

# Graphes

## 1 Définitions

### 1.1 Graphes

Un graphe **non orienté** est la donnée d'un couple  $G = (V, E)$ , où  $V$  est un ensemble fini non vide et  $E \subset \{\{x, y\} \mid (x, y) \in V^2\}$ . Un graphe **orienté** est la donnée d'un couple  $H = (S, A)$ , où  $S$  est un ensemble fini non vide et  $A \in \mathcal{P}(S^2)$ .

Les éléments de  $V$  et  $A$  sont appelés **sommets du graphe**. On parlera, pour désigner les éléments de  $E$  dans un graphe non-orienté d'**arêtes** du graphe, et pour un graphe orienté d'**arcs**.

Si  $e = \{x\} \in E$  (avec  $x \in V$ ),  $e$  est une **boucle** sur  $x$  (idem pour  $e = (x, x)$  pour  $x \in S$ ). Pour  $(x, y) \in V^2$ , on dit que  $x$  et  $y$  sont **voisins** ssi  $\{x, y\} \in E$ . Cette notion se précise dans un graphe orienté : on dit que  $x \in S$  est un **successeur** (resp. **prédécesseur**) de  $y$  ssi  $(y, x) \in A$  (resp.  $(x, y) \in A$ ).

On appelle **voisinage** du sommet  $x$  l'ensemble  $\mathcal{V}(x)$  de ses voisins. Le **degré** de  $x$ , noté  $\deg x$ , est le cardinal de ce voisinage.

Pour les graphes orientés, on distingue le **degré sortant** de  $x$ , noté  $\deg^+ x$ , le nombre de successeurs de  $x$ , du **degré entrant** de  $x$ , noté  $\deg^- x$ , le nombre de prédécesseurs de  $x$ .

*On supposera par la suite que l'on travaille avec des graphes sans boucles.*

**Propriété :** Soit  $G = (V, E)$  un graphe. On a  $\sum_{x \in V} \deg(x) = 2 \text{ card}(E)$

▷ On a :

$$\sum_{x \in V} \deg(x) = \sum_{x \in V} \sum_{y \in V} \mathbb{1}_{\{x, y\}} = \sum_{(x, y) \in V^2} \mathbb{1}_{\{x, y\}} = 2 \sum_{\{x, y\} \in V^2} \mathbb{1}_{\{x, y\}} = 2 \text{ card}(E).$$

Car le graphe est sans boucle. Il faudrait sinon ajouter le nombre de boucles présentes dans le graphe.

### 1.2 Accessibilité, connexité

*On fixe  $G = (V, E)$  un graphe non orienté, et  $H = (A, S)$  un graphe orienté.*

Soit  $s = (s_i) \in V^{n+1}$ . On dit que  $s$  est une **chaîne de**  $G$  ssi  $\forall i \in \llbracket 1, n \rrbracket, \{s_i, s_{i+1}\} \in E$ . On dit alors que  $s$  est une chaîne de longueur  $n$  et qui relie  $s_0$  et  $s_n$ .

Soit  $s = (s_i) \in A^{n+1}$ . On dit que  $s$  est un **chemin de**  $G$  ssi  $\forall i \in \llbracket 1, n \rrbracket, \{s_i, s_{i+1}\} \in E$ . On dit alors que  $s$  est une chaîne de longueur  $n$  et qui relie  $s_0$  à  $s_n$ .

On dit alors que  $s_n$  est **accessible** depuis  $s_0$ . Par ailleurs, si  $s_n = s_0$ , on dit que  $s$  est un **cycle** pour un graphe non-orienté, ou un **circuit** dans un graphe orienté.

Si tous les  $(s_i)$  sont distincts, on dit que  $s$  est **élémentaire**.

**Remarque :** Il y a toujours un nombre fini de chaînes élémentaires, mais si  $G$  (resp.  $H$ ) a des cycles (resp. des circuits), il y a un nombre infini de chaînes (il suffit de tourner en rond...).

**Exercice 1:** Définir la relation entre les circuits/chemins d'un graphe, qui met en relation deux circuits/chemins ssi ils relient les mêmes sommets. Est-ce une relation d'équivalence ?

**Propriété :** La relation  $\leftrightarrow$  définie sur  $V^2$  par  $x \leftrightarrow y$  ssi  $x$  est accessible depuis  $y$  est une relation d'équivalence.

