Project 4: Light up the Skies!

CMPE 250, Data Structures and Algorithms, Fall 2020

Instructor: H. B. Yılmaz
TAs: Hazer Babür, Rıza Özçelik
Contacts: hazer.babur@boun.edu.tr
riza.ozcelik@boun.edu.tr

Due: **January 23, 2021, Saturday, 23.55**

“Introduce a little anarchy, upset the established order and everything becomes chaos. You’ll work hard to run bat-signal, adios... uhuhhhaahahahah..”.

1 Description

Hi, there. I’m Commissioner Gordon. First of all, thank you all for helping me to collect the Bat-Signal. But I have bad news. Just when we were assembling the last part of Bat-Signal, a piece of card is dropped from the machine. On the back of the card, the note above was written. When we tried to run Bat-Signal later, we understood what the note meant. Joker had sabotaged all cables of Bat-Signal and set up a setting. Now, we are in this situation: we have lots of cables, and ports just as much as the cables. Each cable produces a different voltage when connected to a different port. To run Bat-Signal we need the maximum voltage and each cable must be connected to one port. Finally, we have only one chance of trying...

So, we are looking for fast programmers who can calculate the maximum voltage in this setting, therefore we prepared some test cases.

I believe that this will be the last game of Joker. Let’s light up the sky!

2 Solution Guidelines

Let us share what we figured out until now in order to find a fast solution.

- This problem is solvable with a minimum-cost maximum-flow (MCMF) modeling approach and we expect you to model the test cases accordingly. We are aware that MCMF might be unfamiliar to you, but an excellent programmer is the one who finds a way through in the darkness!
- For MCMF modeling, you will need **cost and capacities** on each edge. Determine them wisely to find the maximum voltage.
- When the modeling is complete, you will find the algorithm for MCMF problems. The algorithm needs to find *negative cycles* and you are free to use the code and explanation in this link.

3 Input & Output

Your code must read the name of the input and output files from the command line. We will run your code as follows:

- `g++ *.cpp *.h -std=c++11 -o project4`
- `./project4 inputFile outputFile`

If you do not use any ".h" file. Your code will be compiled as follows: `g++ *.cpp -std=c++11 -o project4`

Make sure that your final submission compiles and runs with these commands

3.1 Input

Each input file contains several test cases and each test case consists of a matrix that specifies the output voltage when a cable is plugged into a port. The format of the input file is as follows:

- The first line contains an integer T, that denotes the number of test cases in this input file.
- The next line contains another integer N which is the number of cables (and ports) in the next test case.
- The subsequent N lines denote an $N \times N$ integer matrix V, in which V_{ij} is the output voltage when the cable i is plugged into port j.
- The rest of the lines denote other test cases, where each has the same format: an integer N and an $N \times N$ integer matrix V.

Input	Output
2	11
2	17
6 8	
3 4	
3	
2 4 6	
7 6 4	
5 3 1	

Table 1: Sample input and output files

- In all input files, $1 \leq T \leq 30, 1 \leq N \leq 150, 1 \leq V_{ij} \leq 2000$.

An example input file is displayed in Table 1 and Figure 1 illustrates the two test cases in the sample input with graphs.

3.2 Output

The output file is expected to contain the maximum voltage to run Bat-Signal for each test case in the input file in a separate line. The correct output file for the sample input is provided in Table 1.

In the first test case, we have two cables and two ports. To obtain the maximum voltage, we match the first cable with the second port (8V), the second cable with the first port (3V), totaling 11V. Note that we matched each cable with a single port and all of the cables and ports are matched.

In the second test case, we connect cable 1 to port 3 (6V), cable 2 to port 2 (6V), and cable 3 to port 1 (5V) and obtain the maximum voltage which is equal to 17 V. Remark that an alternative matching with the same output voltage is also possible but we are interested only in the maximum voltage.

Grading

We will automatically grade your submission with multiple files besides the ones we already provided. Therefore, **ensure that your submission is runnable with the commands we provided and satisfies all project requirements.** The auto-runnability of the submission will constitute **10/100** of your final grade and passing the test cases will yield **90/100** points. If your submission is not auto-runnable, then **you will receive 0 at first and then have to issue an objection**.

