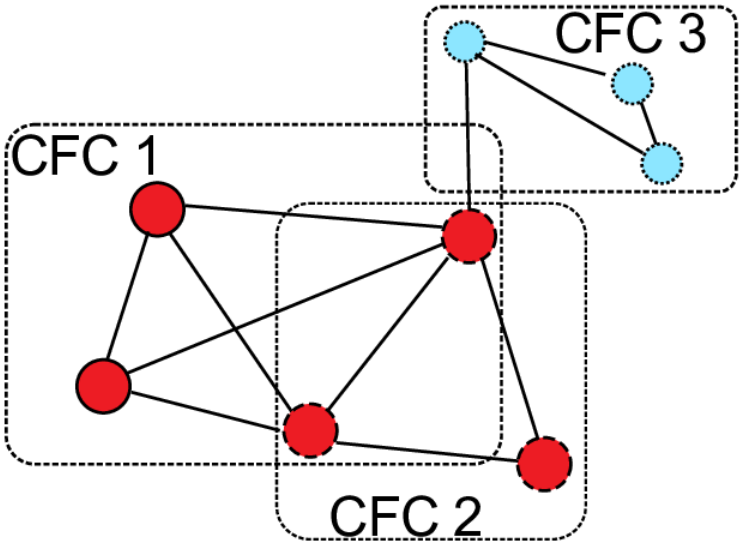
ALGORITHME D'OPTIMISATION SUR LES GRAPHES



Koccou MINAGNIKPO

Janvier 2024

Table des matières

1 Introduction

Face aux défis de la gestion du trafic urbain, une stratégie envisagée par les municipalités consiste à transformer les rues bi-directionnelles en sens uniques pour fluidifier la circulation. Ce problème, dans le domaine des graphes, revient à déterminer si un graphe non orienté peut être orienté en un graphe fortement connexe, permettant ainsi un accès complet et bidirectionnel à toutes les zones de la ville. Le théorème de Robbins (1939) établit une condition clé pour cela : un tel arrangement est possible si le graphe initial ne contient aucune arête déconnectante. Ce rapport se concentre sur l'implémentation des algorithmes de Schmidt pour identifier les composantes 2-connexes et 2-arêtes connexes d'un graphe, fournissant ainsi une méthode pour répondre à cette question cruciale en termes de planification urbaine.

2 Calcule des composantes 2-connexes d'un graphe

L'algorithme de Schmidt pour identifier les composantes biconnectées d'un graphe s'appuie sur le parcours en profondeur (DFS) pour explorer de manière systématique le graphe. Durant ce parcours, chaque sommet se voit attribuer un numéro d'ordre de visite (index DFS) et une valeur "low". La valeur "low" d'un sommet reflète le sommet le plus ancien (en termes d'ordre de visite) accessible par des arêtes de retour ou des arêtes descendantes dans l'arbre DFS.

L'aspect clé de cet algorithme réside dans l'identification des sommets d'articulation et des ponts, qui sont critiques pour déterminer les composantes biconnectées. Un sommet d'articulation est un point où, s'il est retiré, le graphe se scinde en composantes plus petites. De même, un pont est une arête dont la suppression augmente le nombre de composantes connexes du graphe.

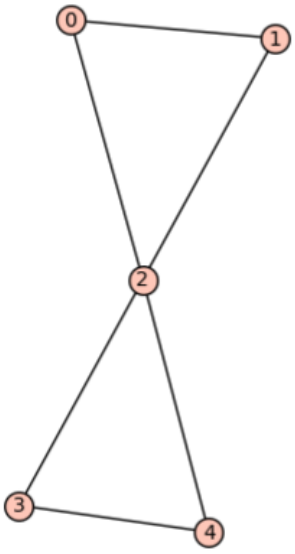
Pour garder la trace des arêtes explorées, l'algorithme utilise une pile. Lorsqu'un sommet d'articulation est découvert, toutes les arêtes depuis la dernière occurrence d'un tel sommet sont extraites de la pile pour former une nouvelle composante biconnectée. À la fin du parcours DFS, toutes les arêtes restantes dans la pile forment également une composante biconnectée.