- ▷ Soit  $x \in V$ . On a bien  $x \leftrightarrow x$  : la chaîne de longueur  $n = 0$   $s = (x)$  convient.
- ▷ Soit  $(x, y) \in V^2$ , tel que  $x \leftrightarrow y$ . Alors par définition il existe  $s = (s_0, \dots, s_n) \in V^{n+1}$  tel que  $s_0 = x$  et  $s_n = y$ , et  $\forall i \in \llbracket 0, n \rrbracket, \{s_i, s_{i+1}\} \in E$ . Considérons  $s' = (s_n, s_{n-1}, \dots, s_1, s_0)$ .  $s'$  est une chaîne reliant  $y$  et  $x$ . En effet,  $s'_0 = s_n = y$  et  $s'_n = s_0 = x$ , et  $\forall i \in \llbracket 0, n \rrbracket, \{s'_i, s'_{i+1}\} = \{s_{n-i}, s_{n-i-1}\} = \{s_k, s_{k+1}\} \in E$  en posant  $k = n - i - 1 \in \llbracket 0, n \rrbracket$ .
- ▷ Soit  $(x, y, z) \in V^3$  tel que  $x \leftrightarrow y$  et  $y \leftrightarrow z$ . Comme  $x \leftrightarrow y$ , il existe  $s = (s_0, \dots, s_n) \in V^{n+1}$  une chaîne avec  $s_0 = x$  et  $s_n = y$ . Comme  $y \leftrightarrow z$ , il existe  $t = (t_0, \dots, t_m) \in V^{m+1}$  une chaîne avec  $t_0 = y$  et  $t_m = z$ . Considérons  $u = (s_0, \dots, s_n, t_1, \dots, t_m)$ .  $u$  est bien une chaîne car  $\forall i \in \llbracket 0, n+m \rrbracket$ , soit  $i \in \llbracket 0, n \rrbracket$  et dans ce cas on a  $\{u_i, u_{i+1}\} = \{s_i, s_{i+1}\} \in E$ , soit  $i = n$  et on a  $\{u_i, u_{i+1}\} = \{s_n, t_1\} \in E$ , soit  $i \in \llbracket n+1, n+m \rrbracket$  et  $\{u_i, u_{i+1}\} = \{t_{i-n}, t_{i-n+1}\} \in E$ . On a par ailleurs  $u_0 = x$  et  $u_{n+m} = z$ , donc  $x \leftrightarrow z$ .

**Exercice 2:** Définir une relation d'équivalence similaire pour  $H$ , où l'on doit avoir un chemin dans chaque sens entre deux points en relation.

Une **composante connexe** de  $G$  est une classe d'équivalence pour la relation d'équivalence définie ci-dessus. Si  $G$  n'admet qu'une composante connexe, on dit que  $G$  est un **graphe connexe**. Dans le cas de la relation d'équivalence sur les graphes orientés, on appelle les classes d'équivalences **composante fortement connexe**.

Soit  $W \subset V$  avec  $W \neq \emptyset$ .  $W$  est **connexe** ssi  $\forall (x, y) \in W^2$ , il existe une chaîne reliant  $x$  et  $y$ .

**Propriété :**  $W$  est une composante connexe ssi  $W$  est connexe minimal, c'est à dire si  $\forall W' \subset V \setminus \{W\}, W \subset W', W'$  n'est pas connexe.

Soit  $G' = (V', E')$  un graphe.  $G'$  est un **sous-graphe** ssi  $V' \subset V, E' \subset E$ .

Soit  $V' \subset V$  Le **graphe induit par  $G$  sur  $V'$**  est  $G' = (V', \{\{x, y\} \in E \mid (x, y) \in V'^2\})$ .

**Propriété :** Un ensemble de sommets est connexe ssi le graphe qu'il induit est connexe.

- ▷ En exercice.

## 1.3 Types de graphes

Un graphe non-orienté (resp. orienté) est dit **acyclique** s'il ne contient aucun cycle élémentaire (resp. aucun circuit).

Un **arbre** est un graphe connexe acyclique. (cf TD pour caractérisation).

Un graphe acyclique décomposé en ses composantes connexes (qui sont donc des arbres), est appelé **forêt**.

Un graphe non-orienté  $(V, E)$  est dit **bipartite** ssi il existe une partition  $\{W_1, W_2\}$  de  $V$  telle que toutes les arêtes de  $E$  aient une extrémité dans  $W_1$  et l'autre dans  $W_2$ .

