Arborescences et représentations informatiques des graphes

Benoît Robillard

robillard@ensiie.fr

13 décembre 2007

Plan

Plan

1. Arborescences
   - Définitions
   - Arborescence couvrante
   - Centres et diamètres
2. Représentation des graphes
   - Matrice d'adjacence
   - Liste d'adjacence
   - Avantages et inconvénients de chaque représentation
3. Parcours des graphes
   - Parcours en profondeur d'abord
   - Parcours en largeur d'abord
   - Parcours en meilleur d'abord

Première définition

Définition : racine

Dans un graphe orienté on appelle racine un sommet r tel qu'il existe un chemin de r à tout autre sommet du graphe.

Première définition

Définition : racine

Dans un graphe orienté on appelle racine un sommet r tel qu'il existe un chemin de r à tout autre sommet du graphe.

Définition : arborescence

Une arborescence est un arbre muni d'une racine. Une arborescence où chaque sommet a au plus deux fils dont il est le père est dite binaire.

Remarque

Les "arbre informatiques" sont des arborescences.

Terminologie

feuille : nœud sans fils

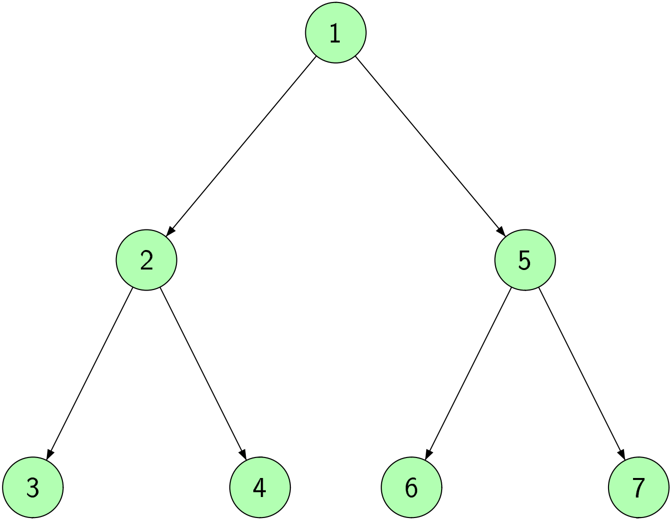
branche : chemin de la racine à une feuille

hauteur (ou profondeur) : longueur de la plus longue branche

ascendant de x : nœud situé entre r et x

descendant de x : nœud tel qu'il existe un chemin de x à ce nœud

Un exemple d'arborescence



Définitions équivalentes

Théorème : définitions équivalentes d'une arborescence

Soit $G = (V, E)$ un graphe d'ordre n ($n \geq 2$). Les définitions suivantes sont équivalentes :

1. G est un arbre admettant une racine r ; (1)
2. il existe un sommet r qui est lié à tout autre sommet par un chemin unique et élémentaire ; (2)
3. G est connexe et il existe un sommet r tel que $d^{-}(r) = 0$ et $d^{-}(x) = 1$ pour tout x de $V(G)$ tel que $x \neq r$; (3)
4. G est sans cycle et il existe un sommet r tel que $d^{-}(r) = 0$ et $d^{-}(x) = 1$ pour tout x de $V(G)$ tel que $x \neq r$. (4)

Exercice

Montrer que les quatre définitions sont équivalentes.

Définition équivalentes : démonstration (1)

On va montrer que (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (1)

(1) $\Rightarrow$ (2)
r est une racine donc il existe un chemin de r à tout autre sommet du graphe. De plus, le chemin est unique et élémentaire car sinon G posséderait un cycle et ne serait donc pas un arbre.

(2) $\Rightarrow$ (3)
Le graphe G est connexe car toute paire de sommets est reliée par une chaîne passant par r. Reste à montrer les résultats sur les degrés, nous allons le faire par l'absurde.
Supposons que $d^-(r)$ soit strictement positif. Dans ce cas il existe (au moins) un sommet s qui est prédécesseur de r, et par conséquent un cycle passant par x et r. À partir de tout chemin de x à un autre sommet on peut donc construire un autre chemin reliant ces deux sommets en passant par le cycle, ce qui est contradictoire.

Définition équivalentes : démonstration (2)

(3) $\Rightarrow$ (4)

Le nombre d'arcs m est égal à $\sum_{i=1}^{n} d^{-}(x_i) = n - 1$. De plus G est connexe donc on a $\nu(G) = m - n + p = (n-1) - n + 1 = 0$, et donc G ne possède pas de cycle.

