Table des matières

1 Décomposition en composantes connexes

1.1 Graphe non-orienté

Considérons un graphe non-orienté $G = (V, E)$. Ce graphe n'est pas nécessairement connexe, mais on peut le décomposer en différentes composantes avec chacune des composante connexe.

Définition 1. Une partie $X \subseteq V$ d'un graphe $G = (V, E)$ est dite *connexe*, si pour tous $x, y \in X$, avec $x \neq y$, il existe $v_1, \ldots, v_n \in X$ (pour un certain n) tel que

$$x - v_1 - v_2 - \cdots - v_{n-1} - v_n - y,$$

où l'on note $a - b$ lorsque $\{a, b\} \in E$.

On dit qu'un graphe $G = (V, E)$ est *connexe* si V est connexe.

On notera $a \mathrel{-\!\!\star} b$ lorsqu'il existe un chemin (possiblement vide dans le cas $a = b$) de a vers b.

Q1. Montrer que $-\!\!\star$ est une relation d'équivalence sur V.

Q2. Reformuler la définition 1 en terme de classe d'équivalence de $-\!\!\star$.

La décomposition de G en composantes connexes est possible car les classes d'équivalences de $-\!\!\star$ forment un partitionnement de V. La question est maintenant de déterminer ce partitionnement.

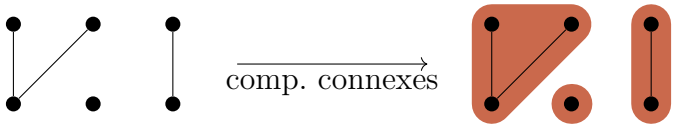


Figure 1 | *Composantes connexes dans un graphe*

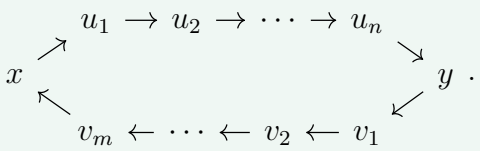
Q3. Comment peut-on calculer simplement la composante connexe contenant un sommet $v \in V$ fixé ?

Q4. En déduire une méthode pour calculer le partitionnement en composantes connexes.

1.2 Graphe orienté – algorithme de Kosaraju

Considérons un graphe $G = (V, E)$ orienté.

Définition 2. Une partie $X \subseteq V$ est *fortement connexe* si, pour tout $x, y \in V$ avec $x \neq y$, il existe $u_1, \ldots, u_n$ et $v_1, \ldots, v_m$ (pour un certain couple (n, m)) tels que



On note alors $x \overset{\star}{\rightleftarrows} y$ lorsqu'il existe un chemin de x à y (noté $x \to^\star y$) et un chemin de y à x (noté $x \,^\star\!\leftarrow y$). Ces deux chemins peuvent être distincts.

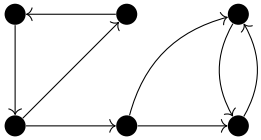


Figure 2 | *Un graphe orienté*

Q5. Justifier brièvement que $\overset{\star}{\rightleftarrows}$ est une relation d'équivalence sur V. Que sont les classes d'équivalences de cette relation ?

Q6. En appliquant la méthode utilisée en question 4 au graphe en figure 2, que donne le partitionnement ?

Dommage, on n'a pas les composantes fortement connexes. Il faut travailler un peu plus pour les trouver...

Un parcours d'un graphe (en largeur, en profondeur, peu importe) induit une forêt, c'est-à-dire un graphe non-orienté dont les composantes connexes sont des arbres. Autrement dit, une forêt, c'est un graphe acyclique car un arbre est un graphe non-orienté acyclique et connexe.

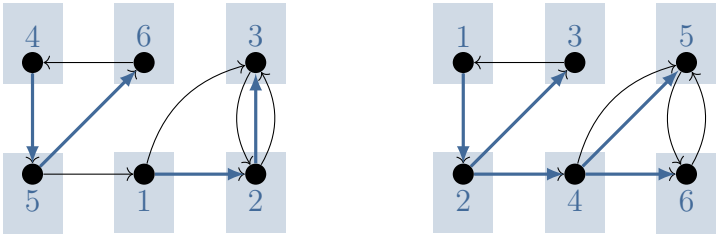


Figure 3 | *Deux forêts de parcours*

Avant de passer à l'algorithme pour décomposer G en composantes fortement connexes, on doit réaliser un *parcours en profondeur daté*.

