EFREI 2020/2021
SM610I Graph Theory
L3, L3NEW

Project

Search for shortest paths using Floyd-Warshall algorithm

IMPORTANT: Read the entire document carefully before you start working. This will help you avoid unpleasant surprises.

Floyd-Warshall algorithm allows one to compute minimum value paths from any vertex to any other vertex in an oriented and valued graph.
This algorithm is based on the Warshall algorithm of computation of the transitive closure of a graph. The computation of transitive closure is adapted in order to keep, among all the paths connecting 2 vertices, one of those having the smallest value.

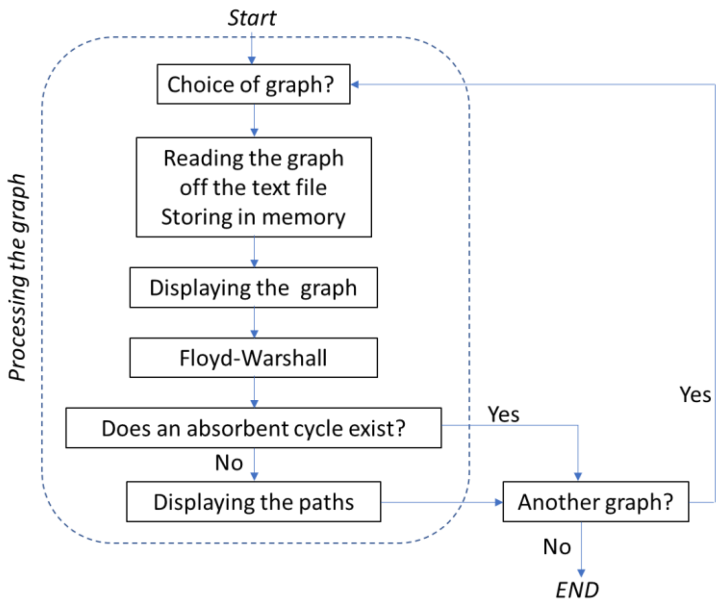
The Floyd-Warshall algorithm is not discussed either in exercises or in lectures. This project therefore requires some research work on your part.

GENERAL CONDITIONS

The project is to be carried out in teams. Your TD teacher will tell you into how many teams your TD group has to split.
You will need to use one of the following programming languages: C, C++, Java, or Python.
A project defense is scheduled for mid-November. You have to send the result of your work to your teacher beforehand. Test graphs, on which your program will be tested, will be sent to you some time before the presentation. You do not need those test graphs when developing the program: the program should work on any graph.

WORK TO DO

Illustration :



Main loop:

Your program must be able to execute the requested operations on a series of graphs, at the user's choice. During the presentation, it should not be necessary to restart your program to test the next graph.

Processing of a graph:

First, the user tells you which graph should be analyzed. The graphs will be identified by their sequential numbers.

The graph is then loaded into memory according to its description contained in a text file. See Appendix. The "read" graph is stored in data structures. From this point on, your program must never re-access the text file: only the memory representation is used.

> The choice of representing the graph in memory as a string of characters mimicking the lines of the text file will not be efficient if your program has to process large graphs. This is true whatever you dress it in, including tables, vectors, sets, ... The triplets suggested for a text representation of a graph are not a good choice for memory storage.
> **Your choice of a more efficient data structure will play a role in your grade.**

The graph is then displayed on the screen as a matrix: an adjacency matrix plus a matrix of values, or both combined.
Pay close attention to the legibility of this display, especially the alignment of the columns and the identification of the vertices (row and column titles).

The Floyd-Warshall algorithm is then executed. The intermediate values of the algorithm data (matrices usually named 'L' and 'P') should be displayed.

When you come to the results of the algorithm, your program should first of all say if the graph contains at least one absorbent circuit.

If no absorbent circuit is present, the last operation consists of displaying the shortest paths. You can display them all, or provide a user interface loop:

```
Path ?
If Yes then
    Initial vertex?
    Final vertex?
    Displaying the path
    Redo
If No then stop
```

GRAPHS TO CONSIDER

Your program must be able to correctly process any graph that meets the criteria below:

- The graph is oriented
- The graph is valued by arbitrary integer numbers (negative integers OK, zero value OK)
- The vertices are labeled by integers from 0 to the number of vertices minus one
- There is at most one edge going from vertex x to vertex y

Your program should not impose any limits on the number of vertices, values of edges, or number of edges.