Enfin, ces composantes biconnectées sont converties en sous-graphes pour faciliter leur analyse et interprétation. Cette approche permet de décomposer efficacement un graphe en ses composantes les plus robustes en termes de connectivité, offrant des insights importants pour des applications allant de la théorie des réseaux à la planification urbaine.

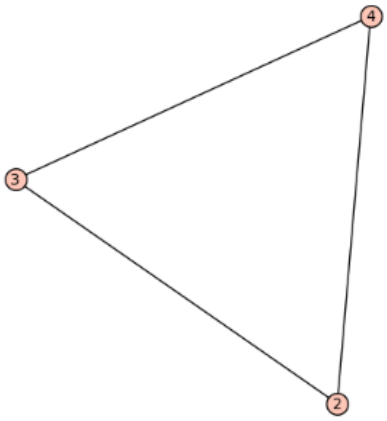
2.1 Exemples

Considérons un exemple concret pour illustrer le calcul des composantes 2-connexes d'un graphe à l'aide de l'algorithme de Schmidt

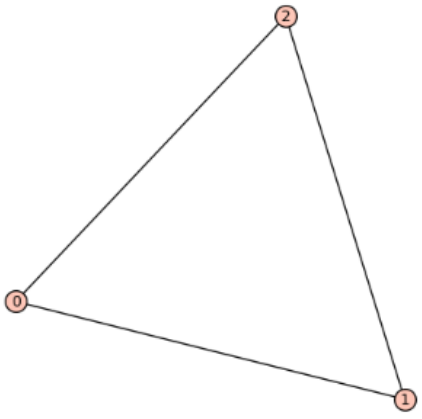
Graphe original:



Compososante biconnectée 1:



Composante biconnectée 2:



3 Calcule des composantes 2-arêtes connexes d'un graphe

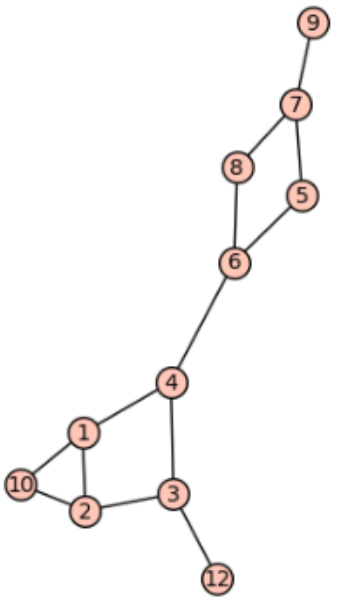
Cette méthode, basée sur l'approche développée par Jens M. Schmidt, utilise un parcours en profondeur (DFS) pour explorer le graphe et identifier les ponts, c'est-à-dire des arêtes dont la suppression augmenterait le nombre de composantes connexes du graphe. L'algorithme commence par attribuer à chaque sommet un indice de découverte et une valeur "low" lors du parcours DFS. Les arêtes qui relient un sommet à un sommet non parent et ayant une valeur "low" supérieure à l'indice de découverte du sommet sont identifiées comme ponts. Ces ponts sont essentiels car leur suppression divise le graphe en composantes plus petites et indépendantes.

