PALACIO Vincent
ROBIN Florian

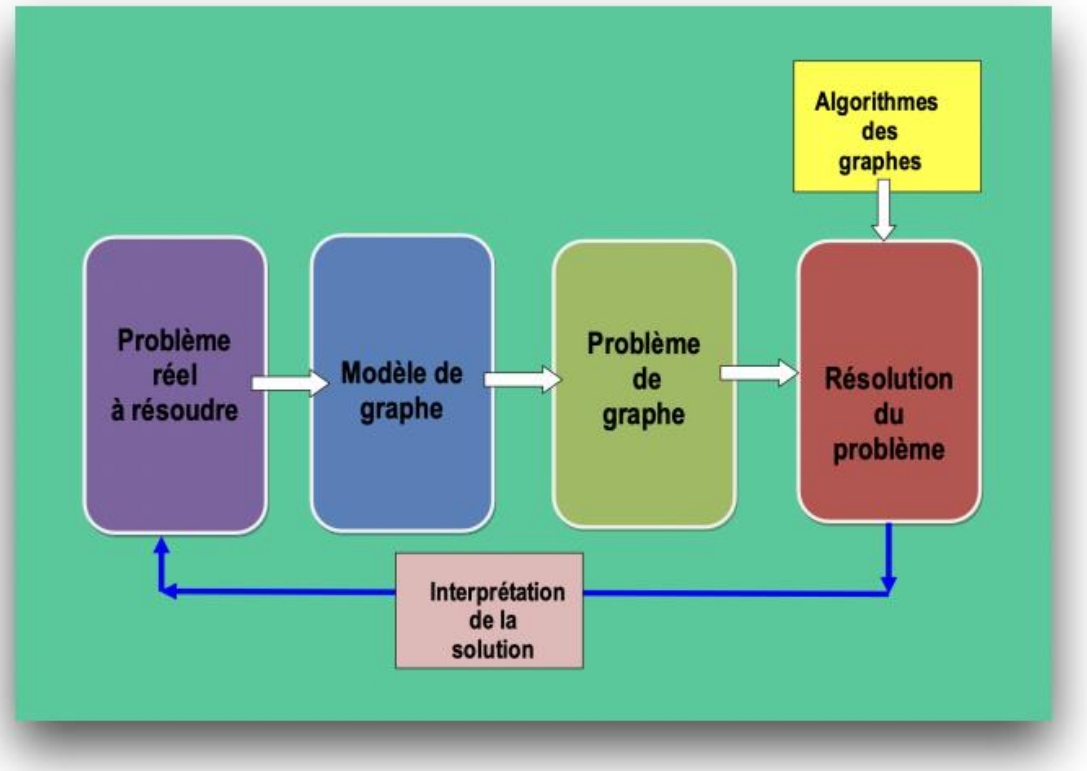
Compte Rendu du TP1

Sujet : Les test de connexité des réseaux « critiques »

Objectif du TP : Étude de la connexité d'un réseau

I -Position du problème :

Lors des phases de conception et surtout de test d'un réseau de communication dans les domaines critiques tels que la sécurité, la défense ou la finance, il est nécessaire de garantir certaines fonctionnalités. Notamment, il est primordial de garantir une communication depuis n'importe quel nœud du réseau vers n'importe quel autre. Il faut donc être capable de réaliser cette communication peu importe la configuration du réseau. Ce travail est réalisé par l'ingénieur chargé de la conception du réseau.



Dans une partie du réseau, garantir une communication entre deux nœuds revient à étudier la connexité de ce sous-réseau et donc à l'analyse de la connexité du modèle de graphe. Ainsi, on peut ramener ce problème à un problème de recherche de composantes fortement connexes dans un graphe.

Plusieurs outils ont été utilisés pour l'étude du graphe :

• Il existe un algorithme, dit Kosaraju-Sharir, qui recherche toutes les composantes fortement connexes d'un graphe. L'analyse du graphe reposera donc sur cet algorithme décrit ci-dessous.

