/ 20 P

Programming Task P1.

Enrollment Key: `ExamTime2019`

Submission: see Section 3 of the Technical Guide

Grid

Given is a square grid of size $N \times N$. Rows and columns are indexed from 0 to $N-1$ and each cell at index (row, column) contains a positive value (its cost) as shown in the example below.

You are asked to traverse the grid from the top row to the bottom row, one row at a time. In each step, you can move from a cell (i, j) in row i to either of the cells $(i+1, j-1)$, $(i+1, j)$, or $(i+1, j+1)$ in row $i+1$ as depicted below. As start you can pick any cell in the first row. The overall cost of a traversal is the sum of the costs of all visited cells, including start and end.

Devise an algorithm to find a path from top to bottom of the grid that minimizes the overall traversal cost.

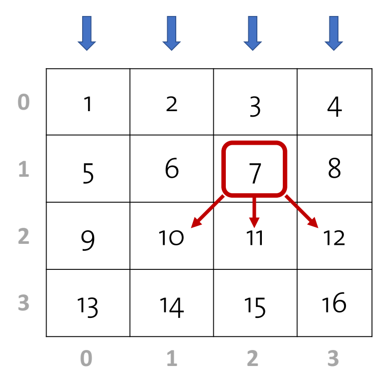
Complete the implementation of the following method:

- **`solveGrid(`***grid***`)`**: calculates the minimal cost to traverse the grid from top to bottom.

To test your implementation, the computed cost is written to the output. Your implementation can assume an $N \times N$ grid, where $1 \leq N \leq 10^3$, and the cost of each cell is an integer p with $1 \leq p \leq 10^6$.

Example

Consider a 4×4 grid, as illustrated below. You can start the traversal at each of the 4 columns, beginning at cell $(0, 0)$, $(0, 1)$, $(0, 2)$ or $(0, 3)$.



From each cell (i, j) you can move to cell $(i+1, j)$, $(i+1, j-1)$ or $(i+1, j+1)$. For example from cell $(1, 2)$, you can either move to cell $(2, 1)$, $(2, 2)$ or $(2, 3)$, as illustrated above.

The minimal cost to reach the bottom of the grid is through cells $(0, 0) \rightarrow (1, 0) \rightarrow (2, 0) \rightarrow (3, 0)$, and the price is $1 + 5 + 9 + 13 = 28$.

Grading

Overall, you can obtain a maximum of 20 judge points for this programming task. Assuming $N \times N$ is the size of the grid, you can obtain up to:

- **20 points** for time $O(N^2)$ solution.
- **15 points** for time $O(N^2 \cdot \log(N))$ solution.
- **10 points** for time $O(N \cdot 3^N)$ solution.

We assume that each solution is provided with reasonable hidden constants.

Instructions

For this exercise, we provide a program template as an Eclipse project in your workspace that helps you reading the input and writing the output. Importing any additional Java class is **not allowed** (with the exception of the already imported ones `java.io.{InputStream, OutputStream}` and `java.util.Scanner` class).

The project also contains data for your local testing and a `JUnit` program that runs your `Main.java` on all the local tests – just open and run `GridTest.launch` in the project. The local test data are different and generally smaller than the data that are used in the online judge.

Submit only your `Main.java`.

The input and output are handled by the template – you should not need the rest of this text.

Input The input of this problem consists of a number of test-cases. The first line contains T, the number of test-cases. Each of the T cases is independent of the others, and contains several lines:

1. The first line of each test case contains the amount of rows/columns of the grid $N \in \{1, \ldots, 10^3\}$.
2. Then it is followed by $N \times N$ numbers, that correspond to the the price $p \in \{1, \ldots, 10^6\}$ at each cell in the grid. The numbers are distributed in N lines, each having N integers separated by one or several white space characters. Each such line corresponds to one row of the grid.

Output For every case, the output is the minimal cost of the traversing the grid from top to bottom.

The output contains one line for each test-case. More precisely, the i-th line of the output contains an integer number that represents the minimal cost of traversing the grid from top to bottom. The output is terminated with an end-line character.

Example input:

```
2
4
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  16
4
14  7   9   16
4   10  12  13
5   1   2   15
8   6   11  3
```

Example output:

```
28
18
```