(4) $\Rightarrow$ (1)

G est sans cycle et $\sum_{i=1}^{n} d^{-}(x_i) = n - 1$ donc $\nu(G) = m - n + p = (n-1) - n + p = 0$ d'où $p = 1$. Ainsi, G est connexe et sans cycle donc c'est un arbre.

Nous allons montrer par l'absurde que tout sommet $x \neq r$ est un descendant de r.

Soit x un sommet qui n'est pas un descendant de r. Puisque tout sommet $s \neq r$ est de degré intérieur 1 on sait que chaque ascendant de x possède lui-même un ascendant. Cela implique qu'il existe un cycle passant par x, car x ne peut avoir plus de $(n-1)$ ascendants, et contredit donc le fait que G est sans cycle.

Arborescence couvrante

Définition : arborescence couvrante

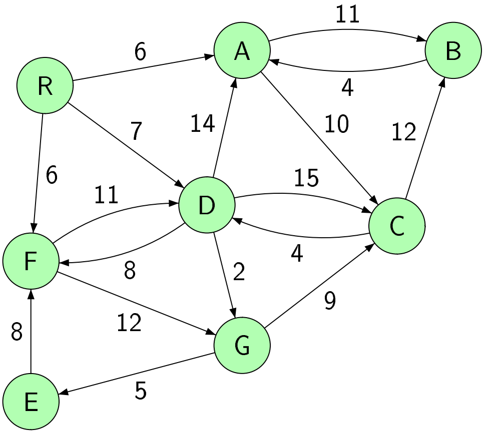
Une arborescence couvrante d'un graphe $G = (V, E)$ est une arborescence $A = (V', E')$ telle que $V'(A) = V(G)$ et $E'(A) \subseteq E(G)$.

Algorithme optimal et rapide (*cf. poly*).

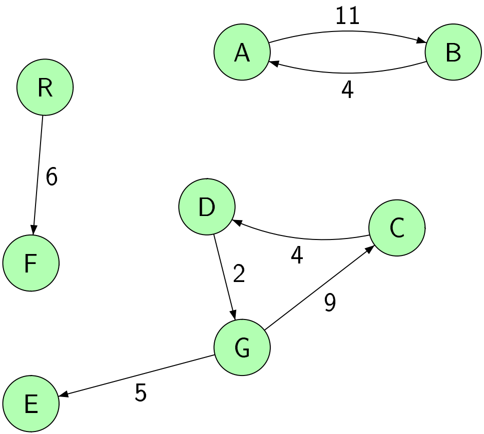
Exercice : Application de l'algorithme

Exercice

Trouver une arborescence couvrante optimal du graphe suivant



Algorithme : première itération (1)



Ce graphe a deux circuits : $\mu_1 = (AB)$ et $\mu_2 = (CDG)$.

Algorithme : première itération (2)

On calcule les nouveaux poids :

- Pour μ_1 :
  - $w(R, A) = w(R, A) - W(B, A) = 6 - 4 = 2$
  - $w(D, A) = w(D, A) - W(B, A) = 14 - 4 = 10$
  - $w(C, B) = w(C, B) - W(A, B) = 12 - 11 = 1$
- Pour μ_2 :
  - $w(A, C) = w(A, C) - W(G, C) = 10 - 9 = 1$
  - $w(R, D) = w(R, D) - W(C, D) = 7 - 4 = 3$
  - $w(F, G) = w(F, G) - W(D, G) = 12 - 2 = 10$
  - $w(F, D) = w(F, D) - W(C, D) = 11 - 4 = 7$

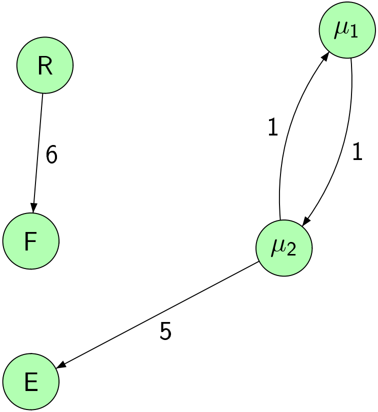
Algorithme : première itération (2)

On calcule les nouveaux poids :

- Pour μ_1 :
  - $w(R, A) = w(R, A) - W(B, A) = 6 - 4 = 2$
  - $w(D, A) = w(D, A) - W(B, A) = 14 - 4 = 10$
  - $w(C, B) = w(C, B) - W(A, B) = 12 - 11 = 1$
- Pour μ_2 :
  - $w(A, C) = w(A, C) - W(G, C) = 10 - 9 = 1$
  - $w(R, D) = w(R, D) - W(C, D) = 7 - 4 = 3$
  - $w(F, G) = w(F, G) - W(D, G) = 12 - 2 = 10$
  - $w(F, D) = w(F, D) - W(C, D) = 11 - 4 = 7$

On garde le nouvel arc de poids minimum pour chaque circuit, donc (C, B) et (A, C) puis on contracte les circuits en un seul sommet.