- **Etape 1** : lancer un premier parcours en profondeur sur G pour obtenir une liste des sommets de G triée dans l'ordre décroissant par rapport à la date fin.

- **Etape 2** : lancer un second parcours en profondeur sur le graphe dual Gt (S,At) avec, dans chaque boucle de parcours un marquage des sommets dans l'ordre spécifié par liste triée.

• Nous avons décidé de développer les différents programmes aidant à l'analyse du graphe en C++, à l'aide de l'environnement de développement intégré Visual Studio Code et aussi en utilisant le code fourni.

• Nous avons également utilisé la librairie BoostGraph afin de visualiser les graphes qui seront créés.

De plus il existe dans un réseau plusieurs types de nœuds domaines en fonction des arcs qui sont attribués aux nœuds.

- Le domaine source : il ne peut que recevoir des informations.
- Le domaine intermédiaire : il peut aussi bien envoyer ou recevoir des informations.
- Le domaine puits : il ne peut qu'envoyer des informations.

Ces nœuds domaines seront différencier plus facilement grâce à la librairie de boostGraph.

Nous pouvons donc diviser la résolution de ce problème 3 étapes, tout d'abord nous allons présenter le modèle du graphe du système étudié en lien avec la communication du réseau. Ensuite il suffit d'appliquer cette modélisation à notre cas d'étude. Pour finir nous montrerons que le problème exposé ci-contre se ramène à un problème de composante fortement connexe

II- Réalisation :

1) Présenter le modèle du graphe en lien avec le système étudié :

Voici la configuration du modèle de graphe pour le cas étudié :
La décision a été prise d'utiliser un graphe orienté G pour modéliser le réseau.

De plus G = (S, A) où :

• S est l'ensemble des sommets s ∈ S représentant chacun un nœud du réseau

• A est l'ensemble des arcs (s i, s j) ∈ A représentant chacun une connexion entre le nœud représenté par son extrémité initiale s i et le nœud représenté par son extrémité terminale s j.

il faut également déterminer si on utilise un graphe orienté ou non orienté, d'après le cas d'étude exposé à l'aide de la matrice ci-dessous, nous pouvons dire qu'il faut utiliser un modèle de graphe orienté.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	0	1	0	0	0	0	0	1	1	0	0	0	1
2	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0
3	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
5	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
15	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

G=(S,A,) où S: ensemble fini de sommets, A: un ensemble fini d'arêtes. Ils sont définis tel que :

- Chaque sommet Si (i=1…,15) représente un domaine du sous réseau.
- Chaque arête (Si, Sj) correspond à une connexion entre deux domaines du sous réseau.

2) Montrer de quelle manière le problème exposé se rapporte à un problème relevant de la théorie des graphes :

Étudier la connexité de certaines parties du réseau (communication entre deux nœuds) revient à chercher les différentes composantes fortement connexes du graphe représentant ce réseau.

Tout d'abord définissons la connexité d'un graphe :

Un graphe G=(S,A) orienté (non orienté) est dit connexe SI x, y$\in$ S $\Rightarrow \exists$ une chaîne reliant x et y

Cependant dans notre cas nous avons besoin de définir les composantes fortement connexes, c'est-à-dire un sous-graphe de G fortement connexe maximal.

Pour qu'un sous graphe soit maximal il faut également respecter la propriété ci-dessous :

Soit Gi=(Si,Ai) un sous-graphe connexe de G=(S,A). On dit que Gi est connexe maximal SI $\forall$Gk=(Sk,Ak) sous-graphe connexe de G $\Rightarrow$ Gi n'est pas sous-graphe de Gk.

3) Exposer la solution retenue en mettant en œuvre une solution algorithmique issue de la théorie des graphes :

Nous nous sommes donc appuyés sur l'algorithme Kosaraju-Sharir, qui recherche toutes les composantes fortement connexes d'un graphe. Cet algorithme consiste à parcourir une première fois en profondeur le graphe, ce qui donne un ordre entre les sommets. (voir figure 1).

