Combinatorial optimization
Cocontest semester project assignment:
Smart city with not so smart sewers

Industrial Informatics Research Center
http://industrialinformatics.fel.cvut.cz/

February 22, 2021

Abstract

This document introduces the assignment for the Cocontest semester project.

1 Motivational example

In an unnamed city in the mid-east part of Europe, the sewer system was modernized. To increase the flow of incentives from the EU parliament, the city council wanted the city to become "Smart" — every sewer tube now contains various sensors and cameras. Everything can be monitored through the web interface. Even though the extra cost compared to the immense benefits of such features was negligible, the total cost of the system was, in fact, astronomical. Thus, other much less important non-essential features like the sewer cleaning system were omitted during the realization.

When the time came to do the first cleaning, the "volunteers" group was selected from the nearby fire department to go into the sewer and clean it by pumping water to the tubes manually. Unfortunately for the firemen, a prominent extravagant architect Penrose was hired for the project. His works were famous for over-complicated and very often physically impossible constructions. Using his own extension of Euclidean geometry, this unsung genius designed a system of tubes defying even the laws of physics. Driven by striving for the impossible, he introduced the concept of Penrose stairs[1] into the sewer system. In the sewer system made of tubes with a certain slope, the water could actually return in a cycle back to the original place, which would be otherwise impossible. Although this design seemed genial at first glance, it turned out it brings some complications to the firemen. The water — now enriched by the sewer contents — was heading their way from a different tube. Unfortunately for them, the control system operating the smart sewer system happened to be powered by a sophisticated AI, which produced somewhat less sophisticated decisions. Thus, the poor firemen were mercilessly swept by the tidal wave of sewer contents.

The remaining firemen are now refusing to enter the sewers until these cycles are removed. The remaining budget was already fairly allocated to various consulting companies involved in the important process of selecting a company to do the job. Luckily, scribbling of the sewer design was recovered and can be analyzed. So, the city council must find the cheapest way for cycle elimination so that the already very high budget deficit would not get much higher.

To block a specific tube in the system (i.e., removing some directed cycle in which the tube is present), one needs to dig a hole above the tube to enter it with the necessary equipment. However, this is not equally expensive for every tube in the system. If the tube lies under a busy street or a building, it can be much more costly to dig there than digging above a tube placed under a park (if Greenpeace members conduct no obstructions). On top of that, the tube might

[1] https://en.wikipedia.org/wiki/Penrose_stairs

be a part of more than just one cycle, so finding an optimal way to minimize all digging costs is a very complex problem.

As city council members refuse to cut their salaries to reallocate the budget, the council asked the nearby university department to organize some contest, which would quietly, under the radar, solve the entangled sewer problem. Their plan is to abuse poor students to develop a good solution to the problem. In return, students will be fairly rewarded with a valuable experience and the sense of accomplishment obtained by solving such a complex problem, which is all the reward a student actually needs. In addition, the name of the author of the best solution will be forever engraved on the slides of Combinatorial Optimization and in the sewers, thus granting the author further bragging rights and earn the respect of their peers.

2 Formal problem statement

You are given a directed graph $G = (V, E)$ with the set of vertices $V = \{1, 2, \ldots, n\}$ and directed edges E. Each edge $e \in E$, $e = (i, j)$, is associated with a cost $c_e \in \mathbb{Z}_+$. The goal is to find a set of edges D with the minimal sum of weights of edges while the set $E \setminus D$ contains no cycles. The resulting graph does not have to be connected, but think about what it would mean for the optimality of the solution (and poor people of the city using the sewer system to flush their waste).

Hence, the formal problem statement can be described as follows:

$$\min_{D \subseteq E} \sum_{e \in D} c_e$$

subject to:

$$(V, E \setminus D) \text{ is acyclic graph.}$$

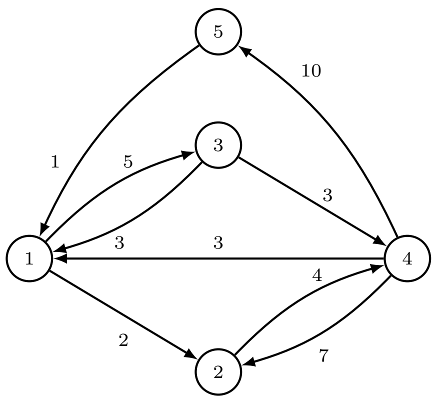


Figure 1: Example of the original sewer system as a directed graph.

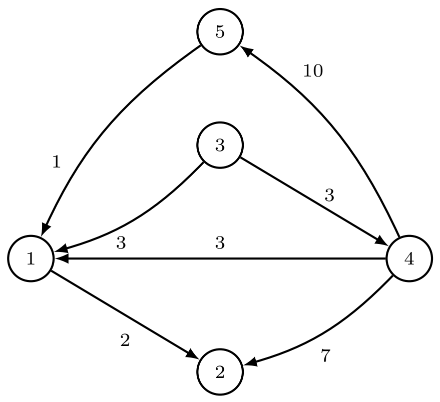


Figure 2: Example of the sewer system with blocked cycles as a directed graph.

