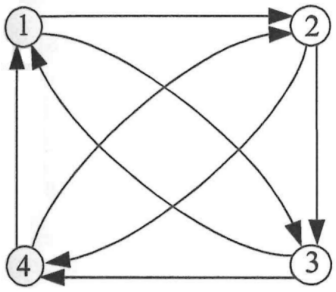
Project

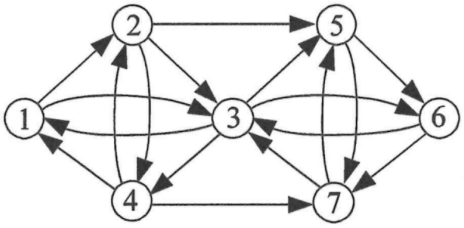
The project of the OR course consists of writing a Julia program using the JuMP library to solve an optimization problem. The problem we address is an application of the max-flow problem. It consists of finding a feature of a network, called the strong edge connectivity (SEC), which measures the fault resistance of a network. In this document, we first present the graph problem and then describe the program you will have to write. The project can be carried out in pairs (of 2 students).

1 The problem and its solution

The project consists of determining a characteristic of a directed graph called the strong edge connectivity (SEC). A directed graph is said to be strongly connected if there is a directed path between every ordered pair of vertices. The strong edge connectivity (SEC) of a directed graph is simply the least number of edges that must be removed in order to deny it the property of being strongly connected. Equivalently, if the strong edge connectivity of a directed graph is K, the are at least K edge disjoint paths between every ordered pair of vertices.



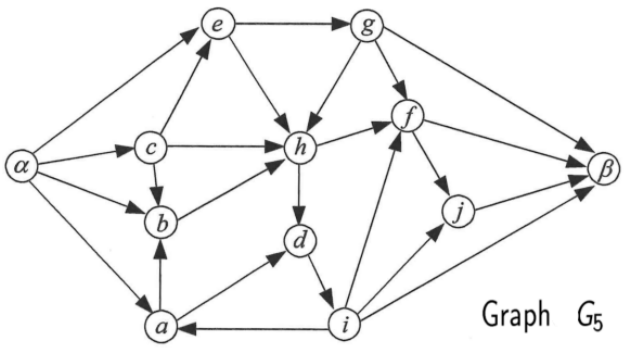
Graph $\mathcal{G}_3$



Graph $\mathcal{G}_4$

For example, the SEC of G_3 is 2 : if we remove one edge, the graph remains strongly connected but if we remove the edges $(3,1)$ and $(3,4)$, all the paths outgoing vertex 3 are removed : G_3 is no longer strongly connected. The SEC of G_4 is 2 as well : the directed cycle $(1,2,5,6,7,3,4,1)$ shows that G_4 is strongly connected and keeps this property if we remove one edge not belonging this cycle ; furthermore, if we remove one edge of the cycle, G_4 remains strongly connected. Finally, if we remove $(1,2)$ and $(1,3)$ no path outgoes vertex 1.

We know want to determine the SEC of a directed graph of n vertices. We first introduce the following notation : given any two distinct vertices s and t, $P(s,t)$ represents the maximum number of edge disjoints path going from s to t. (Two edge disjoint paths have no edge in common, but they may have some vertices in common). For example, in G_5, $P(\alpha,\beta)=3$:



Graph G_5

*Le nombre maximum de chemins arc-disjoints entre α et β correspond au flot maximum entre α et β car on assigne une capacité de 1 à chaque arête du graphe.

Given a directed graph G and two vertices of G, α and β, it is possible to compute $P(\alpha, \beta)$ by considering the following network flow problem : the source is α, the target is β and a capacity of 1 is assigned to each ~~vertex~~ edge. If $valmax$ designates the maximum value of a flow in the network, then $valmax = P(\alpha, \beta)$ and $P(\alpha, \beta)$ edge disjoint paths are made of edges with a flow of 1 in an integer maximum flow.

How can we compute the SEC of a graph ? It is the minimum value of the $P(s, t)$ for any ordered pair of vertices (s, t). However, it is not necessary to consider all the $n(n-1)$ ordered pairs of vertices : we can consider only n ordered pairs of vertices. Let v_0, v_1, ..., v_{n-1} be the vertices of G and let $v_n = v_0$. The SEC of G is :

On boucle pour comparer le dernier au premier sommet.