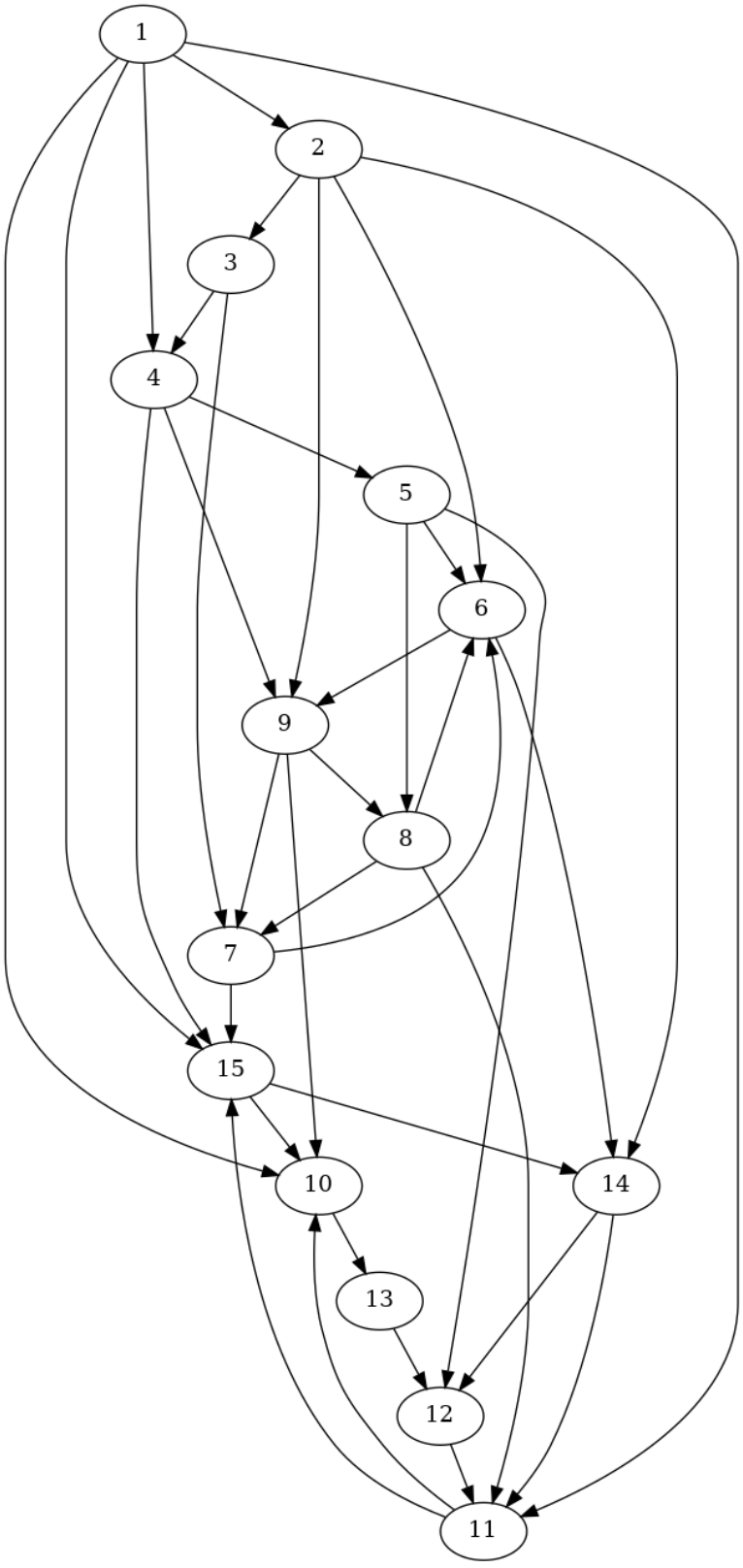


Figure 1 : graphe ordonné entre les sommets

Ensuite, il faut parcourir une deuxième fois en profondeur le graphe dual (voir figure 2), en explorant les sommets dans l'ordre inverse de l'ordre donné par le premier parcours. Ce deuxième parcours produit des arbres qui sont les composantes fortement connexes du graphe.

