Problem Bangye
Task review
Time Limit: *5 Seconds*

To design the programming tasks for 2055 ACM ICPC Taipei site, there is a committee of N members. Each member has designed a task, and thus there are N tasks in total. For the quality of the game, the tasks must be reviewed before they can be used. Now, the question is how to assign the review work. Greg, as the chair of the committee, decides that an assignment is *feasible* if and only if each task is reviewed by one member and each member reviews exactly one task. Of course, any one cannot review his/her own task. To find a good review assignment, Greg models it as the following optimization problem.

The members are labelled from 0 to $N-1$. The task designed by member i is called task i. Greg creates a directed graph $G = (V, E, w)$, where $V = \{0, 1, \ldots, N-1\}$ denotes the members, as well as the tasks. If member i can review task j, then there is an edge (arc) from i to j with weight $w(i, j)$, where the weight is the difficulty of this review. For a directed graph $G = (V, E, w)$, an edge subset F is a *cycle cover* if these edges form some disjoint cycles and cover all the vertices, i.e., each vertex has in-degree and out-degree one in the subgraph induced by F. Thus, each feasible assignment corresponds to one cycle cover.

For example, Figure 1.(a) shows all possible review edges: member 0 can review task 1 with difficulty 4; member 0 can review task 2 with difficulty 8; member 1 can review task 0 with difficulty 5; and so on. The assignments in Figure 1.(b) is not feasible since it is not a cycle cover, and both assignments in (c) and (d) are feasible.

However, there may be more than one feasible assignment. If the difficulties are too small, the members feel boring. But if the difficulties are too large, the loads are heavy. For a set of edges F, let $\max(F) = \max_{e \in F} w(e)$ and $\min(F) = \min_{e \in F} w(e)$. To find an assignment to make the committee happy, Greg defines the following measures. For a cycle cover C, the load is defined by

$$L(C) = \sum_{e \in E, w(e) \leq \max(C)} w(e);$$

the boringness is defined by

$$B(C) = \sum_{e \in E, w(e) \geq \min(C)} w(e);$$

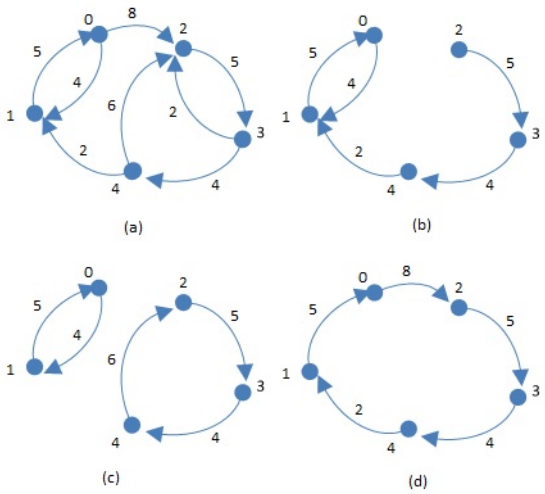

Figure 1: Examples of review assignments.

and the unhappyness is given by

$$S(C) = \max\{L(C), B(C)\}.$$

The goal of this problem is to find a cycle cover with minimum unhappyness. Let C and D be the cycle covers in Figure 1.(c) and (d), respectively. We have $\max(C) = 6$, $\min(C) = 4$, $\max(D) = 8$, and $\min(D) = 2$. Thus, $L(C) = 2 + 2 + 4 + 4 + 5 + 5 + 6 = 28$, $B(C) = 4 + 4 + 5 + 5 + 6 + 8 = 32$, and therefore $S(C) = 32$. For D, we have $L(D) = B(D) = S(D) = 36$. So C has a smaller unhappyness and thus is better than D.

Technical Specification

1. The number of vertices (committee member) N is between 2 and 500.
2. The difficulty for each task review, i.e. the edge weight $w(i, j)$, is between 1 and 10000.

Input File Format

The first line of the input file contains an integer T, $1 \leq T \leq 8$, which

indicates the number of test cases. The first line of each test case contains two integer N and M, where M is the number of edges. The following M lines contain the data of edges, one line for one edge. The data of each edge consists of three integer i, j, and $w(i, j)$. Any two consecutive numbers in the same line are separated by a space.

Output Format

For each test case, output the minimum unhappyness in one line. If there is no feasible assignment, then output -1.

Sample Input

```
2
5 8
0 1 4
0 2 8
2 3 5
3 4 4
3 2 2
1 0 5
4 2 6
4 1 2
3 2
0 1 10
1 2 6
```

Output for the Sample Input

```
32
-1
```

1 Solution

A cycle cover in a directed graph $G = (V, E)$ is equivalent to a perfect matching in the corresponding bipartite graph $H = (V \cup V, E)$. Thus, for an

edge subset $|F|$, it can be determined in $O(\sqrt{N}|F|)$ time if there is a cycle cover.

The objective function (unhappyness) of this task is determined by $\max(F)$ and $\min(F)$ for a feasible edge subset F (containing a cycle cover). That is, we in fact need to find a "best weight interval" $[x, y]$ such that the edge subset $E_{x,y} = \{e \in E \mid w(e) \in [x, y]\}$ is feasible and minimizes the unhappyness.

The first observation is that for a fixed x, the unhappyness is non-decreasing in y as long as it is feasible. Therefore, we can always pick the smallest feasible y.

Suppose all the possible edge weights are $w_1, w_2, \ldots, w_W$. A naive algorithm of trying all weight intervals is surely too bad. A better, but not good enough, algorithm works as follows: Start with $x = w_1$ and find the smallest feasible y. For i-th iteration, we increase x from w_i to w_{i+1} and search the best y from its current weight, that is, not necessarily going back and from w_{i+2}. This algorithm uses only $O(W)$ passes for finding perfect matching. However, $O(W\sqrt{N}|F|)$ will be not good enough. Note that this algorithm is better than the method that linear searches for x and binary searches for y.

Since the best y can be thought of as a function of x, the unhappyness is a function of x. The second crucial observation is that the load $L(x)$ is a non-decreasing function and the boringness $B(x)$ is a strictly decreasing function. Therefore, the unhappyness $S(x)$ is bitonic (strictly decreasing and then non-decreasing), or convex. The minimum of a bitonic function can be found by 3-ry search. For each searched x, the best y can be determined by binary search. Thus we takes $O(\log^2 W)$ passes.

Remarks: For 3-ry searching the minimum of function f in a possible range $[p, q]$, we test the points $x_1 = (2p + q)/3$ and $x_2 = (p + 2q)/3$. If $f(x_1) \leq f(x_2)$, the minimum is in range $[p, x_2]$; and otherwise in $[x_1, q]$. Note the equality does matter: Since the function is decreasing and then non-decreasing, for $p \leq x_1 < x_2 \leq q$, if $f(x_1) = f(x_2)$, then the minimum is in range $[p, x_2]$ but not the other side.

2 Test Plan

An optimal algorithm and a linear-pass algorithm. Test cases:

- an infeasible data (no perfect matching).
- random data with upper bound ($n = 500, W = 10000$).

- The optimal weight interval is very wide.
- The best x is small.
- The best x is large.

Checking data:

- no multiple edges.
- upper and lower bound of weights, number of vertices.
- checking solution: check if the output is a perfect matching. check the cost by scanning the original edge data.