Algorithm Design Laboratory with Applications

Prof. Stefano Leucci

Problem: *Che$$board.*

There is an $n \times n$ chessboard in which each square has some non-negative amount of coins placed on it. More precisely, the square at coordinates (i, j) with $i, j \in \{1, \ldots, n\}$ has $c_{i,j} \in \mathbb{N}$ one-dollar coins placed on it.

Your must place any n rooks (the chess pieces) on distinct squares the chessboard so that no two distinct rooks attack one another (a rook attacks another if they are placed on the same row or on the same column). When you place a rook in square (i, j) you collect the $c_{i,j}$ coins on (i, j).

Design an algorithm that, given n and the amount of coins $c_{i,j}$ on each square (i, j), returns the maximum amount of dollars that you can collect.

Input. The input consists of a set of instances, or *test-cases*, of the previous problem. The first line contains the number T of test-cases. The first line of each test case contains the integer n. The i-th of the following n lines describes the i-th row of the chessboard and contains the values $c_{i,1}, c_{i,2}, \ldots, c_{i,n}$.

Output. The output consists of T lines. The i-th line is the answer to the i-th test-case and contains an integer corresponding to the maximum amount of dollars that can be collected.

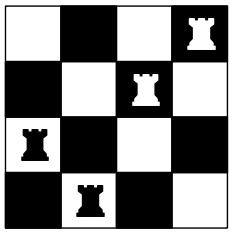
Assumptions. $1 \leq T \leq 10$; $1 \leq n \leq 2^9$; $\forall i, j \in \{1, \ldots, n\}\ c_{i,j} \leq 2^{10}$.

Example.

Instance

2	1	0	5
0	0	3	0
6	0	8	0
0	4	2	1

Solution



Input (the first test-case corresponds to the example above):

```
1
4
2 1 0 5
0 0 3 0
6 0 8 0
0 4 2 1
```

Output:

```
18
```

Requirements. Your algorithm should require $O(n^3)$ time (with reasonable hidden constants).

Notes. A reasonable implementation should not require more than 5 seconds for each input file.