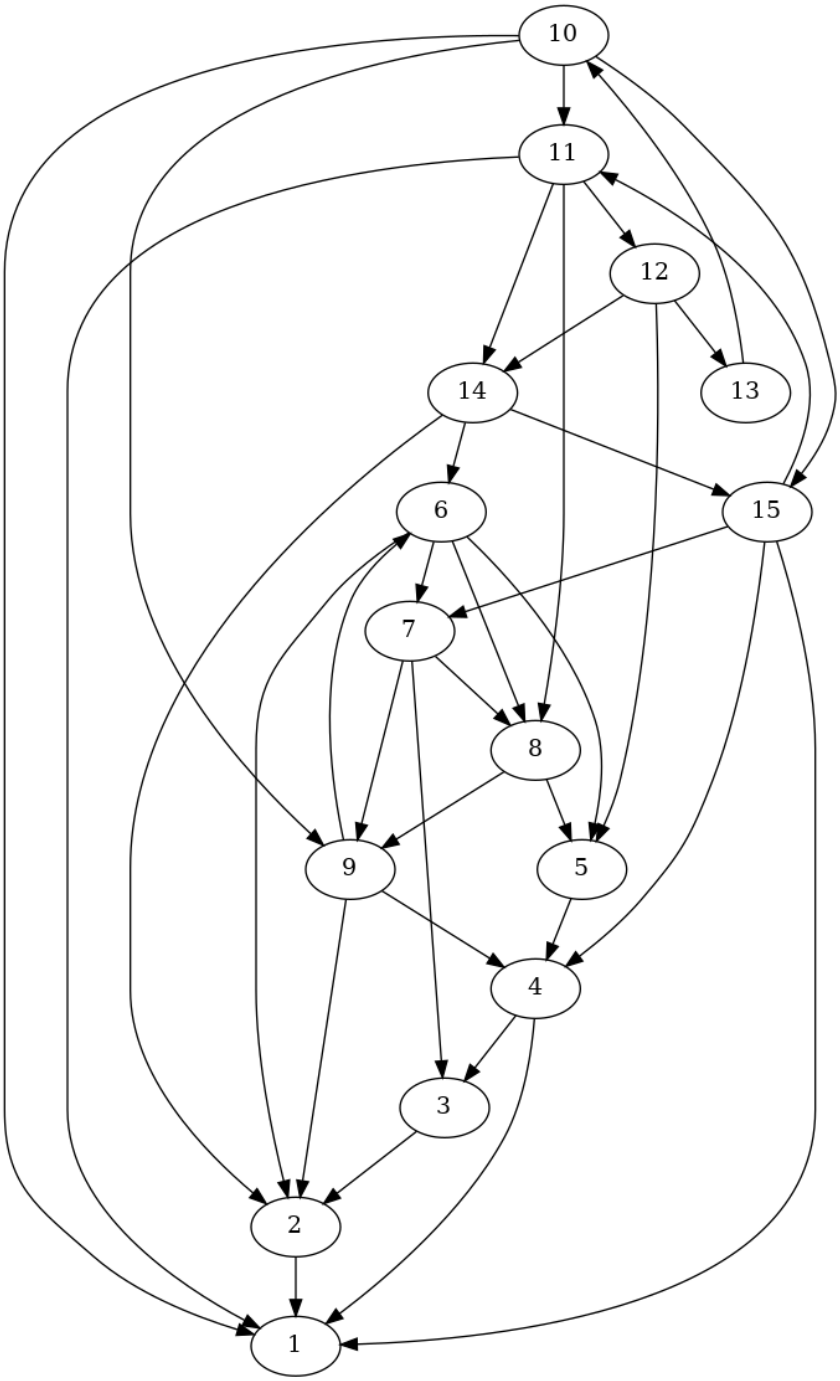


Figure 2 graphe dual

Lorsque le graphe est donné sous forme de liste d'adjacence, l'algorithme a une complexité linéaire qui dépend du nombre de sommets et d'arcs du graphe. Ainsi, la complexité est en O(|S|+|A|).
Nous avons retenu la solution suivante, la classe Graphe comporte deux attributs :

