# Connecting Supertrees (supertrees)

Gardens by the Bay este un mare parc natural din Singapore. În parc sunt $n$ turnuri, cunoscute ca supercopaci. Aceste turnuri sunt numerotate de la $0$ la $n-1$. Vrem să construim o mulțime constând în **zero sau mai multe** poduri. Fiecare pod conectează o pereche de turnuri distincte și poate fi traversat în **ambele** direcții. Nu este permis ca două poduri să conecteze aceeași pereche de turnuri.

Un drum de la un turn $x$ la un turn $y$ este o secvență formată din unul sau mai multe turnuri astfel încât:

- primul element din secvență este $x$,
- ultimul element din secvență este $y$,
- toate elementele din secvență sunt **distincte**, și
- fiecare două elemente consecutive (turnuri) din secvență sunt conectate printr-un pod.

Observați că, prin definiție, există exact un drum de la un turn la el însuși și că numărul de drumuri de la turnul $i$ la turnul $j$ este exact același ca numărul de drumuri de la turnul $j$ la turnul $i$.

Arhitectul sef responsabil cu designul își dorește ca podurile să fie construite astfel încât oricare ar fi $0 \leq i, j \leq n-1$ să existe exact $p[i][j]$ drumuri distincte de la turnul $i$ la turnul $j$, unde $0 \leq p[i][j] \leq 3$.

Construiți o mulțime de poduri care satisfac condițiile arhitectului șef, sau determinați că acest lucru este imposibil.

## Detalii de implementare

Trebuie să implementați următoarea funcție:

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

- $p$: o matrice de dimensiuni $n \times n$ respectând condițiile arhitectului șef.
- Dacă o construcție este posibilă, funcția va apela exact o dată funcția `build` (vezi mai jos) pentru a raporta construcția, după care va returna $1$.
- Altfel, procedura va returna $0$, fără să apeleze niciodată funcția `build`.
- Această funcție va fi apelată exact o dată.

Funcția `build` este definită dupa cum urmează:

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

- $b$: o matrice de dimensiuni $n \times n$, unde $b[i][j] = 1$ semnifică existența unui pod ce conectează turnurile $i$ și $j$, altfel avem $b[i][j] = 0$.
- Observați că această matrice trebuie să satisfacă $b[i][j] = b[j][i]$ oricare ar fi $0 \leq i, j \leq n - 1$ si $b[i][i] = 0$ oricare ar fi $0 \leq i \leq n - 1$.

# Exemple

## Exemplu 1

Să considerăm următorul apel:

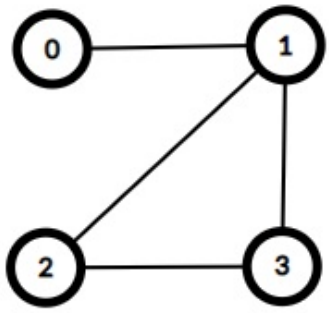
```
construct([[1, 1, 2, 2], [1, 1, 2, 2], [2, 2, 1, 2], [2, 2, 2, 1]])
```

Acesta înseamnă că trebuie să existe exact un drum de la trunul $0$ la turnul $1$, iar pentru toate celelalte perechi de turnuri $(x, y)$ astfel încât $0 \leq x < y \leq 3$, trebuie să existe exact două drumuri de la turnul $x$ la turnul $y$.

Condițiile arhitectului șef pot fi întrunite construind $4$ poduri, conectand perechile de turnuri $(0, 1)$, $(1, 2)$, $(1, 3)$ si $(2, 3)$.

Pentru a raporta aceasta solutie, functia `construct` trebuie sa faca urmatorul apel:

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




Si apoi sa returneze $1$.

În acest caz există mai multe construcții care respectă condițiile, toate fiind considerate corecte.

## Exemplu 2

Să considerăm următorul apel:

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

Acesta înseamnă că nu trebuie să fie nicio modalitate de a călători de la un turn la celălalt. Singura metodă de a satisface condițiile în acest caz este să nu construim niciun pod.

Prin urmare, funcția `construct` trebuie să facă următorul apel:

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

După care funcția `construct` va returna $1$.

### Exemplu 3

Să considerăm următorul apel:

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

Acesta înseamnă că trebuie să existe exact $3$ drumuri între turnul $0$ și turnul $1$. În acest caz, condițiile nu pot fi satisfăcute. Prin urmare, funcția `construct` trebuie să returneze $0$ fără să apeleze niciodată funcția `build`.

## Restricții

- $1 \leq n \leq 1000$
- $p[i][i] = 1$ (oricare ar fi $0 \leq i \leq n-1$)
- $p[i][j] = p[j][i]$ (oricare ar fi $0 \leq i, j \leq n-1$)
- $0 \leq p[i][j] \leq 3$ (oricare ar fi $0 \leq i, j \leq n-1$)

## Subtaskuri

1. (11 puncte) $p[i][j] = 1$ (oricare ar fi $0 \leq i, j \leq n-1$)
2. (10 puncte) $p[i][j] = 0$ sau $1$ (oricare ar fi $0 \leq i, j \leq n-1$)
3. (19 puncte) $p[i][j] = 0$ sau $2$ (oricare ar fi $i \neq j$, $0 \leq i, j \leq n-1$)
4. (35 de puncte) $0 \leq p[i][j] \leq 2$ (oricare ar fi $0 \leq i, j \leq n-1$) și există cel puțin o construcție care satisface condițiile.
5. (21 de puncte) $0 \leq p[i][j] \leq 2$ (oricare ar fi $0 \leq i, j \leq n-1$)
6. (4 puncte) Fără restricții suplimentare.

## Sample grader

Graderul citește inputul în următorul format:

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

Outputul graderului are următorul format:

- linia $1$: valoarea returnată de `construct`.

Daca valoarea returnată de `construct` este $1$, atunci graderul mai afișează următoarele:

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