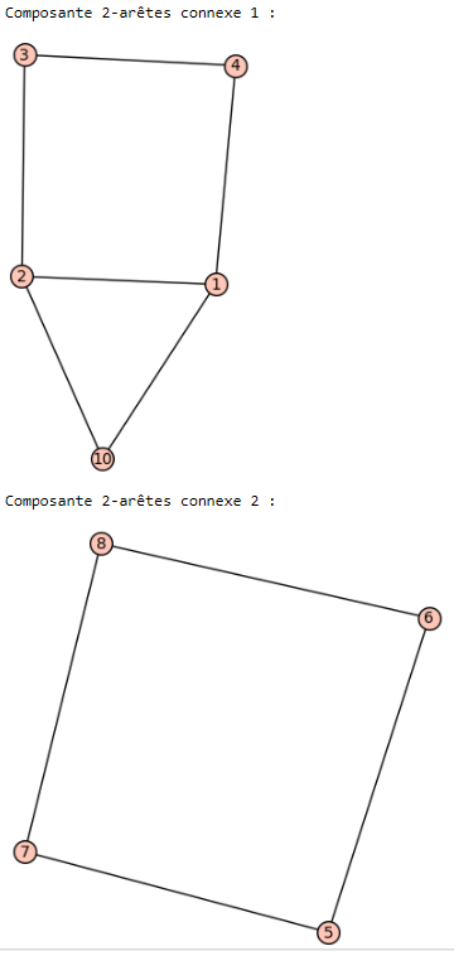
Après avoir identifié tous les ponts, l'algorithme les supprime du graphe. Chaque composante connexe restante, c'est-à-dire chaque sous-graphe qui ne peut être davantage divisé par la suppression d'une seule arête, est considérée comme une composante 2-arêtes connexe. Ces composantes sont ensuite extraites et visualisées séparément.

3.1 Exemples

Les composantes 2-arêtes connexes du graphe, obtenues après l'identification et la suppression des ponts grâce à un parcours en profondeur, sont maintenant calculées et prêtes à être visualisées.

Graphe original :





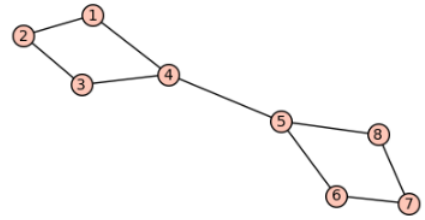
4 Exercice 2

Le code commence par définir une fonction interne dfs (pour Depth First Search, ou recherche en profondeur) qui est utilisée pour parcourir le graphe. Plusieurs structures de données sont initialisées pour stocker les informations nécessaires pendant le parcours : visited pour marquer les sommets visités, disc pour stocker l'ordre de découverte des sommets, low pour retenir le sommet accessible le plus bas dans l'arbre de recherche, parent pour conserver le sommet parent dans l'arbre de recherche, et bridges pour stocker les ponts trouvés. La fonction principale lance une recherche en profondeur (DFS) à partir de chaque sommet non visité, mettant à jour les valeurs de disc, low, et parent, essentielles pour identifier les ponts. Un pont est détecté lorsqu'un sommet v (voisin de u) a une valeur low plus grande que la valeur disc de u, signifiant qu'il n'existe pas de chemin alternatif (sans passer par l'arête (u, v)) qui relie v ou ses descendants

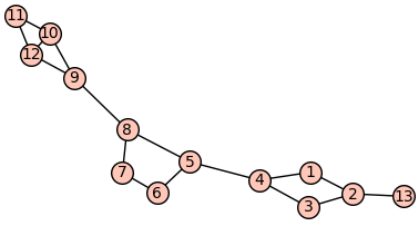
dans l'arbre DFS à un ancêtre de u. Si cette condition est vérifiée, l'arête (u, v) est un pont et est ajoutée à l'ensemble des ponts. À la fin, la fonction find bridges renvoie l'ensemble des ponts trouvés dans le graphe. Dans le code donné, un graphe G est créé et la fonction find bridges est appelée pour trouver et afficher les ponts de ce graphe.

4.1 Exemple

Le code ci-dessous identifie et affiche les ponts dans le graphe donné en utilisant l'algorithme de recherche en profondeur (DFS) pour détecter les arêtes déconnectante :




```
{(4, 5)}
```




```
{(8, 9), (2, 13), (4, 5)}
```

5 Exercice 3

Démonstration. La preuve se déroule en deux étapes :

Étape 1 : Si un graphe est fortement connexe, alors chaque arc est dans un circuit.

Considérons un graphe fortement connexe et prenons un arc quelconque (u, v) dans ce graphe. Par définition de la forte connectivité, il existe un chemin de v à u. En ajoutant l'arc (u, v) à ce chemin, nous formons un circuit qui inclut l'arc (u, v). Ceci démontre que chaque arc est présent dans au moins un circuit.