Lors d'un parcours en profondeur, on a deux cas :

- ▷ soit on continue d'explorer le sous-arbre, et dans ce cas, on ne fait rien de plus ;
- ▷ soit on passe à un nouveau sous-arbre (s'applique aussi si l'on n'a rien exploré avant), et dans ce cas, on note la « date de fin » pour l'ancien sous-arbre, et on note la « date de début » pour le sous-arbre suivant.

Dans la figure 3, on n'a noté que les dates de débuts. On rajoute les dates de fin, comme montré dans la figure 4. On note « date de début »/« date de fin ».

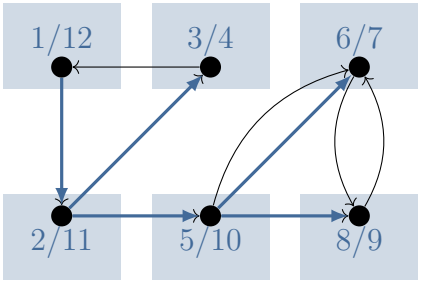


Figure 4 | *Parcours en profondeur daté*

Concrètement, les dates de début ne nous intéressent pas. Par contre, à l'aide des dates de fin (en les regardant dans l'ordre décroissant), on peut obtenir un ordre de parcours des sommets qui nous sera très utile.

On notera $\mathrm{fin}(x)$ la date de fin d'un sommet x à partir d'un parcours en profondeur daté.

Définition 3. Étant donné $G = (V, E)$, le *graphe quotient* est le graphe $\hat{G} = (\hat{V}, \hat{E})$ défini par :

- $\hat{V}$ est l'ensemble des composantes fortement connexes de G, *i.e.* les classes d'équivalences pour $\overset{\star}{\rightleftarrows}$;
- $\hat{E}$ est l'ensemble défini comme

$$\hat{E} = \left\{ (C_1, C_2) \in \hat{V} \times \hat{V} \;\middle|\; \exists x \in C_1, \exists y \in C_2, (x, y) \in E \right\}.$$

Figure 5 | *Graphe quotient*

Q7. Montrer que le graphe $\hat{G}$ est acyclique (on pourra procéder par l'absurde).

Définition 4. Un *tri topologique* d'un graphe $G = (V, E)$ est une relation d'ordre totale $\preceq$ sur V telle que si $x \to y$ alors $x \preceq y$.

Dans un graphe contenant un cycle, il est impossible de trouver un tri topologique.

Q8. Trouver un ordre topologique du graphe en figure 6.

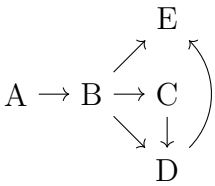


Figure 6 | *Un autre graphe (acyclique) exemple*

Q9. On réalise un parcours en profondeur daté dans un graphe acyclique, et on construit l'ordre $\preceq$ défini par : $x \preceq y$ si et seulement si $\text{fin}(x) \geq \text{fin}(y)$. Montrer que l'on a bien construit un ordre topologique.

On définit $\text{fin}(X)$ pour $X \subseteq V$ par $\text{fin}(X) := \max_{x \in X} \text{fin}(x)$. Ainsi, la date $\text{fin}(X)$ correspond à la date où l'on a fini le parcours de X.

Q10. Démontrer que, si $C \to_{\hat{G}} C'$ alors $\text{fin}(C) > \text{fin}(C')$.

Une implémentation optimisée de l'algorithme 1 peut se faire avec une complexité en $O(|V| + |E|)$. En effet, l'étape 1 se fait en $O(|V| + |E|)$, l'étape 2 se fait en $O(|E|)$, et l'étape 3 se fait en $O(|V| + |E|)$.

Algorithme 1 Algorithme de Kosaraju

1 : Réaliser un parcours en profondeur daté de G.
2 : Inverser les arêtes de G pour trouver $G^\intercal$
3 : Réaliser un parcours en profondeur de $G^\intercal$ en regardant les sommets de $G^\intercal$ dans l'ordre des dates de fin décroissantes.

L'étape 2 n'est pas *vraiment* nécessaire : il suffit, dans le parcours de l'étape 3, de regarder les arêtes « à l'envers ».

Q11. Montrer, par récurrence sur n, que les n premiers arbres parcourus dans l'étape 3 sont des composantes connexes.
Dans l'hérédité, en notant $x \in V$ le sommet utilisé pour débuter le parcours, A_x l'arbre engendré par le parcours débuté à x, et C_x la composante fortement connexe contenant x, on pourra montrer $A_x = C_x$ par double-inclusion.

De cela, on en déduit que l'algorithme de Kosaraju (algorithme 1) est correct.

