetails

Du sollst die folgende Funktion implementieren:

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

- $p$: ein $n \times n$-Array mit den Bedingungen des Architekten.
- Falls die Bedingungen erfüllbar sind, soll die Funktion `build` genau einmal aufgerufen werden (siehe unten), um die Lösung einzureichen. Als Rückgabewert soll anschließend $1$ zurückgegeben werden.
- Andernfalls soll die Funktion $0$ zurückgeben, ohne `build` aufzurufen.
- Die Funktion wird genau ein Mal aufgerufen.

Die Funktion `build` ist wie folgt definiert:

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

- $b$: ein $n \times n$-Array, mit $b[i][j] = 1$, falls eine Brücke zwischen Turm $i$ und Turm $j$ gebaut werden soll, oder $b[i][j] = 0$ andernfalls.
- Das Array soll $b[i][j] = b[j][i]$ für alle $0 \leq i, j \leq n - 1$ und $b[i][i] = 0$ für alle $0 \leq i \leq n - 1$ erfüllen.

# Beispiele

## Beispiel 1

Betrachte den folgenden Funktionsaufruf:

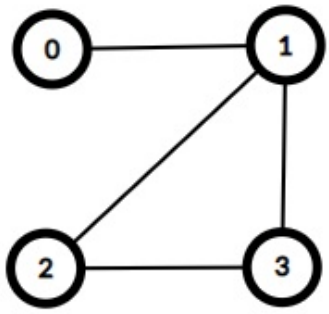
```
construct([[1, 1, 2, 2], [1, 1, 2, 2], [2, 2, 1, 2], [2, 2, 2, 1]])
```

Dies bedeutet, dass es genau einen Pfad zwischen Turm $0$ und Turm $1$ geben soll. Für alle anderen Paare von Türmen $(x, y)$ mit $0 \leq x < y \leq 3$ soll es genau zwei Pfade von Turm $x$ zu Turm $y$ geben.

Dies kann durch 4 Brücken erreicht werden, welche die Turmpaare $(0, 1)$, $(1, 2)$, $(1, 3)$ und $(2, 3)$ verbinden.

Um diese Lösung einzureichen, soll die Funktion `construct` den folgenden Funktionsaufruf tätigen:

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




Dann soll die Funktion `construct` den Wert $1$ zurückgeben.

In diesem Beispiel gibt es mehrere Konstruktionen, welche die Bedingungen erfüllen. Alle würden als korrekt bewertet werden.

## Beispiel 2

Betrachte den folgenden Funktionsaufruf:

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

Dies bedeutet, dass es nicht möglich sein soll, von Turm $0$ zu Turm $1$ zu gelangen. Das kann nur

dann erfüllt sein, wenn keine Brücken gebaut werden.

Daher soll die Funktion `construct` den folgenden Funktionsaufruf machen:

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

Anschließend soll die Funktion `construct` den Wert $1$ zurückgeben.

### Beispiel 3

Betrachte den folgenden Funktionsaufruf:

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

Dies bedeutet, dass es genau 3 Pfade von Turm $0$ zu Turm $1$ geben soll. In diesem Beispiel kann man die Brücken nicht so bauen, dass alle Bedingungen erfüllt sind. Daher soll die Funktion `construct` den Wert $0$ zurückgeben, ohne die Funktion `build` aufzurufen.

## Beschränkungen

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

## Teilaufgaben

1. (11 Punkte) $p[i][j] = 1$ (für alle $0 \leq i, j \leq n-1$).
2. (10 Punkte) $p[i][j] = 0$ oder $1$ (für alle $0 \leq i, j \leq n-1$).
3. (19 Punkte) $p[i][j] = 0$ oder $2$ (für alle $i \neq j$, $0 \leq i, j \leq n-1$).
4. (35 Punkte) $0 \leq p[i][j] \leq 2$ (für alle $0 \leq i, j \leq n-1$) und es gibt mindestens eine Konstruktion, welche alle Bedingungen erfüllt.
5. (21 Punkte) $0 \leq p[i][j] \leq 2$ (für alle $0 \leq i, j \leq n-1$).
6. (4 Punkte) Keine weiteren Beschränkungen.

## Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

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

Die Ausgabe des Beispiel-Graders hat das folgende Format:

- Zeile $1$: Der Rückgabewert von `construct`.

Falls der Rückgabewert von `construct` $1$ war, gibt der Beispiel-Grader zusätzlich folgende Werte aus:

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