**CS-302 Design and Analysis of Algorithms**

**Project**

**PART I (DUE 27th April 2020)**

My apartment has n computers. My friend’s apartment also has n computers. In each apartment, 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 ≤ 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] ≤ 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

You can develop the understanding of flows in the network by consulting section 26.1 from the textbook. This question is taken from onlinejudge ([link](https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=567&page=show_problem&problem=2650)). You may know that **UVa Online Judge** is an [online automated judge](https://en.wikipedia.org/wiki/Online_judge) for programming problems. There is an automated system that evaluates the submissions. You are required to make an online account on onlinejudge and submit your code. The automated system will test the code and provide you with acceptance/rejection report.

You are required to check your code using online judge and submit the evaluation snapshot by the onlinejudge.org along with your code to the submission portal at google classroom

.

**Part II (Due 5th May 2020)**

Each student has been assigned a research paper (See table below). You are encouraged to work in a group for this part. You have to understand the paper (background, research problem, solution, implication, code) and make a 10 minute presentation (preferably in a group). The presentations will be conducted from May 5th Onwards.

Section F

|  |  |
| --- | --- |
| [Best Nearly Sorted Sorting Algorithm](https://drive.google.com/open?id=1ftlwn1NkrbwmhByse45eY3aTYPniAUdH) | 15I-0303,  16I-0237  18I-0424  18I-0444  18I-0445  18I-0472 |
| [Randomized Speedup of](https://drive.google.com/open?id=1MDqnwOIja72EoMDyrzJjNVYnKpYVtEUc)  [the Bellman{Ford Algorithm](https://drive.google.com/open?id=1MDqnwOIja72EoMDyrzJjNVYnKpYVtEUc)  Algo 4 | 18I-0489  18I-0500  18I-0505  18I-0523  18I-0555 |
| [Randomized Speedup of](https://drive.google.com/open?id=1MDqnwOIja72EoMDyrzJjNVYnKpYVtEUc)  [the Bellman{Ford Algorithm](https://drive.google.com/open?id=1MDqnwOIja72EoMDyrzJjNVYnKpYVtEUc)  Algo 5 | 18I-0489  18I-0500  18I-0505  18I-0523  18I-0555 |
| ﻿[A K-Means++-Based](https://drive.google.com/open?id=1MKJsBP9Otjw2GtiqRnSlnZDbDVcvV00o)  [Minimum Spanning Tree Algorithm](https://drive.google.com/open?id=1MKJsBP9Otjw2GtiqRnSlnZDbDVcvV00o) | 18I-0559  18I-0582  18I-0590  18I-0606  18I-0620 |
| [An MST Heuristic Algorithm to](https://drive.google.com/open?id=1MKhzt1qTE-2678KgWu8Wr1pS26GLPpwf)  [Approximate Solutions for ATSP’s](https://drive.google.com/open?id=1MKhzt1qTE-2678KgWu8Wr1pS26GLPpwf) | 18I-0632  18I-0699  18I-0704  18I-0705  18I-0715 |
| ﻿[An Improved Computation of the PageRank Algorithm](https://drive.google.com/open?id=1zjO3sBWxabM3zwFe5hT8QqIbcBk1S7yu) | 18I-0786  18I-0835  18I-1561  18I-1658  18I-1587 |

Section G

|  |  |
| --- | --- |
| [Zig-zag Sort: A Simple Deterministic Data-Oblivious Sorting Algorithm Running in O(n log n) Time](https://drive.google.com/open?id=1mNW3S9tl47UBjArUWImR-zYWQ653hwRo) | 17I-0161  18I-0416  18I-0421  18I-0460  18I-0680 |
| [A Simple Min-Cut Algorithm](https://drive.google.com/open?id=1MLBvIik14-I9_jdhR9d6kmZmJlPOJ2os) | 18I-0513  18I-0522  18I-0562  18I-0574  18I-0577 |
| [A Fast Algorithm to Find All-Pairs Shortest Paths in Complex Networks](https://drive.google.com/open?id=1sJUocRzDU4JUJ_V3NwUGymhrJPe2xjKm) | 18I-0587  18I-0615  18I-0658  18I-0659  18I-0671 |
| [A Faster Algorithm for Betweenness Centrality∗](https://drive.google.com/open?id=1R2twR-sfz_tQMIVhQlkCm0IXuy7XMqcw) | 18I-0695  18I-0732  18I-0744  18I-0800  18I-1565 |

**Part III (Due 18th May 2020)**

You are supposed to implement the algorithm in your respective paper, show its correctness with 5 concrete examples and analyze its running time. The implementation part is supposed to be an individual effort and will be subjected to plagiarism check.