**Algorithme pour décider si un graphe est bipartite :** Il suffit de créer deux ensembles  $W_1$  et  $W_2$ . On prend un point au hasard dans le graphe, et on le place dans  $W_1$ . On place alors tous les voisins de  $x$  dans  $W_2$ , puis tous les voisins des voisins de  $x$  dans  $W_1$ , et ainsi de suite, récursivement. Si jamais il y a conflit (on doit placer un élément dans  $W_1$  alors qu'il est déjà présent dans  $W_2$  par exemple), alors le graphe n'est pas bipartite. Une fois que l'on ne peut plus ajouter d'éléments à un des deux ensembles, si des points du graphe n'ont toujours pas été ajoutés, on recommence le processus sur l'un de ces points, jusqu'à ce que tous les points aient été ajoutés. Si le processus s'est fait sans conflit, le graphe est bipartite.

## 2 Parcours

### 2.1 Définitions

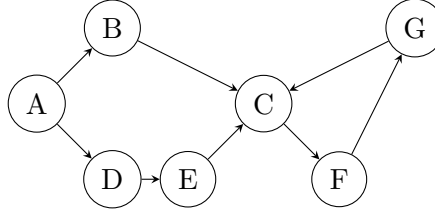
On définit, pour  $W \subset V$  la bordure de  $W$  par :

$$\mathcal{B}(W) = \{y \in V \setminus W \mid \exists x \in W, x, y \in E\}$$

Dans le cadre d'un graphe orienté, pour  $T \subset S$  :

$$\mathcal{B}(T) = \{y \in V \setminus T \mid \forall x \in T, y \in \mathcal{V}(x)\}$$

On dit que  $L \in V^n$  (ou  $S^n$  pour un graphe orienté) est un parcours ssi  $\forall i \in \llbracket 1, n \rrbracket, L_i \in \mathcal{B}(\{L_j \mid j \in \llbracket 1, i \rrbracket\})$ .

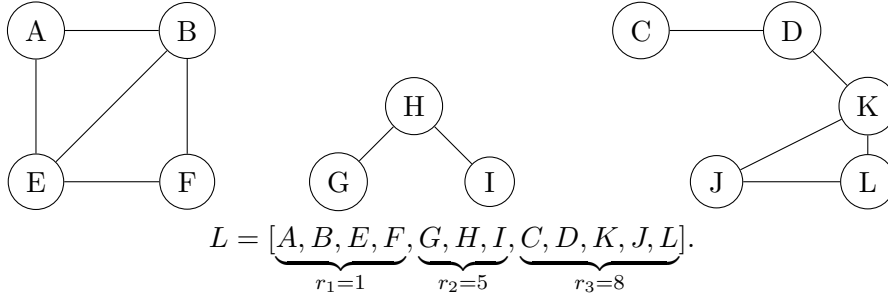


Un parcours du graphe orienté ci-dessus est  $L = [F, G, C, E, A, B, D]$

De plus, on dit que  $L_i$  est un **point de régénération du parcours** ssi  $\mathcal{B}(\{L_j \mid j \in \llbracket 1, i \rrbracket\}) = \emptyset$ . [? ?]

Enfin, en notant  $\mathcal{R}$  l'ensemble des points de régénération de  $L$ , on dit que  $F = (V, P)$  est une forêt d'arborescences associée au parcours  $L$  ssi  $F$  respecte les trois propriétés suivantes :

1.  $\forall i \in \llbracket 1, n \rrbracket, L_i \in \mathcal{R}$  ou  $\exists j \in \llbracket 1, i \rrbracket, L_i \in \mathcal{V}(L_j), (L_j, L_i) \in P$  ;
2.  $\forall u \in V \setminus \mathcal{R}, \exists ! w \in \mathcal{R}, (w, u) \in P$  (on dit alors que  $w$  est le **père** de  $u$ ) ;
3.  $F = (V, P)$ .  $P$  est minimal parmi les ensemble vérifiant 1.



**Propriété :** Soit  $G = (V, E)$  un graphe non-orienté. Soit  $W \subset V$ . Si  $\mathcal{B}(W) = \emptyset$ , alors il n'existe aucune chaîne reliant un sommet de  $W$  et un sommet de  $V \setminus W$ .

