Task Bribe 2

⏳ 0 sec. 💾 0 MB

The task is output-only. More detailed information is described at the end of the statement.

Bobi has a connected undirected graph without multiple edges, but this time he knows what it is and unfortunately he will not have to bribe the committee in order to find it out! The similarity, however, is that Bobi will again be looking for a special spanning tree of his graph.

Bobi's goal is to find a spanning tree of the graph which is as close as possible to having a total edge weight greater than that of 67% of all spanning trees. Since there may be many spanning trees with equal weights, for a given spanning tree we count half of the spanning trees with its total weight as ones with a smaller total weight. The scoring is described in more detail in section *Scoring*.

Write a program **`bribe2`** which finds this spanning tree.

Input

Each test consists of t subtests. The first line contains t. The first line of each subtest contains N and M – respectively the number of vertices and edges in the graph. The next M lines contain three numbers u_i, v_i, w_i, which means that there is a bidirectional edge between u_i and v_i with weight w_i.

Output

On t lines of the standard output print $N - 1$ numbers each – the indices of the edges forming the found spanning tree for each subtest. The indices start from 1.

Constraints

- $1 \leq t \leq 10$
- $1 \leq N \leq 5\ 000$
- $1 \leq M \leq 10\ 000$
- $1 \leq u_i, v_i \leq N$, $u_i \neq v_i$
- $1 \leq w_i \leq 10\ 000$

Scoring

The result for each test is equal to the minimum of the results of the subtests. The result of a subtest is computed in the following way:

Let $\mathcal{T}$ be the set of all distinct spanning trees of the graph, $k = |\mathcal{T}|$, and the total weight of a spanning tree T be $w(T) = \sum_{e \in T} w(e)$.

Let T be the spanning tree printed by you and let

$$p = \frac{|\{T' \in \mathcal{T} \ : \ w(T') < w(T)\}| + \frac{1}{2}\,|\{T' \in \mathcal{T} \ : \ w(T') = w(T)\}|}{k},$$

i.e. p is the fraction of the spanning trees in the graph whose total weight is smaller than yours, where the spanning trees with the same total weight as yours are counted with weight $\frac{1}{2}$.

If T is not a valid spanning tree, the result for the subtest is 0. Otherwise the result is

$$s = \max\left(0,\ 1 - \frac{|p - 0.67|}{0.33}\right).$$

Uploading the output files

The name of the output file for the test with number xx must be

test.xx.out

For tests from 1 to 9 the number is written with a leading zero. For example, the output for the eighth test must be in the file test.08.out, and the output for the tenth – in the file test.10.out.

You upload to the system a ZIP archive which contains at least one output file. The output files must be located directly in the archive, and not in a subdirectory. Files with other names are ignored. If a file for a given test is missing from the archive, this upload does not change your result for that test.

Every upload of an archive counts as one submission, regardless of how many output files the archive contains. If you upload outputs for the same test in more than one submission, the best result you have achieved on that test is taken for the final standing. An invalid output gives 0 points for the respective test, but has no effect if you have better results for the same test from a previous submission.

For convenience, you can use the following code, which opens each of the test files in the respective stream (standard input/output) and then calls a function `solve`:

```
for(int test = 1; test <= 10; test++) {
    string s = (test<10?"0":"") + to_string(test);
    freopen(("test."+s+".in").c_str(), "r", stdin);
    freopen(("test."+s+".out").c_str(), "w", stdout);
    solve();
}
```

This way the function `solve` can read the input data via `cin` and print the output data via `cout`.