Étape 2 : Si chaque arc est dans un circuit, alors le graphe est fortement connexe.

Supposons maintenant que chaque arc du graphe soit inclus dans au moins un circuit. Pour démontrer que le graphe est fortement connexe, considérons deux sommets quelconques u et v. Si $u = v$, il existe un chemin trivial de u à v. Si $u \neq v$, prenons un arc (x, v) menant à v, sachant que v est inclus dans au moins un circuit. Il existe alors un chemin de v à x en suivant le circuit incluant (x, v). De même, il existe un chemin de u à x puisque chaque arc est dans un circuit. En combinant ces chemins, nous obtenons un chemin de u à v via x, ce qui prouve que pour chaque paire de sommets u et v, il existe un chemin de u à v, et donc le graphe est fortement connexe. □

6 Exercice 4

Démonstration. Considérons les deux parties de cette affirmation séparément.

Partie 1 : Si un graphe est fortement connexe, son graphe sous-jacent est-il 2-arête connexe ?

Un graphe dirigé est dit fortement connexe si pour chaque paire de sommets u et v, il existe un chemin dirigé de u à v et un chemin dirigé de v à u. Cependant, cette propriété ne garantit pas que la suppression d'une arête quelconque dans le graphe sous-jacent (non dirigé) laisse le graphe connexe. Par exemple, considérons un graphe dirigé en forme de cycle. Bien qu'il soit fortement connexe, son graphe sous-jacent n'est pas 2-arête connexe. Par conséquent, un graphe fortement connexe n'implique pas que son graphe sous-jacent soit 2-arête connexe.

Partie 2 : Si le graphe sous-jacent d'un graphe dirigé est 2-arête connexe, le graphe dirigé est-il fortement connexe ?

Un graphe non dirigé est 2-arête connexe si la suppression de n'importe quelle arête laisse le graphe connexe. Cependant, cela ne signifie pas que dans le graphe dirigé correspondant, pour chaque paire de sommets u et v, il existe un chemin dirigé de u à v et un chemin dirigé de v à u. Prenons l'exemple d'un graphe sous-jacent qui est un cycle simple, clairement 2-arête connexe. Si les arêtes de ce cycle sont toutes dirigées dans la même direction, alors le graphe dirigé n'est pas fortement connexe car il n'y a pas de chemin dans les deux sens entre chaque paire de sommets. Ainsi, un graphe sous-jacent 2-arête connexe ne garantit pas que le graphe dirigé soit fortement connexe.

En conclusion, l'affirmation est incorrecte. Un graphe fortement connexe ne garantit pas que son graphe sous-jacent soit 2-arête connexe, et la 2-arête connexité d'un graphe sous-jacent ne garantit pas la forte connectivité du graphe dirigé correspondant. Ces concepts traitent de propriétés différentes dans les graphes dirigés et non dirigés, et ne sont pas interchangeables. □

7 Conclusion

En conclusion, nous nous sommes concentrés sur plusieurs aspects clés de la théorie des graphes. En nous basant sur l'algorithme de Schmidt, nous avons d'abord implémenté des méthodes pour calculer les composantes 2-connexes et 2-arêtes connexes d'un graphe. Cette approche nous a permis de mieux comprendre la structure et la robustesse des graphes en identifiant des sous-graphes plus résistants aux perturbations. Ensuite, nous avons écrit un code pour identifier une arête déconnectante dans un graphe, ce qui est crucial pour évaluer la vulnérabilité d'un réseau.

En plus de ces aspects pratiques, nous avons également abordé deux importantes preuves théoriques. La première concerne la forte connectivité d'un graphe et sa relation avec la présence de chaque arc dans au moins un circuit. La seconde preuve examine l'affirmation selon laquelle un graphe est fortement connexe si et seulement si son graphe sous-jacent est 2-arête connexe, une proposition qui nous a amenés à explorer la différence entre la connectivité dans les graphes dirigés et non dirigés