Algorithme : deuxième itération (1)



Ce graphe a un circuit : $\mu_3 = (\mu_1 \mu_2)$.

Algorithme : Deuxième itération (2)

On calcule les nouveaux poids de μ_3 :

- $w(R, A) = w(R, A) - W(\mu_2, \mu_1) = 6 - 1 = 5$
- $w(R, D) = w(R, D) - W(\mu_1, \mu_2) = 7 - 1 = 6$
- $w(F, D) = w(F, D) - W(\mu_1, \mu_2) = 11 - 1 = 10$
- $w(F, G) = w(F, G) - W(\mu_1, \mu_2) = 12 - 1 = 11$

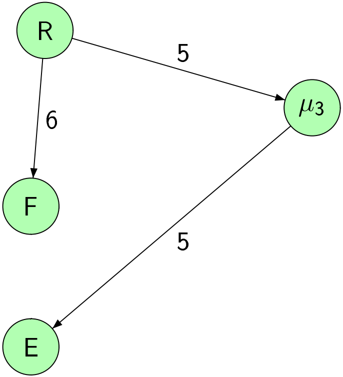
Algorithme : Deuxième itération (2)

On calcule les nouveaux poids de μ_3 :

- $w(R, A) = w(R, A) - W(\mu_2, \mu_1) = 6 - 1 = 5$
- $w(R, D) = w(R, D) - W(\mu_1, \mu_2) = 7 - 1 = 6$
- $w(F, D) = w(F, D) - W(\mu_1, \mu_2) = 11 - 1 = 10$
- $w(F, G) = w(F, G) - W(\mu_1, \mu_2) = 12 - 1 = 11$

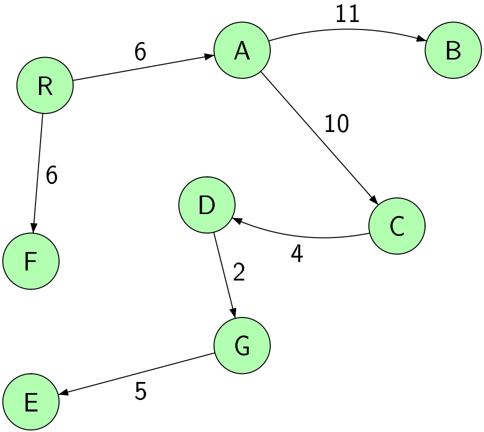
On garde le nouvel arc de poids minimum, donc (C, B), puis on contracte le circuit en un seul sommet.

Algorithme : finalisation (1)



Ce graphe est sans circuit : reste à redéployer les circuits.

Algorithme : finalisation (2)



Le poids de l'arborescence est $6+6+11+10+4+2+5 = 44$.

Définitions (1)

Définition : Graphe quasi-fortement-connnexe

Un graphe G est quasi-fortement-connexe s'il admet une racine. Par exemple les arborescences sont exactement les arbres quasi-fortement-connexes.

Définition : écart

L'écart entre deux sommets x et y, noté $e(x, y)$ d'un graphe quasi-fortement-connexe G est défini comme suit :

- $e(x, y) = 0$ si $y = x$;
- $e(x, y) =$ nombre min d'arcs de x à y si y descendant de x ;
- $e(x, y) = +\infty$ sinon.

Définitions (2)

Définitions : écartement,centre, rayon et diamètre

L'écartement d'un sommet x d'un graphe quasi-fortement-connexe G est le maximum des écarts avec tous les autres sommets du graphe et est noté $E(x)$.
Le centre d'un graphe quasi-fortement-connexe G est un sommet d'écartement minimal, et on appelle alors rayon de G, noté $\rho(G)$ cet écartement. Le diamètre de G, noté $\delta(G)$ est quant-à-lui l'écartement maximal du graphe.