3 Rules

If you decide to choose the contest as your semestral project, then you are expected to implement a correct solver for Smart city with not-so-smart sewers. The implementation will be submitted to BRUTE `https://cw.felk.cvut.cz/brute/` where it will be automatically evaluated (the number of submissions is not limited). The grading is a combination of the ability to find good solutions and the achieved rank relative to other students (w.r.t. the objective function). Therefore, you can acquire a small number of points even if your solver is not very efficient relative to other students.

In BRUTE, you will find 3 tasks related to the contest. Each task has specific instances, rules, and grading. The contest is split into different tasks to avoid re-evaluation of the instances (which is time-consuming) and so that you can implement a specific solver for each task.

1. `SP_CC_O`: you have to implement an exact MILP solver for the problem. If your solver solves all the instances in this task optimally, then you will get 3 points for this task. If the solver returns a suboptimal solution for **ANY** instance in this task, then the evaluation of your solver is stopped, and you will get 0 points in this task.
2. `SP_CC_T`: the goal is to find the best possible feasible solution within the specified time limit, i.e., the optimal solutions are not required, and you are encouraged to implement clever heuristics solving these instances. For each instance in this task, you will obtain some fraction of the point if your solution's cost is not worse than our threshold (8 points at max).
3. `SP_CC_R`: similarly as in `SP_CC_T`, in this task, we are also interested in finding the best possible feasible solution within the specified time limit. However, your solver's evaluation will depend on how good your solver is relative to other students' solvers, i.e., the number of points obtained will depend on your rank (5 points at max).

Some general contest rules also apply:

1. Usage of the single-purpose problem-specific solvers is prohibited (i.e., a MILP solver is allowed, but somebody's else code for solving "Smart city with not so smart sewers" like the problem is not).
2. Every participant is required to write their own code. However, sharing ideas and discussing the problem is encouraged.

4 Input and Output Format

In `SP_CC_O`, your solver will be called as

```
$ ./your-solver PATH_INPUT_FILE PATH_OUTPUT_FILE
```

whereas in `SP_CC_T` and `SP_CC_R` we include a time limit

```
$ ./your-solver PATH_INPUT_FILE PATH_OUTPUT_FILE TIME_LIMIT
```

- `PATH_INPUT_FILE` and `PATH_OUTPUT_FILE`: similarly as in homeworks, these parameters represent the path to the input and output files, respectively (see below for a description of the file formats).
- `TIME_LIMIT`: a float representing the time limit in seconds given to your solver. Your solver will be killed after the time limit is reached, and you will be awarded 0 points. Hence, your solver's output is considered only if your program exits with status code 0 before it times out.

The input file has the following form (we use one space as a separator between values on one line):

$$\begin{array}{lll} |E| & & \\ i_1 & j_1 & c_1 \\ \vdots & \vdots & \vdots \\ i_{|E|} & j_{|E|} & c_{|E|} \end{array}$$

The output file has the following format:

$$\begin{array}{ll} obj & \\ i_1 & j_1 \\ \vdots & \vdots \\ i_{|D|} & j_{|D|} \end{array}$$

where *obj* is an integer cost of all removed edges, and the edges listed are from the set D of deleted edges. You can assume that all numbers in the input and output file are integers.

Example 1

This example corresponds to the motivation example.

Input:

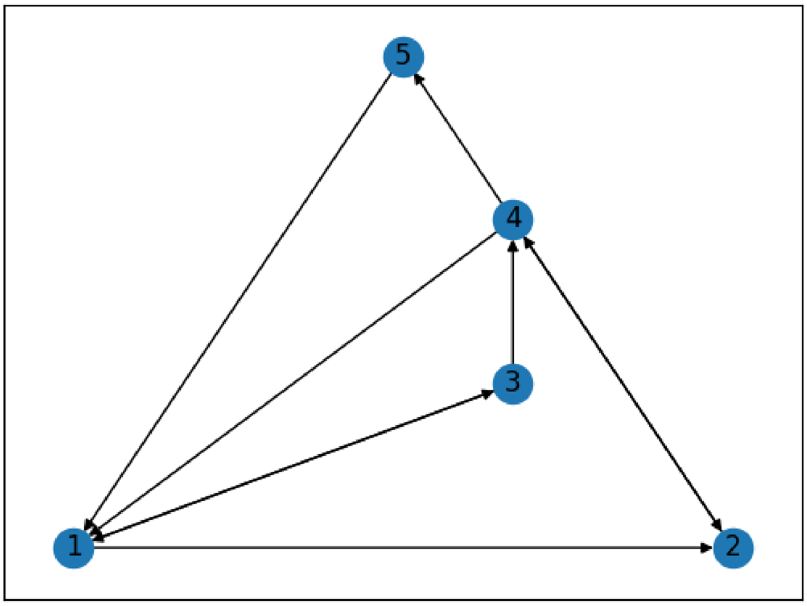
```
9
1 2 2
1 3 5
2 4 4
3 1 3
3 4 3
4 1 3
4 2 7
4 5 10
5 1 1
```

Output:

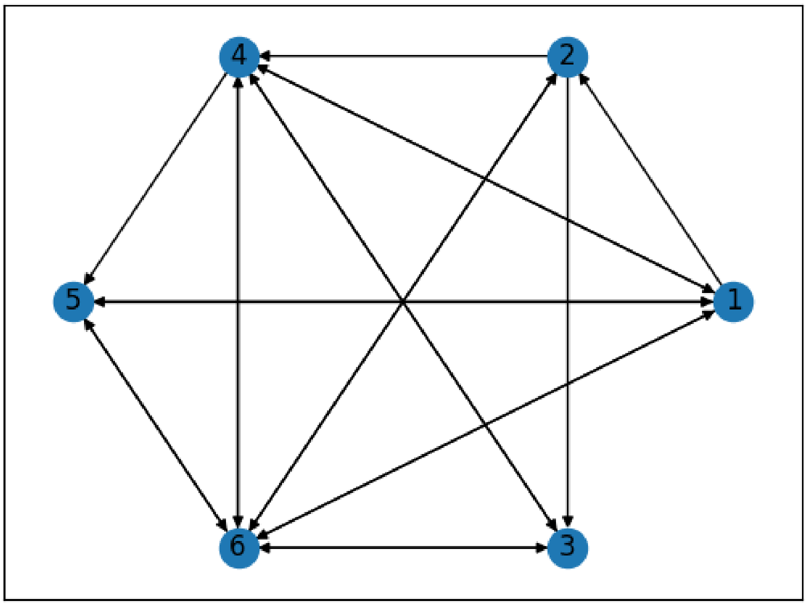
```
9
1 3
2 4
```

5 Images of smaller public instances

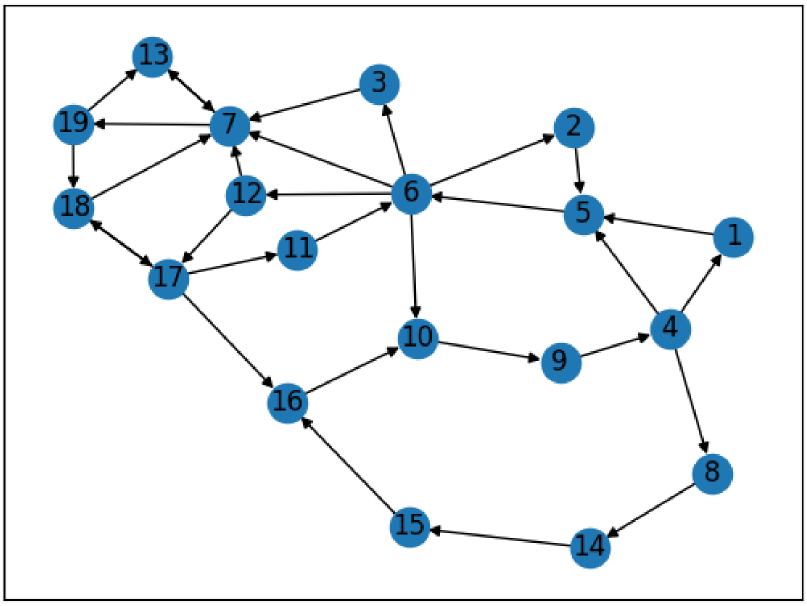
To further help you visualize the problem, you are provided with these graphs representing the first 4 public instances (without weights). If there is a two-way edge present between two vertices, the arrow is bidirectional. Because of clarity, weights are omitted from the images.



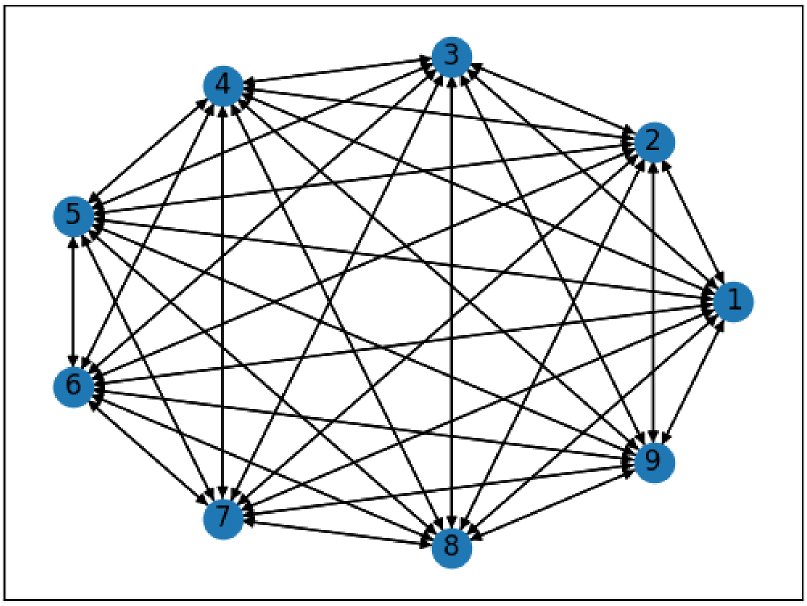
0.txt (also the same as the example instance mentioned above)



1.txt



2.txt



3.txt