Q12. Déterminer les composantes fortement connexes de l'arbre donné en figure 7. On utilisera exactement l'algorithme de Kosaraju, en détaillant chaque étape.

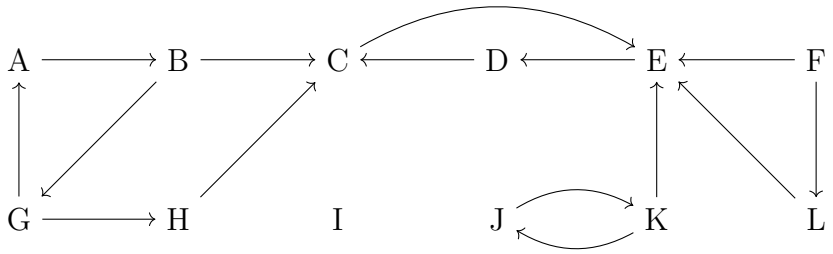


Figure 7 | *Graphe pour la question 12*

2 Arbre couvrant de poids minimal

Dans cette partie, on s'intéresse à des graphes non-orientés pondérés. Un tel graphe est un 3-uplet $G = (V, E, p)$ où $p : E \to \mathbb{N}$, où p est une fonction dite *de pondération* (positive).

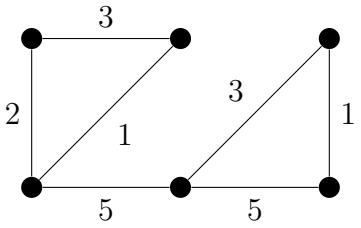


Figure 8 | *Graphe non-orienté pondéré*

Définition 5. Étant donné $G' = (V', E')$, on dit que G' est un *arbre* si G' est connexe et acyclique.

Q13. Montrer que les trois points suivants sont équivalents :

1. $G' = (V', E')$ est connexe et acyclique ;
2. $G' = (V', E')$ est connexe et $|E'| = |V'| - 1$;
3. $G' = (V', E')$ est acyclique et $|E'| = |V'| - 1$.

Définition 6. On dit que $G' = (V', E')$ est un *arbre couvrant* d'un graphe $G = (V, E)$ si

- $V = V'$;
- $E' \subseteq E$;
- G' est un arbre.

Définition 7. Étant donné $G = (V, E, p)$ un graphe pondéré, le

poids d'un sous-graphe $G' = (V', E')$ est

$$p(G') := \sum_{e \in E'} p(e).$$

Le problème que l'on essaie de résoudre est le suivant :

ACPM | **Entrée.** Un graphe pondéré $G = (V, E, p)$;
Sortie. Un arbre couvrant G' de poids $p(G')$ minimal.

Pour résoudre ce problème, on cherche un arbre couvrant, donc la seule donnée qui nous intéresse est E' (car, à la fin, on a $V = V'$).

L'algorithme utilisé pour résoudre ce problème est l'*algorithme de Kruskal*. C'est un algorithme *glouton*.

Algorithme 2 Algorithme de Kruskal (résolvant ACPM)

```
1 : E' ← ∅
2 : On trie E par ordre croissant des poids
3 : pour {x, y} ∈ E dans l'ordre 2 faire
4 :     si x ≠*_{E'} y alors
5 :         E' ← E ∪ {{x, y}}
6 : retourner G' = (V, E')
```

Démontrons la correction totale (*i.e.* correction et terminaison) de l'algorithme de Kruskal (algorithme 2).

Q14. Montrer la terminaison de l'algorithme 2. Quelle est sa complexité ?

Q15. Montrer que le graphe G' retourné par l'algorithme 2 est un arbre.

Il ne reste qu'à montrer que l'algorithme de Kruskal retourne un arbre de poids minimal, un arbre optimal. Pour cela, c'est souvent la même

méthode avec les algorithmes glouton : on rapproche l'optimal du résultat donné par l'algorithme glouton. Dans la suite, on notera ACPM pour arbre couvrant de poids minimal.

On se propose de montrer l'invariant suivant :

$$\text{il existe un ACPM } T \text{ tel que } G' \subseteq T \tag{1}$$

On note ici $(V, E) \subseteq (V', E')$ lorsque (V, E) est un *sous-graphe* de (V', E'), *i.e.* lorsque $V \subseteq V'$, $E \subseteq E'$ et $E \subseteq \wp_2(V)$, où $\wp_2(V)$ est l'ensemble des paires ($\neq$ couples) d'éléments de V.

Q16. Prouver l'invariant (1).

