Practical Assignment 1, A&D, 2025-2026

Problem Statement

Every year the *International Four Day Marches Nijmegen* or also the *Vierdaagse* is organised in Nijmegen. This is the largest multi-day marching event in the world with over 40.000 participants. During the same week, a large free festival called the *Vierdaagsefeesten* is held with many stages throughout the city. The stages are connected by roads. Some roads can be used by buses, some by pedestrians and some by both. See for example the undirected graph in Figure 1.

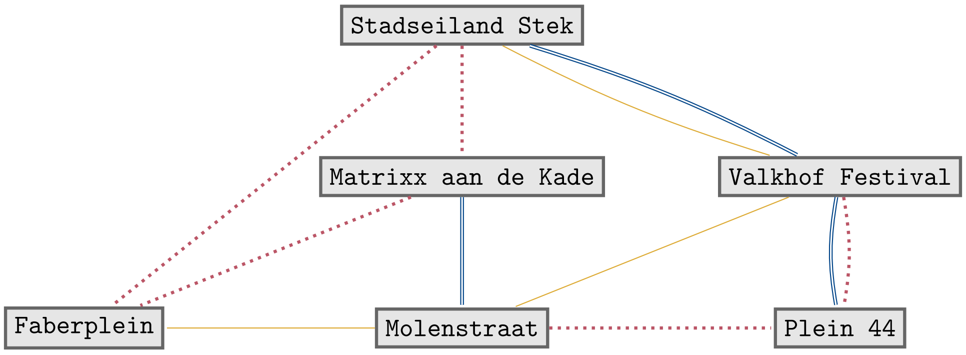

Figure 1: Map of *Vierdaagsefeesten* stages with the edges: *pedestrian* (dotted), *bus* and *both* (double line).

The municipality wants to close roads during the *Vierdaagsefeesten* to control the traffic and prevent overcrowding. Design an algorithm that computes the **maximum number of roads** they can close such that all the stages are still reachable from any other stage when either only travelling by bus or only walking.

Example

For the map displayed in Figure 1, at most four roads can be removed. Figure 2 displays the roads that may be removed in dashed gray lines.

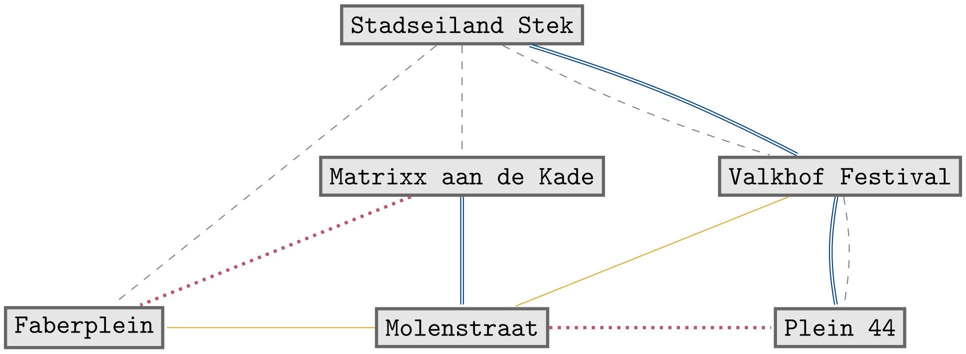

Figure 2: Solution for Figure 1 where edges that are removed are dashed.

Instructions

You may work in groups of two. For this assignment, you have to hand in two things:

- Source code. We will run the code for grading.
- A report. In the report, you have to explain the algorithm and analyze it.

Source code

- You are allowed to submit a solution in either C++, Java or Python 3. If you prefer another language, please contact teaching assistant Christos Aronis asap via Discord or `christos.aronis@ru.nl` (no guarantees!).
- All code for the assignment should be in one file.
- You must write all code yourself, copying code from fellow students or the internet is considered plagiarism. You can only use libraries that come with a minimal installation of the language.
- Hand in clear and well-documented code using sensible variable and function names.

DomJudge You can test your code using DOMjudge. For each assignment, there are two problems on DOMjudge: a practice version and a grading version. The practice version shows the input and expected output so that you can debug any errors. You can submit your code to the practice version as many times as you need. If you feel confident in your solution, you can submit it to the grading version. If necessary, you can submit multiple times to the grading version as well, but only your **last** submission is saved.

Note that testing code takes quite some processing power. Hence, submitting often will cause long wait times and should be avoided. Also, the wait times will probably get longer as the deadline gets near, so be sure to start on time.

Report Besides handing in code, a report should be handed in where you explain your algorithms and analyse their correctness and runtime complexity. The report determines almost half of the final grade! We expect a clearly written high-level explanation of how your algorithm works, and a convincing analysis of correctness and asymptotic complexity. The report can have **at most 6 pages and should follow the template** provided in *template.tex*.

Submission The deadline for sending in your solution is **November 3rd**. You must submit your **report** via **Brightspace** as a group assignment for the group "Practical 1" (for which all team members need to enroll, even if you submit alone!). You must submit your **source code** via **DOMjudge**. Only one team member has to submit a solution; the names and student numbers of both team members must be mentioned in the report.

Grading Grades will be determined as follows. You may earn up to 100 points for your solution:

- 21 points for the explanation of your algorithm.
- 12 points for the correctness analysis.
- 12 points for the complexity analysis.
- 50 points for the test results.
- 5 points for the quality of the code.

If you have any questions, first contact your workgroup TA. For more difficult or technical questions, do not hesitate to contact Christos Aronis via Discord (there is a special channel practical-assignments) or via email `christos.aronis@ru.nl`. You may also contact course coordinator Jurriaan Rot via `jurriaan.rot@ru.nl`.

Input

- The first line contains the following two numbers:
  - $1 \leq S \leq 10000$, the number of stages.
  - $1 \leq R \leq< 50000$, the number of roads between stages.
- The following R lines each represent one road containing:
  - stage s_0 with $1 \leq s_0 \leq S$,
  - stage s_1 with $1 \leq s_1 \leq S$,
  - $T \in \{0, 1, 2\}$ where $0 \mapsto$ *pedestrian*, $1 \mapsto$ *bus* and $2 \mapsto$ *both.*

  Note that the graph described by the stages and roads is undirected.

Output Return the maximum number of edges that can be removed and the minimum cost of travelling to all stages with the bus after removing the edges. If not all stages are reachable when travelling only by bus or only by walking return -1.

Sample inputs and outputs The sample input and outputs corresponding to Example 1:

Input	Output

6 11
0 1 0
0 4 0
1 4 0
3 5 0
2 5 0
0 2 1
2 3 1
3 4 1
0 2 2
2 5 2
1 3 2
	4

Input	Output

3 3
0 1 0
0 2 1
1 2 2
	0

Input	Output

3 3
0 1 0
0 2 1
1 2 1
	-1