FTRAVELAB

Đoàn thanh tra gồm hai thành viên A và B cần lần lượt đi kiểm tra n công ty. Để có thể sử dụng các thông tin của một số công ty trong việc kiểm tra các công ty quan trọng và việc thanh tra được tiến hành nhanh chóng, họ sẽ phân công các công ty cho nhau sao cho:

- Mỗi công ty chỉ được kiểm tra bởi đúng một thanh tra.
- Các thanh tra có thể kiểm tra các công ty theo thứ tự lần lượt từ công ty 1 đến công ty ***n***.
- Tổng quãng đường di chuyển của hai thanh tra là ít nhất.

Lưu ý do tính chất đặc thù của ngành thanh tra, để di chuyển đến công ty ***t*** sau khi vừa kiểm tra xong công ty ***s***, các thanh tra sẽ phải đi theo một đường đi đã được quy định trước. Đường đi này chưa chắc đã là đường đi ngắn nhất giữa hai công ty **s** và **t**.

Gọi các giao điểm theo thứ tự mà A thanh tra là a_1, a_2, ..., a_x các giao điểm theo thứ tự mà B thanh tra là b_1, b_2, ..., b_y, ta có:

- Mỗi số nguyên từ 1 đến n xuất hiện trong dãy a_1, a_2, ..., a_x , b_1, b_2, ..., b_y đúng một lần.

- $a_1 < a_2 < ... < a_x$

- $b_1 < b_2 < ... < b_y$

- Gọi d(s, t) là quãng đường di chuyển từ công ty s đến công ty t theo đường đi đã được quy định trước, thì $d(a_1, a_2) + d(a_2, a_3) + ... + d(a_{x-1}, a_x) + d(b_1, b_2) + d(b_2, b_3) + ... + d(b_{y-1}, b_y)$ nhỏ nhất.

Yêu cầu: Tính tổng quãng đường di chuyển ngắn nhất có thể của cả hai thanh tra.

Dữ liệu:

- Dòng đầu tiên gồm một số nguyên dương n ($1 \le n \le 1000$).
- n dòng tiếp theo, dòng thứ i, số nguyên thứ j ($a[i,j] \le 10^3$) là quãng đường di chuyển từ công ty i đến công ty j theo đường đi đã được quy định trước.

Dữ liệu vào đảm bảo nếu i = j thì $a_{ij} = 0$.

Kết quả

- Gồm một dòng duy nhất chứa một số nguyên là tổng quãng đường di chuyển ngắn nhất của cả hai thanh tra

FTRAVELAB.INP	FTRAVELAB.OUT
4 0 2 3 4 2 0 1 5 3 2 0 7 8 6 5 0	3

Giải thích: Người A: lần lượt đi kiểm tra 1 → 2→ 3; Người B đi kiểm tra: 4

```
#include <bits/stdc++.h>
#define fi "FTRAVELAB.INP"
#define fo "FTRAVELAB.OUT"
using namespace std;
const int N = 1e3+5;
int n, a[N][N];
int f[N][N], dd[N][N];

void doc(){
	cin>>n;
	for(int i=1; i<=n; i++)
		for(int j=1; j<=n; j++){
				cin>>a[i][j];
				dd[i][j] = 0;
		}

}

// goi f[x][y] = chi phi it nhat khi xuat phat tu (x, y), A xp tu x, B xp tu y, di het ts n
// QHD: xet aint TH
int tinh(int x, int y){
	if(dd[x][y] == 1) return f[x][y];
	dd[x][y] = 1;
	if(max(x, y) == n) f[x][y] = 0;
	else{
		int z = max(x, y) + 1;
		f[x][y] = min(tinh(x, z) + a[y][z], tinh(z, y) + a[x][z]);
	}
	return f[x][y];
}

void xuly(){
	cout<<tinh(0, 0);
}

int main(){
//	freopen(fi, "r", stdin);
//	freopen(fo, "w", stdout);
	ios_base :: sync_with_stdio(false);
	cin.tie(0);	cout.tie(0);

	doc();
	xuly();
	return 0;
}
```