Q17. Conclure quant à la correction totale de l'algorithme 2.

Q18. Donner un ACPM sur le graphe en figure 8 en utilisant l'algorithme de Kruskal. On détaillera les étapes de l'algorithme.

3 Quelques problèmes NP-complets sur les graphes.

On rappelle le problème

n-SAT	**Entrée.** Une formule φ sous n-FNC **Sortie.** La formule φ est-elle satisfiable ?

Q19. Pour quelles valeurs de n, n-SAT est-il **NP**-complet ?

3.1 Problème k-COLORATION.

[ORAUX CCINP MPI 2023]

Définition 8. On dit d'un graphe qu'il est *coloriable* si on peut colorier tous les sommets du graphe tels que deux sommets voisins sont de couleurs différentes. Un graphe est dit *k-coloriable* s'il est coloriable avec k couleurs.

On considère le problème suivant :

k-COLORATION	**Entrée.** Un graphe non-orienté $G = (V, E)$ **Sortie.** Existe-t-il une k-coloration de G ?

On se propose de montrer que k-COLORATION se réduit au problème k-SAT.

Dans le but de construire cette réduction, on introduit quelques notations. Soit $G = (V, E)$ un graphe non-orienté, et $\mathscr{C}$ un ensemble de k couleurs. On crée les variables $p_{v,c}$ pour $v \in V$ et $c \in \mathscr{C}$.

Q20. On commence par construire une FNC qui traduit le fait que deux sommets voisins ne peuvent pas être de la même couleur. On procèdera ainsi :

1. Pour u et v deux sommets voisins et une couleur c, construire une clause qui traduit que i et j ne peuvent pas être de la même couleur c.
2. Créer une 2-FNC qui traduit que u et v ne peuvent pas être de la même couleur.
3. Conclure quant au but initial de la question.

Q21. On souhaite traduire le fait qu'un graphe est k-coloriable. Pour cela, il est nécessaire que les sommets voisins ont des couleurs différentes, et que tous les sommets soient coloriés.

1. Traduire cette deuxième condition pour le graphe G sous la forme d'une FNC.
2. Conclure en construisant une FNC qui traduit qu'un graphe G est k-coloriable.

Q22. Montrer que k-COLORATION se réduit au problème k-SAT.

Q23. Montrer que k-COLORATION est dans **NP**. Pour quelles valeurs de k le problème k-COLORATION est-il **NP**-complet ?

3.2 Problèmes CLIQUE et STABLE.

Définition 9. Étant donné un graphe $G = (V, E)$, on dit qu'un sous-ensemble $V' \subseteq V$ est

- ▷ une *clique* si les sommets de V' sont deux-à-deux reliés ;
- ▷ un *stable* si les sommets de V' sont deux-à-deux non-reliés.

La *taille* d'une clique ou d'un stable est $|V'|$.

On considère les deux problèmes suivants :

CLIQUE
Entrée. Un graphe $G = (V, E)$ et $k \in \mathbb{N}$
Sortie. Existe-t-il une clique de taille $\geq k$?

STABLE
Entrée. Un graphe $G = (V, E)$ et $k \in \mathbb{N}$
Sortie. Existe-t-il un stable de taille $\geq k$?

Q24. Montrer que CLIQUE se réduit polynomialement à STABLE, puis que STABLE se réduit polynomialement à CLIQUE.

On se propose de montrer la **NP**-complétude du problème CLIQUE par réduction polynomiale depuis le problème 3-SAT.

Soit φ une formule sous forme 3-CNF. On note n le nombre de clauses de φ. Soient $(\ell_{i,j})_{i\in[\![1,n]\!],j\in\{1,2,3\}}$ les $3n$ littéraux tels que

$$\varphi = \bigwedge_{i=1}^{n} (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}).$$

On construit le graphe $G = (V, E)$ avec :

$$V = \{(i,j) \mid i \in [\![1,n]\!], j \in \{1,2,3\}\}$$
$$E = \left\{ \{(i_1,j_1),(i_2,j_2)\} \;\middle|\; \ell_{i_1,j_1} \neq \neg\ell_{i_2,j_2} \text{ et } \neg\ell_{i_1,j_1} \neq \ell_{i_2,j_2} \text{ et } i_1 \neq i_2 \right\}$$

Q25. Représenter le graphe construit à partir de la formule

$$(x \vee x \vee y) \wedge (\neg x \vee \neg y \vee \neg y) \wedge (\neg x \vee y \vee y).$$

Q26. Montrer que le problème 3-SAT se réduit polynomialement au problème CLIQUE.