- ▷ Par l'absurde, supposons qu'il existe une chaîne  $\gamma$  de longueur  $l$  telle que  $\gamma_0 \in W$  et  $\gamma_l \in V \setminus W$ . On a  $\gamma_0 \neq \gamma_l$  donc  $l > 0$ . On peut alors définir  $i_0 = \min \{i \in \llbracket 1, l \rrbracket \mid \gamma_i \notin W\}$ . Par définition de  $i_0$ ,  $\gamma_{i_0-1} \in W$ . Par définition, une chaîne  $\{\gamma_{i_0}, \gamma_{i_0}\} \in E$ , autrement dit  $\gamma_{i_0} \in \mathcal{V}(\gamma_{i_0-1})$ . Donc  $\gamma_{i_0} \in \mathcal{B}(W)$  : absurde.

**Propriété :** Soit  $G = (V, E)$  un graphe non-orienté, et  $L = (L_i)_{i \in \llbracket 1, n \rrbracket}$  un parcours de  $G$ . Si l'ensemble des points de régénération s'écrit  $R = \{L_{r_k} \mid k \in \llbracket 1, K \rrbracket\}$  avec  $(r_k)$  croissant, alors  $G$  admet  $K$  composantes connexes, à savoir les  $(C_k)_{k \in \llbracket 1, K \rrbracket}$  définis par  $C_k = \{L_i \mid i \in \llbracket r_k, r_{k+1} \rrbracket\}$  avec  $r_{K+1} = n + 1$ .

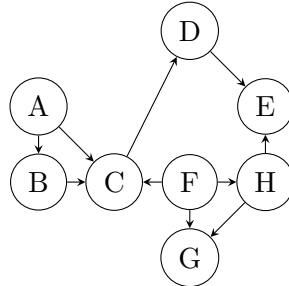
- ▷ Soit  $k \in \llbracket 1, K \rrbracket$ . Montrons que  $C_k$  est connexe maximal.
- ▷ Si  $C_k = V$ , il est trivialement connexe. Sinon, soit  $u \in V \setminus C_k$ . MQ  $\hat{C}_k = C_k \cup \{u\}$  n'est pas connexe. Par définition d'un parcours. Il existe  $i_u \in \llbracket 1, n \rrbracket$  tq  $u = L_{i_u}$ . Comme  $u \notin C_k$ ,  $i_u \notin \llbracket r_k, r_{k+1} \rrbracket$ . Si  $i_u < r_k$ , on pose  $W = \{L_i \mid i \in \llbracket 1, r_k \rrbracket\}$ . Ainsi,  $L_{i_u} \in W$  et  $\mathcal{B} = \emptyset$  car  $L_{i_k}$  est point de régénération. D'après le lemme, il n'existe aucun chemin reliant  $L_{i_u}$  et  $L_{i_k} \notin W$  donc  $\hat{C}_k$  n'est pas connexe. Autre cas... Ainsi  $C_k$  est maximal.
- ▷ Par l'absurde, on suppose que  $C_k$  n'est pas connexe. On peut donc dire qu'il existe  $i \in \llbracket r_k, r_{k+1} \rrbracket$  tel que  $L_{r_k} \not\leftrightarrow L_i$ . On considère alors  $i_0 = \min \{j \in \llbracket r_k, r_{k+1} \rrbracket \mid L_j \not\leftrightarrow L_{r_k}\}$ . Par minimalité de  $i_0$ ,  $\forall j \in \llbracket r_k, i_0 \rrbracket, L_{r_k} \leftrightarrow L_j$ . donc  $\forall j \in \llbracket r_k, i_0 \rrbracket, L_j \not\leftrightarrow L_{i_0}$  (sinon on aurait  $L_{i_0} \leftrightarrow L_j \leftrightarrow L_{r_k}$ ). De plus, puisque  $L_{r_k} \in R$ ,  $\mathcal{B}(\{L_j \mid j \in \llbracket 1, r_k \rrbracket\}) = \emptyset$ . Donc d'après le lemme,  $\forall j \in \llbracket 1, r_k \rrbracket, L_{i_0} \not\leftrightarrow L_j$ . Donc  $L_{i_0} \notin \mathcal{B}(\{L_j \mid j \in \llbracket 1, i_0 \rrbracket\})$ . Par définition d'un parcours on a nécessairement  $\mathcal{B}(\dots) = \emptyset$  donc  $L_{i_0} \in R$  : Absurde, car  $L_{r_k}$  et  $L_{r_{k+1}}$  sont 2 points de régénération consécutifs.