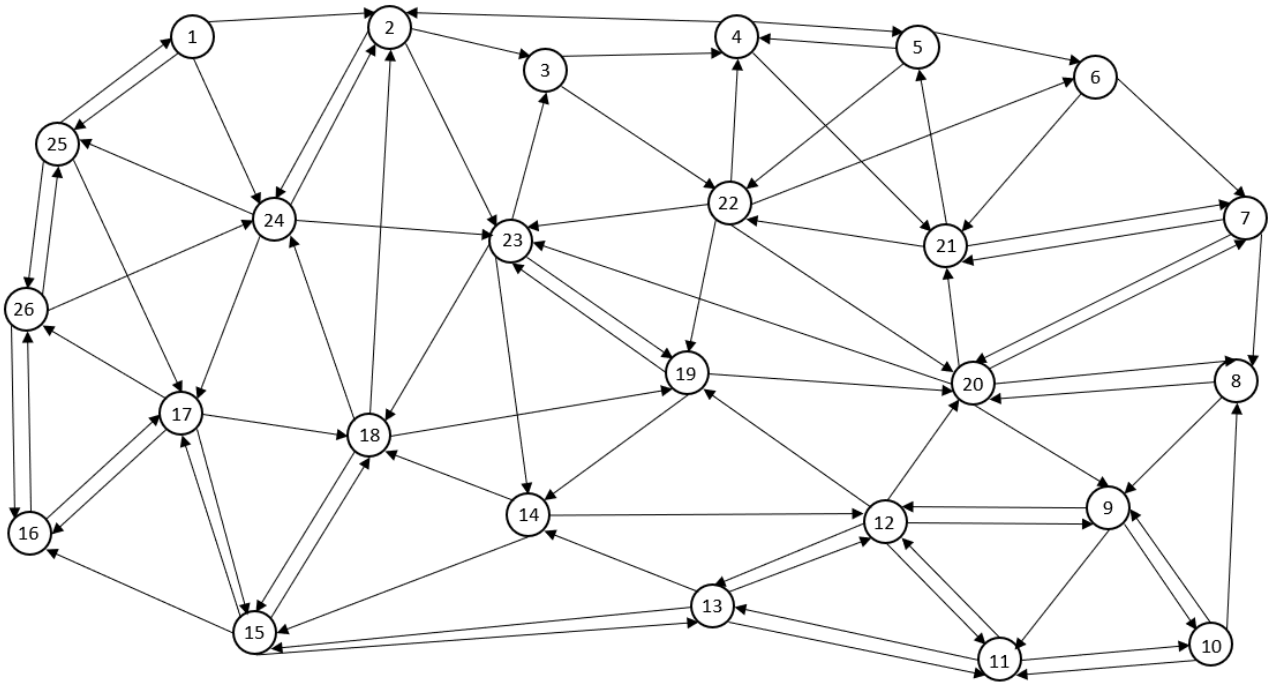
$$\min_{i \in \{0,\dots,n-1\}} P(v_i, v_{i+1})$$

Computing the SEC of G consists therefore of computing n max-flow problems.

2 Description of the project

You have to compute the SEC of the graphs G_3, G_4, G_5 (described above) and G_1 (drawn below). The graphs will be represented by their adjacency matrix. You will have to compute the SEC of the graph using Julia/JuMP and the free solver Cbc. The steps of the project are :

1. Write a Julia function named "P" that computes the value $P(\alpha, \beta)$ for a given adjacency matrix representing a graph G, and for two given vertices α and β of G. The value $P(\alpha, \beta)$ is computed by solving the corresponding max-flow problem. To solve the max-flow problem, you will write the corresponding Linear Programming model and solve it with the Cbc solver (it is not asked to implement Ford-Fulkerson's algorithm : solve the LP model instead).
2. Compute the SEC of G by computing $\min_{i \in \{0,\dots,n-1\}} P(v_i, v_{i+1})$.
3. Compute and indicate the minimum set of edges that has to be removed to make the graph no longer strongly connected. This set of edges corresponds to the edges associated with the minimal cut of the flow minimizing the $P(v_i, v_{i+1})$
4. Bonus : generate a connected graph of 1000 vertices and compute its SEC.



Graph G_1

3 Deadline

May 26th

The deadline of the project is ~~April 22nd~~, 2023, 23 :59. You have to send your Julia program to the email address eric.soutil@lecnam.net Indicate the name of all the members of your team at the first line of your program. Your program should be sufficiently commented and the solution of the 4 instances should be clearly indicated at the beginning of the program.

4 Instances

Here are the adjacency matrices (in Julia syntax) of the graph G1 drawn in the project brief, and the one of another graph G2, to test your program.

```
G1= [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0;
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0;
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1;
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0;
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0;
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0;
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0;
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0;
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0]

G2=[0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 1;
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0;
1 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 0 0;
0 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0;
0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 0;
0 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0;
1 0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1;
```

```
1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0 1;
0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1;
1 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1;
1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1;
0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0;
1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1;
1 1 0 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0;
0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1;
1 0 1 0 1 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 0 1 0 0;
1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1;
0 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 0 0;
0 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 1 0 1 1 1 0 1 0;
0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0;
1 0 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0;
1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 0 1 1;
0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1;
0 1 0 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 0;
1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1;
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0]
```