... it is important to realize that any lock can be picked with a big enough hammer.

`Sun System & Network Admin Manual`

My appartment has n computers. My friend's appartment also has n computers. In each appartment, some pairs of computers are connected to each other with AcidNet cables (ignoring the routers). Each connection has a certain bandwidth (in bytes per second). My friend always brags about the speed of his computer network. He always shows me his n-by-n table that lists the bandwidths between each pair of computers. My network is slower, and I want to rebuild it. So I want to know how I should connect my computers in order to have the same n-by-n bandwidth table.

Since I don't want to buy too many AcidNet cables, you'll need to find a solution with the minimum number of connections. You may use AcidNet cables of any integer bandwidth — they all have the same price at my local Imaginary Hardware Store.

Given a graph, you can compute the all-pairs maximum flow table, right? Now do the opposite: given an n-by-n symmetric table, find a graph with fewest edges that has the given table of all-pairs maximum flows.

Input

The first line of input gives the number of cases, N. N test cases follow. Each one is a line containing n ($0 < n \le 200$), followed by n lines with n integers each, giving the table T.

- $T[u][u]$ will always be 0.
- $T[u][v]$ will always be positive and equal to $T[v][u]$.
- $T[i][j] \le 10000$

$T[u][v]$ is the largest possible speed (in bytes per second) for sending information from computer u to computer v, assuming there is no other traffic on the network.

Output

For each test case, output one line containing '`Case #`x`:`' followed by m — the number of cables I have to buy. The next m lines will each contain 3 integers u, v and w meaning that I need to connect computer u to computer v using an AcidNet cable of bandwidth w. Computers are numbered starting at 0.

If there is no solution, print '`Impossible`'.

Sample Input

```
4
2
0 10
10 0
3
0 1 1
1 0 2
1 2 0
1
0
4
0 2 2 1
2 0 2 2
2 2 0 2
1 2 2 0
```

Sample Output

```
Case #1: 1
0 1 10
Case #2: 2
0 1 1
1 2 2
Case #3: 0
Case #4: Impossible
```