Soit  $L = (L_i)_{i \in \llbracket 1, n \rrbracket}$  un parcours de  $G$ . Pour  $k \in \llbracket 0, n \rrbracket$ , on appelle **sommet ouvert à l'étape  $k$**  (dans  $L$ ) un sommet de  $O_k = \{L_j \mid j \in \llbracket 1, k \rrbracket \text{ et } \mathcal{V}(L_j) \not\subset \{L_i \mid i \in \llbracket 1, k \rrbracket\}\}$ .

$L$  est un **parcours en largeur** (resp. profondeur) de  $G$  ssi  $\forall k \in \llbracket 1, n \rrbracket$ ,  $L_k$  est un point de régénération ou bien  $L_k \in \mathcal{V}(L_{i_0})$  où  $i_0 = \min \{i \in \llbracket 1, n \rrbracket \mid L_i \in O_k\}$  (resp. max). Autrement, chaque sommet du parcours est point de régénération ou bien voisin du premier (resp. dernier) sommet ouvert.

**Remarque :** Lorsque l'on s'intéresse à la forêt d'arborescence associée à un parcours en largeur / profondeur, on choisira comme paire d'un sommet  $L_k$  le premier / dernier sommet ouvert à l'étape  $k$ . Autrement dit, la forêt justifiera que le parcours est en largeur / profondeur.

**Exemple :** Un parcours en largeur du graphe ci-dessous est ABCDEFHG, et un parcours en profondeur de ACDEBFHG.



### 3 Algorithmes de parcours

#### 3.1 Détection de composantes connexes.

$G = (V, E)$  avec  $V = \llbracket 1, n \rrbracket$  un graphe non orienté.

```

1  n <- nb_sommets(G)
2  O <- ensemble des sommets initialement vides
3  F <- ensemble des sommets initialement vides
4  Tres <- tableau indicé par  $\llbracket 1, n \rrbracket$ , initialisé à -1.
5  nc <- 0 //(Numéro de la composante connexe)
6  racine <- 1
7  Tres[racine] <- nc
8  O.ajouter(racine)
9  i <- 0
10 Tant que i < n:
11     Si θ est vide:
12         racine <- un sommet S tel que Tres[s] = -1
13         nc <- nc + 1
14         O.ajouter(racine)
15         Tres[racine] <- nc
16         u <- un sommet extrait de O
17         Pour tout v voisin de u
18             Si Tres[v] = -1:
19                 O.ajouter(v)
20                 Tres[v] <- nc
21         F.ajouter(u)
22         i++
23 Renvoyer $Tres$

```

### 3.2 Détection de graphe bipartie.

### 3.3 Plus court chemin en nombre d'arcs.

Dans un graphe non-orienté, on appelle **distance entre les sommets  $a$  et  $b$** , notée  $\text{dist}(a, b)$ , la longueur minimale d'une chaîne reliant  $a$  et  $b$  (et  $+\infty$  s'il n'en existe pas). On généralise cette notion aux graphes orientés, mais il se s'agit alors plus vraiment d'une distance : la distance de  $a$  à  $b$  n'est pas forcément égale à la distance de  $b$  à  $a$  !

$G = (S, A)$  graphe orienté avec  $S = \llbracket 1, n \rrbracket$ ,  $(a, b) \in S^2$ .

```
1  n <- nb_sommets(G)
2  O1 <- pile d'entiers initialisée vide
3  O2 <- pile d'entiers initialisée vide
4  F <- ensemble initialisé vide
5  D <- tableau d'entiers indicé par  $\llbracket 1, n \rrbracket$  initialisé à  $+\infty$ 
6  P <- tableau d'entiers indicé par  $\llbracket 1, n \rrbracket$  initialisé à  $+\infty$ 
7  prof <- 0
8  O1.ajouter(a)
9  D[a] <- 0
10 P[a] <- a
11
12 Tant que O1 et O2 ne sont pas vides:
13   Si O1 est vide:
14     Transvaser O2 dans O1
15     prof = prof + 1
16   u <- O1.extraire_sommet
17   Pour v successeur de u:
18     Si D[v] =  $+\infty$ :
19       D <- D[u] + 1
20       P(v) <- u
21       O2.ajouter(v)
22   F.ajouter(u)
23 Renvoyer $D[b]$
```