Exercice

1) Montrer qu'un sommet dont l'écartement est fini est une racine du graphe.
2) Montrer qu'un graphe quasi-fortement-connexe de diamètre fini est fortement connexe.

Solution des exercices

1) Un sommet dont l'écartement est fini possède, par définition de l'écartement, un écart fini avec tout autre sommet. Ceci signifie qu'il est ascendant de tout autre sommet différent de lui-même, et par conséquent qu'il est une racine du graphe.

2) Si un graphe quasi-fortement connexe possède un diamètre fini alors, par définition du diamètre, tout sommet possède un écartement fini. Ainsi, tout sommet du graphe est une racine, ce qui induit qu'il existe un chemin entre toute paire de sommets du graphe ; autrement dit, le graphe est fortement connexe.

Plan

1. Arborescences
   - Définitions
   - Arborescence couvrante
   - Centres et diamètres
2. Représentation des graphes
   - Matrice d'adjacence
   - Liste d'adjacence
   - Avantages et inconvénients de chaque représentation
3. Parcours des graphes
   - Parcours en profondeur d'abord
   - Parcours en largeur d'abord
   - Parcours en meilleur d'abord

Définition dans les graphes non-orientés

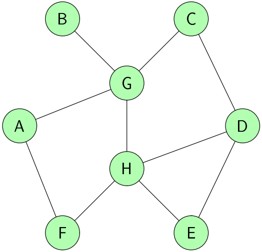
Définition : matrice d'adjacence d'un graphe

Soit $G = (V, E)$ un graphe et Φ une bijection entre V et $\{1, \ldots, n\}$ (i.e. une indexation des sommets représentée par un vecteur de taille n).
La matrice d'adjacence de G est la matrice M de taille n^2 telle que $M_{ij} = 1$ si et seulement si $\Phi(i)$ et $\Phi(j)$ sont voisins dans G.
Si les arêtes de G sont pondérées alors il suffit d'affecter à M_{ij} la valeur de l'arête $(\Phi(i), \Phi(j))$.

Remarque

La matrice d'adjacence d'un graphe non-orienté est symétrique par construction. Il est donc possible de stocker seulement $n(n-1)$ valeurs. Il suffit d'associer à chaque sommet ses voisins de plus grand indice et d'utiliser une fonction d'accès à la matrice qui renvoie M_{ij} si $\Phi(i) < \Phi(j)$ et M_{ji} sinon.

Exemple dans les graphes non-orientés



$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ & & 0 & 0 & 0 & 0 & 1 & 0 \\ & & & 1 & 0 & 0 & 1 & 0 \\ & & & & 1 & 0 & 0 & 1 \\ & & & & & 0 & 0 & 1 \\ & & & & & & 0 & 1 \\ & & & & & & & 1 \\ & & & & & & & \end{pmatrix}$$

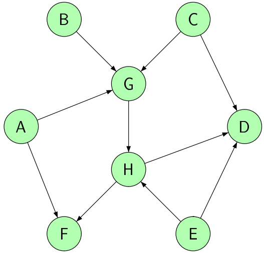
Définition dans les graphes orientés

Dans les graphes orientés la matrice d'adjacence est analogue à celle des graphes non-orientés. Si M est cette matrice alors M_{ij} vaut 1 si et seulement si $(\Phi(i), \Phi(j))$ est un arc du graphe.

Remarque

Comme pour les graphes non-orientés il est possible de ne stocker qu'une moitié de la matrice. Il suffit d'associer à chaque sommet ses voisins de plus grand indice d'affecter 1 à M_{ij} si $(\Phi(i), \Phi(j))$ est un arc, -1 si $(\Phi(j), \Phi(i))$ est un arc et 0 sinon.

Exemple dans les graphes orientés



$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 0 & 1 & 0 \\ & & -1 & 0 & 0 & 1 & 0 \\ & & & 1 & 0 & 0 & -1 \\ & & & & 0 & 0 & -1 \\ & & & & & 0 & 0 \\ & & & & & & 1 \\ & & & & & & \end{pmatrix}$$

Définition des listes d'adjacence

Définition : liste d'adjacence

Soit $G = (V, E)$ un graphe. La représentation par liste d'adjacence de G consiste à associer à chaque sommet du graphe la liste de ses voisins. Il y a donc en mémoire $O(|V| + |E|)$ éléments.