THE PROJECT WORKFLOW

Forming the teams

Number of teams per TD group : 12

The number of students per team will be computed according to the number of students in each TD group. I will put a corresponding anouncement on Moodle in the next few days.

The team lists: I have to receive them on or before the date I'll specify in that same announcement.

No change will be possible after that date. If I do not receive the team lists from the delegates of each TD group, I'll split you into teams myself without any discussion. So please discuss the teams quickly enough and send them to me.

Possible problems : If there is a problem with a team (a student who is not really participating in the work with the others, a student who is sick, etc.), you are required to inform me immediately.

No changes to a team without my direct approval will be accepted.

Programming languages

You can choose between C, C++, Python, Java
However, the language chosen must be sufficiently mastered by all members of the same team so that all can participate. During the presentation, I can ask any question to any member of the team, whether or not they were involved in the development of that program part.
Your program must be able to be compilable/executable by me on my PC. I will tell you what software I have. You will have to adapt to it.

Defense

Presentation + demonstration of your program (precise conditions will be given later) + questions/answers.
It has not yet been decided whether the presentations will take place at EFREI or by videoconference.

A defense has a limited duration. I will have a very tight schedule. It means I won't be able to keep you after the scheduled time.
You are therefore strongly advised to be absolutely ready at the time of the beginning of your defense, which implies (in case it takes place at EFREI):

- - waiting at the door of the class and entering as soon as it is your turn;
- - having prepared your laptop, having put on it all your programs (and files containing the graphs);
- - having checked that the battery is charged;
- - your laptop has been booted and stays in the sleep mode;
- - having a spare computer on which you have also checked the correct functioning of your program, just in case...

Every year there are a lot of students who think they will never have a problem. Every year, there are some who do. As a result, they are stressed and have less time for their defense. What a pity...

Test graphs

Test graphs in graphic form will be provided to you before the presentation, early enough to allow you to code them into .txt files in the format of your choice. These files must be prepared in advance, and they will be a part of the submission package. They must be available on your computer at the time of the presentation.

To test your program, don't wait for these graphs! It would then be too late.
During your work, don't hesitate ^{2}to use all the graphs seen in lectures and exercises, and many more. The more testing you do, the more you can be sure that your program works properly.

Submission

All teams must submit their work to me by the same date. Instructions will be provided at a later date.

The contents of the submission package :

- Source code: Any code file that you typed yourself and no other file (**which means no file produced by the software during compilation or execution**), well commented.
  Every year, there are teams who do not respect this requirement and who get penalty points because of that negligence. Try to avoid it.
- All the .txt files of the test graphs (yes, despite the fact that your teachers provided you with the test graphs), in the same directory as the code.
- A ppt or pdf of the presentation (optional: you can attach it to your submission package if you already have it, but you will be allowed to modify it until your defense).
- Execution traces in the form of a .txt file. You will need to run your program on all of the test graphs mentioned above and provide the corresponding execution traces. The traces file should contain what the user can see on the screen during execution.
  A graph that has not been tested, or for which execution traces are not provided, will be considered as a graph on which your program does not run correctly.

Any file you use (and pass on to me) must be prefixed by your team number: for example, if you have a "main" file and if you use C++, and you are in team Int1-2, that file must have the name Int1-2-main.cpp (or Int1_2_main.cpp).

Appendix – Example of a file representing a graph

Careful : this appendix contains only an ***example*** file structure. You have the right to decide on your own structure if you respect the following conditions:
- the file structure must be simple, easily understandable ;
- the graph represented in the file must directly modifiable in the file, in an extremely simple way (for example if you wish to add or delete an edge, or to change the value of an edge).

Your file can, *for example,* have the following structure:

Line 1	Number of vertices
Line 2	Number of edges
Lines 3 to (3 + the number of edges)	Initial vertex followed by final vertex followed by the edge value

With that model, if the file contains the following text:

```
4
5
3 1 25
1 0 12
2 0 -5
0 1 0
2 1 7
```

The graph will have 4 vertices numbered from 0 to 3 (the numbering must be contiguous) and 5 edges (3,1), (1,0), (2,0), (0,1), (2,1) whose values are, respectively, 25, 12, -5, 0 and 7.