• un entier S correspondant au nombre de sommets du graphe

• une liste chaînée adjacente, où chaque cellule est une liste chaînée d'entier, correspondant à la liste d'adjacence du graphe

Pour finir nous avons mis ci-dessous le résultat final en montrant clairement les composantes à caractère fortement connexe. (voir figure 3).

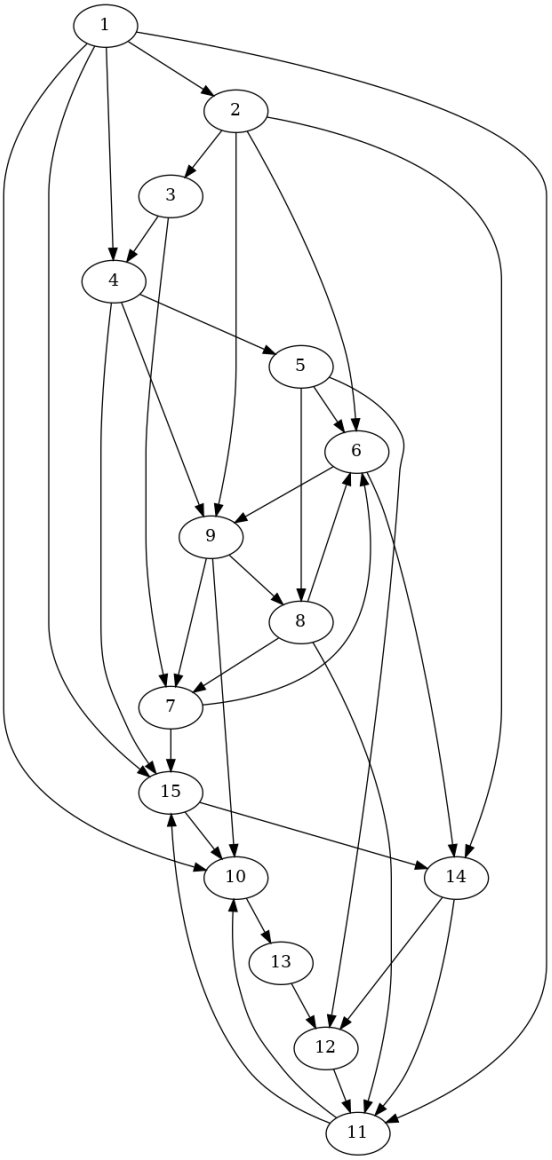


Figure 3 : graphe avec composante fortement connexe

III- Conclusion :

En réalisant cette série de travaux pratiques, nous avons développé nos compétences dans la programmation de modèles et d'algorithmes sur les graphes. Nous avons pu voir qu'il est primordial de savoir déterminer les composantes fortement connexes d'un graphe pour étudier la connexité d'un graphe.
Ainsi nous avons pris connaissance de l'algorithme de Kosaraju-Sharir qui permet de déterminer les différentes composantes fortement connexes d'un graphe et nous avons pu implémenter les parcours en profondeur de graphe notamment grâce aux librairies déjà existante dans le langage C++.
En déterminant les composantes fortement connexes, on peut en déduire la connexité d'un graphe, et lorsque le domaine d'application l'exige, on peut donner la réponse à la question : « est-ce que les nœuds d'un réseau peuvent communiquer avec n'importe quel nœud du même réseau, au moins pour une partie du réseau ». Ce qui permet de s'assurer que le réseau complet sera fonctionnel.