Feuille d'exercices n°3 - Graphes et parcours

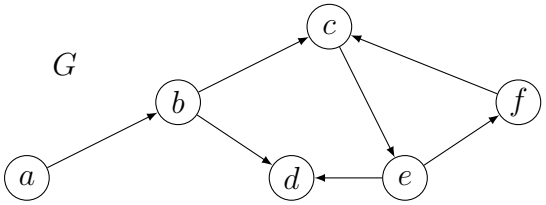
Notions abordées

- Tri topologique
- Révisions sur le parcours de graphe non orienté pour la détection de biparti
- Tri préfixe et graphe transposé
- Détection de circuit

Exercice 1 : Parcours selon le miroir d'un tri préfixe

Le but de cet exercice est de montrer la nécessité de passer au graphe transposé pour calculer une décomposition en CFC. Plus précisément, il s'agit de montrer qu'un parcours de G où les points de régénération sont choisis prioritairement selon le miroir de T un tri préfixe de G ne donne pas la même partition que le parcours de G^t selon T.

Q. 1 Donner T le parcours préfixe du graphe ci-contre produit par un parcours en profondeur dans lequel les successeurs d'un sommets sont étudiés par ordre alphabétique.



Q. 2 Donner un parcours de G^t selon T. Quelle est la partition associée ?

Q. 3 Donner un parcours de G selon le miroir de T. Quelle est la partition associée ?

Q. 4 Conclure.

Q. 5 Que dire du cas où G est acyclique ? Si on choisit T un tri topologique, que donne un parcours de G selon le miroir de T ?

Exercice 2 : Détection de circuit

Q. 1 Donner le pseudo-code d'un algorithme basé sur un parcours en profondeur qui permet de tester si un graphe orienté est sans circuit.

Q. 2 Préciser l'implémentation de graphe la plus adaptée à cet algorithme. Donner alors la complexité de l'algorithme proposé.

Q. 3 Modifier ou compléter l'algorithme pour qu'il fournisse un circuit s'il y en a un.

Exercice 3 : Détection de graphe biparti

On rappelle qu'un graphe non orienté $G = (V, E)$ est dit **biparti** s'il existe $\{V_1, V_2\}$ une partition de V telle que les arêtes de G ont toutes une extrémité dans V_1 et l'autre dans V_2.

Q. 1 Écrire le pseudo code d'un algorithme de parcours qui teste si un graphe est biparti en essayant de le bi-colorier de manière gloutonne.

Q. 2 Dans le pseudo-code proposé, identifier quelles sont les opérations coûteuses, et en déduire :
- quelle représentation de graphe est la plus adaptée pour cet algorithme (c'est-à-dire celle pour laquelle l'algorithme sera de plus faible complexité) ;
- quelle structure de données est utilisée pour représenter les sommets visités/ouverts/fermés.

Q. 3 Implémenter une fonction `est_biparti` qui teste si un graphe est biparti.

Exercice 4 : Tri topologique

On rappelle qu'une liste L d'éléments $L_1, L_2, \dots L_n$ est un **tri topologique** (des sommets) d'un graphe orienté $G = (S, A)$ ssi $\{L_i \,|\, i \in [\![1, n]\!]\} = S$ et $\forall (i, j) \in [\![1, n]\!]^2,\ i < j \Rightarrow (L_j, L_i) \notin A$. On dit parfois que L est une permutation des sommets dans laquelle aucun sommet n'est placé après l'un de ses successeurs, ou encore dans laquelle chaque sommet est placé avant tout ses successeurs.

Q. 1 Donner un exemple de tri topologique d'un graphe orienté qui n'est pas un parcours.

Q. 2 Donner un exemple de parcours d'un graphe orienté qui n'est pas un tri topologique.

Q. 3 Que peut-on dire du premier sommet d'un tri topologique. Combien a-t-il de prédécesseurs ? de successeurs ? Et le dernier ?

Q. 4 Existe-t-il toujours un tri topologique ? Donner une condition nécéssaire et suffisante à l'existence d'un tri topologique dans un graphe.

Q. 5 Proposer un algorithme en pseudo-code qui permet de calculer un tri topologique s'il en existe un, et qui donne une preuve (au sens témoin, pas au sens démonstration) qu'il n'en existe pas sinon. *On peut essayer de construire le tri topologique pas à pas en remarquant que si $(L_i)_{i \in [\![1,n]\!]}$ est un tri topologique de $G = (S, A)$, alors $(L_i)_{i \in [\![2,n]\!]}$ est un tri topologique du graphe induit par $S \setminus \{L_1\}$.*

Q. 6 En utilisant la représentation qui permet la complexité la plus intéressante, implémenter l'algorithme proposé. *On attend une complexité pire cas en $O(n^2)$.*

Q. 7 Comment détecter si un graphe orienté présente des circuits.

Q. 8 Proposer un algorithme, de complexité $\mathcal{O}(|S| + |A|)$ permettant de calculer un tri topologique dans un graphe orienté $G = (S, A)$.