Graphs

Max. Marks: 5

This problem is no longer available for practice. Apology for any inconvenience!

A road network graph of a city is modelled as a collection of vertices and edges and stored as an adjacency list. All edges have a name, length (int), maximum speed (v_{max}, int) and the vertex pair (string) that they connect. The cost of traversal on the edge is given by distance/effective speed. The edges can be of the following types:

- *motorway*: additionally, have toll price, number of lanes, maintenance level. The effective speed is given by:

$$v_{eff}^{motorway} = v_{max}\left(1 - \frac{occupancy}{(max\ occupancy)(no\ of\ lanes)}\right)$$

  The maximum occupancy is a constant (100).

- *pedestrianRoad*: additionally, have width of road, scenic value, current occupancy). The effective speed is given by:

$$v_{eff}^{pedestrianRoad} = v_{max}\left(1 - \frac{occupancy}{max\ occupancy}\right)$$

RECENT SUBMISSIONS

DEVELOPERS	RESULT	LANGUAGE
Believer	✓	Java 8
Believer	✓	Java 8
Akhil	✓	Java 8
Akhil	!	Java 8
Akhil	!	Java 8
Sushant Singh	✓	Java 8
Guillotine	✓	Java 8

View All

?

The maximum occupancy is a constant (1500).

- *cyclistRoad*: additionally, have average curvature. The effective speed is given by:

$$v_{eff}^{cyclist\ Road} = \frac{v_{max}}{curvature}$$

- *swamps*: additionally have walking level difficulty. The effective speed is given by:

$$v_{eff}^{cyclist\ Road} = \frac{v_{max}}{(difficulty)^2}$$

- *lakes*: additionally have width, tidal level, and depth. The effective speed is the same as the maximum speed.

You are given *n* edges. The graph is assumed to be bidirectional. There may be multiple roads of different types between the same set of vertices. Print all edges in a sorted order. First sort all edges from the "from vertex" name. Out of all the outgoing edges, sort the edges as per the cost in an increasing order. In case of a tie, the secondary criterion is the names of all the edges.

Two costs (*a* and *b*) are stated as equal based on a precision till the 4th place of decimal, i.e. *a*=*b* iff abs(*a*-*b*)<0.0001

Input Format: The first input is *T*, the number of test cases. Thereafter, the next input is *n*, the number of edges. Thereafter, each line mentions the from vertex name, to vertex name, type of road (as above), name of road, length of road, speed of road. Additionally, if type is motorways, the toll price, number of lanes, maintenance level and occupancy is mentioned. Similarly, if type is pedestrianRoad, the width of the road, scenic value and occupancy is given. If the type is cyclistRoad, the curvature is given. If the type is swamps, the difficulty is given. Instead if the type is lakes, the width of the lake, tidal level and depth is given.

Output Format: In the mentioned sorted order, for every edge print the from vertex name, to vertex name, name of the road, length of the road, and maximum speed. Thereafter, print the specific details of every road in the same order as the input.

SAMPLE INPUT

```
1
12
k04c4 wzgqr motorway pw619 19 18 197 3 3 192
k04c4 wzgqr swamps vivsj 41 5 2
2n23c k04c4 pedestrianRoad 08x9p 103 3 5 82 1373
k04c4 2n23c motorway evnfm 42 22 564 2 2 92
2n23c k04c4 cyclistRoad sxtn3 56 5 8
i2bni 2n23c pedestrianRoad ucpd1 23 2 1 91 1255
wzgqr 2n23c motorway a2e3l 60 19 696 1 2 34
i2bni k04c4 swamps d5a64 106 5 2
i2bni 2n23c lakes z8x2r 66 2 3 3 16
i2bni k04c4 motorway kbia6 101 24 300 1 5 86
i2bni k04c4 pedestrianRoad yey1w 91 4 2 3 573
i2bni wzgqr lakes t5xle 96 1 8 5 10
```

SAMPLE OUTPUT

```
2n23c k04c4 evnfm 42 22 564 2 2 92
2n23c wzgqr a2e3l 60 19 696 1 2 34
2n23c i2bni z8x2r 66 2 3 3 16
2n23c i2bni ucpd1 23 2 1 91 1255
2n23c k04c4 sxtn3 56 5 8
2n23c k04c4 08x9p 103 3 5 82 1373
i2bni k04c4 kbia6 101 24 300 1 5 86
i2bni 2n23c z8x2r 66 2 3 3 16
i2bni k04c4 yey1w 91 4 2 3 573
i2bni 2n23c ucpd1 23 2 1 91 1255
i2bni k04c4 d5a64 106 5 2
i2bni wzgqr t5xle 96 1 8 5 10
k04c4 wzgqr pw619 19 18 197 3 3 192
k04c4 2n23c evnfm 42 22 564 2 2 92
k04c4 i2bni kbia6 101 24 300 1 5 86
k04c4 wzgqr vivsj 41 5 2
k04c4 i2bni yey1w 91 4 2 3 573
k04c4 i2bni d5a64 106 5 2
k04c4 2n23c sxtn3 56 5 8
k04c4 2n23c 08x9p 103 3 5 82
```

Time Limit:	2.0 sec(s) for each input file.
Memory Limit:	256 MB
Source Limit:	1024 KB
Marking Scheme:	Marks are awarded if any testcase passes.
Allowed Languages:	Java 8

```
1373
wzgqr k04c4 pw619 19 18 197
3 3 192
wzgqr 2n23c a2e3l 60 19 696
1 2 34
wzgqr k04c4 vivsj 41 5 2
wzgqr i2bni t5xle 96 1 8 5
10
```

CODE EDITOR

4
LIVE EVENTS

Enter your code or Upload your code as file. Java 8 (oracle 1.8.0_131)

Save

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.util.*;
5
6  import static java.lang.Math.abs;
7
8  class Vertex{
9      private String name;
10     private String road;
11     private String type;
12     private int LengthOfRoad;
13     private int SpeedOfRoad;
14     private double cost;
15     private double EffectiveSpeed;
16
17     String getName() {
18         return name;
19     }
20
21     void setName(String name) {
22         this.name = name;
23     }
24
25     String getRoad() {
26         return road;
27     }
```

?

1:1

Press Ctrl-space for autocomplete suggestions (accuracy dependent on connection stability).

Provide custom input

COMPILE & TEST SUBMIT

Your Rating:

View all comments

4

LIVE EVENTS

?

+1-650-461-4192

contact@hackerearth.com

Resources

- Tech Recruitment Blog
- Product Guides
- Developer hiring guide
- Engineering Blog
- Developers Blog
- Developers Wiki
- Competitive Programming
- Start a Programming Club
- Practice Machine Learning

Solutions

- Assess Developers
- Conduct Remote Interviews
- Assess University Talent
- Organize Hackathons

Company

- About Us
- Press
- Careers

Service & Support

- Technical Support
- Contact Us

4 LIVE EVENTS

?