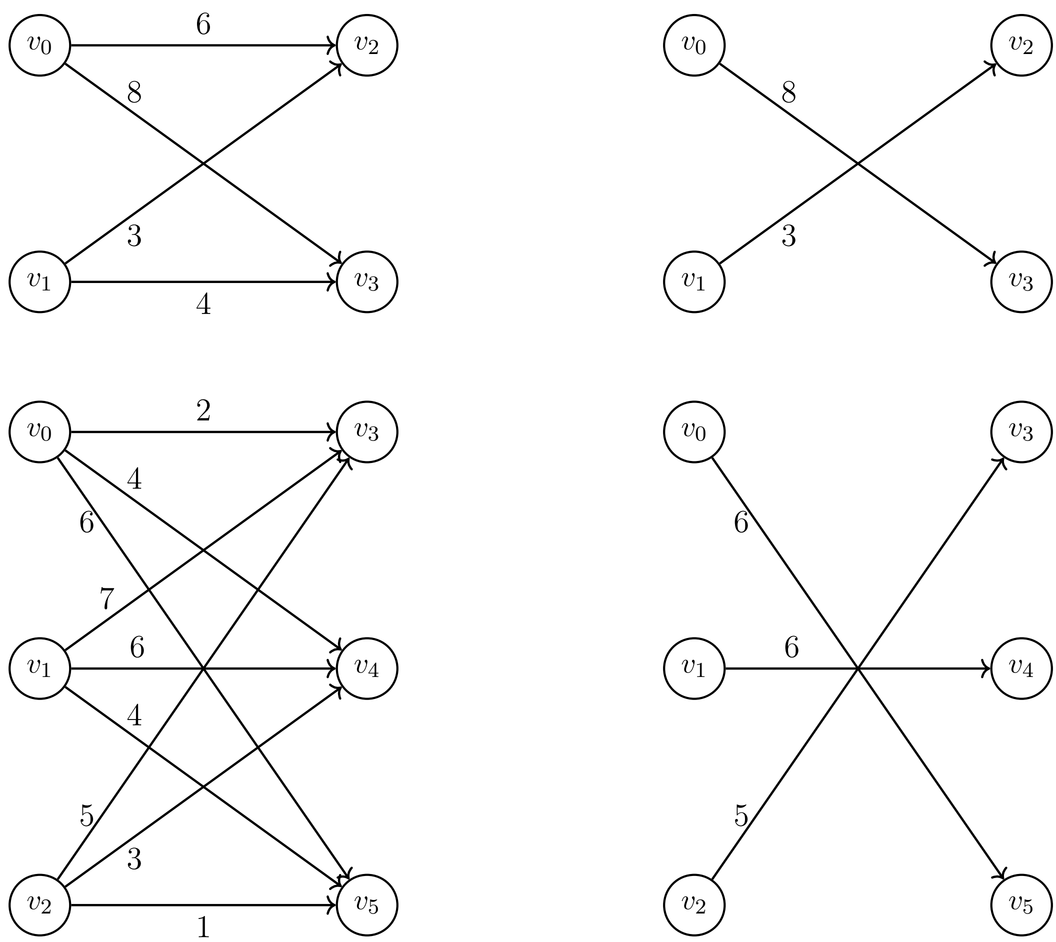

Figure 1: A graph representation (left) and a perfect matching (right) with maximum voltage for the test cases in the sample input file. You should use a different graph representation to model the problem with MCMF.

Submission Details

You are supposed to use GitHub Classroom for the submission same as the previous projects. No other type of submission will be accepted. Also, pay attention to the following points:

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. **Make sure you write and submit your own code. Any sign of cheating will be penalized by at least -100 points.**
- You can add as many files as you can as long as they are in the same folder with main.cpp.
- You are allowed to use **only** the standard C++ library. All requirements of the project must be satisfied with your authentic code, not by any external code. **Otherwise, you will get 0**.
- You are expected to use C++ as powerful, steady, and flexible as possible. Use mechanisms that affect these issues positively.

- Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long. This is very important for partial grading.
- Try to write as **efficient** (both in terms of space and time) as possible since we will **limit the run times** of the submissions to detect the correctness of your implementation.