Edge Searching

Poisonous gas has spread through a tunnel complex modeled by graph *G*. You have been put in charge of a team of mobile agents whose goal is to clean the contaminated complex. Initially, the entire graph *G* is contaminated. The gas cannot spread past your agents and you can clean an edge, by moving an agent along it. However, if at any point of the process there is an unprotected path (without any agents) containing both clean and contaminated edges, then the former become recontaminated. This is strictly forbidden and should it happen, your mission will fail.

The action of only one agent can be performed at a time. The following actions are available:

- Adding an agent on any vertex of *G*
- Removing and agent from any vertex of *G*
- Moving an agent along any edge in *G*

A contaminated edge e={u,v} becomes clean if either:

- There is an agent at *u* and another agent is moved from *u* to *v*
- There is an agent at *u* and *e* is the only contaminated edge incident to *u*; then *e* becomes clean if the agent from *u* moves along it.

Your goal is to clean *G* using as few agents as possible.

Input

At the beginning of the input there is a single positive integer *i*. After that, *i* test cases follow. Each test consists of a graph *G* that is to be searched. The first line of each case is a single positive integer *n*. Graph *G* has *n* vertices. Than all subsequent lines contain a pair of integers *u v*. Each such line means that an edge *u v* exists in *G*. The list of edges ends with a line in which *u*=0 and *v*=0.

All graphs *G* are connected, undirected and simple (there are no loops or parallel edges).

The majority of tested graphs have less than 5000 vertices, but that is not true for all of them.

Output

The output of each test case should consist of series of agent moves, describing search strategy. Each line should start with a single character from the set {a, r, m, d}, which correspond to: adding an agent, removing an agent, moving an agent, and claiming that the task has been completed, respectively.

- a – A leading *a* should be followed by single space character and a positive integer *v*, 0 <= *v* < n. An agent is placed on vertex *v*.
- r - A leading *r* should be followed by single space character and a positive integer *v*, 0 <= *v* <*n*. An agent is removed from vertex *v*.
- m - A leading *m* should be followed by single space character and two positive integers *u*, *v*, separated by a single space character. 0 <= *u* <*n* and 0<= *v* <*n*. An agent is moved from *u* to *v*. If there was no agent at *u* this will result in a Wrong Answer message.

- d – The input is finished. If all edges are clean the case is completed successfully. Otherwise, a Wrong Answer message is issued.

If any of the moves leads to recontamination a Wrong answer message will be issued. Note, that moving an agent among *e* is also considered a recontamination, if the move does not clean *e*.

Scoring

Your goal is to minimize the number of agents used during the cleaning process, that is the maximal total number of agents present on *G* at any point of the cleaning process. If an agent is removed from graph and than added to it again, the total number of agents does not increase.

The number of points given in the ranking is scaled so that it is equal to 10 for the contestant whose solution has the smallest score, and is proportionally less for all solutions with larger scores.

Example

Input

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

Output:

```
a 2
a 2
m 2 1
m 1 0
r 0
m 2 3
m 3 4
m 4 5
d

a 0
a 1
a 2
a 3
a 4
```

```
a 0
m 0 1
r 1
a 0
m 0 2
r 2
a 0
m 0 3
r 3
a 0
m 0 4
r 4
a 1
m 1 2
r 2
a 1
m 1 3
r 3
a 1
m 1 4
r 4
a 2
m 2 3
r 3
a 2
m 2 4
m 4 3
d
```

Scoring:
2 + 6 = 8

Tips

Any graph may be cleaned by placing *n+1* agents, one on each vertex of *G*, and than using the *n+1-th* agent to clean all the edges (see the second example).

There are four classes of test cases:

- In the first, "easy" graph classes, such as paths or cycles, are tested.
- The second class consists of trees.
- The third class consists of structured graphs. Graphs in that class have certain restrictions, such as limited width, etc.
- The last class consists of several big random graphs (up to 16 000 vertices).

Please note

- Till the last week of the series, all submitted codes will be visible to all users and tested on selected data sets only (small instances of classes described above).
- For the last week of the series, submissions will be visible to the submitting contestant, only, and tested on the full set of test cases. All earlier solutions will be rejudged.