1. All Pairs Shortest Paths

Write your own algorithm which finds shortest distances between every pair of vertices in a given edge weighted directed graph G(V, E) with no negative cycle.
The given graphs will be represented as adjacency matrices, which means that the value of matrix[i][j] is the weight w_{ij} from vertice i to j where w_{ij} is in $\mathbb{Z} \setminus \{0\}$.
(the matrix is a NxN square matrix when N is the size of the vertex set V.)

Then you should print out sum of the distances between every pair % 100,000,000.

[Constraints]
- The number of vertices is in [1, 200].
- The number of edges is in [1, 10,000].
- Optimization options are not allowed at compilation.

Note : You may get penalised if the run-time of your algorithm is significantly longer than others.

[File]
Write your implementations in “Solution1.java” file with “Solution1” class.
When you run the code with the following command, the output file “output1.txt” should be generated.
> javac Solution1.java -encoding UTF8 && java Solution1

[Input]
An input file “input1.txt” will be given, which has 10 test cases.
Each case consists of 2 lines; first is N and the latter is E pieces of information about the edges as source, vertex, weight with space in each.

The indices of V starts from 1 and weights will be between -1000 and 1000

inclusively with no case of 0.

[Output]

For each case, you should print the case number as #x where x is the index of the case. Then, print sum(all distances) mod 100,000,000.

You must produce your results as “output1.txt”

[Example]

Input (input1.txt)

```
2 2                                                    ← 1st case
1 2 100 2 1 -50
3 3                                                    ← 2nd case
1 2 100 2 3 -50 3 1 30
…
```

Output (output1.txt)

```
#1 50
#2 240
…
```