Remarque

Pour les graphes non-orientés chaque arête est stockée deux fois en mémoire, puisque les voisinages sont symétriques. Cependant on ne peut utiliser la symétrie pour stocker moitié moins de valeurs car l'accès n'est pas en $O(1)$!

Avantages et inconvénients de chaque représentation

Espace mémoire nécessaire :

- $O(|V|^2)$ pour la matrice d'adjacence
- $O(|V| + |E|)$ pour les listes d'adjacence

Temps d'accès mémoire :

- $O(1)$ pour la matrice d'adjacence
- $O(|\Gamma(x)|)$ pour les listes d'adjacence

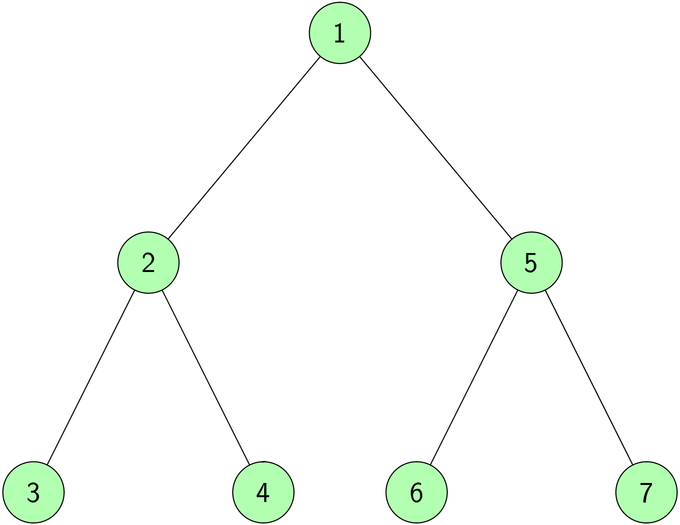
Ainsi on préférera utiliser la représentation par matrice d'adjacence dans des graphes de densité élevée et celle par listes d'adjacence dans des graphes de densité faible (la densité d'un graphe est le quotient du nombre d'arêtes sur le nombre total possible).

Remarque

Il existe également des modèles plus complexes pour tirer parti des avantages des deux représentations (représentation par matrices creuses...).

Plan

Approche intuitive



Définition

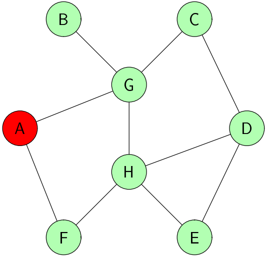
Définition : parcours en profondeur d'abord

Le parcours en profondeur d'abord consiste à parcourir le graphe en développant à chaque fois un fils du noeud courant non encore développé. Si le noeud courant n'a pas de fils non développé on remonte à son père. L'algorithme construit ainsi un arbre couvrant du graphe si le graphe est connexe. Sinon il suffit de l'appliquer à chaque composante connexe.

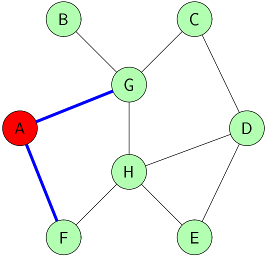
Implantation

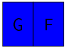
On utilise une pile LIFO (Last In First Out). On empile tous les fils du sommet courant, on dépile le dernier empilé (car la pile est LIFO), et on réapplique le raisonnement. Si le sommet courant n'a pas de fils on dépilera un fils de l'un des ancêtres du sommet courant.

Exemple de parcours en profondeur d'un graphe

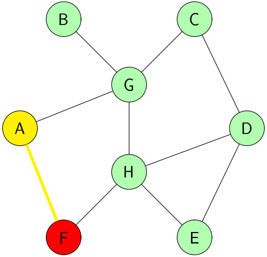


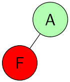
Exemple de parcours en profondeur d'un graphe

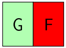




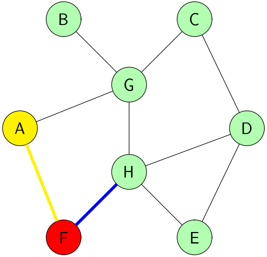
Exemple de parcours en profondeur d'un graphe

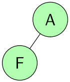


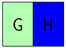




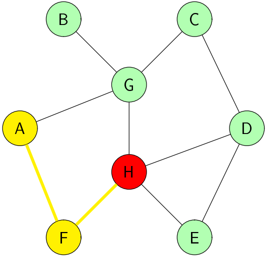
Exemple de parcours en profondeur d'un graphe

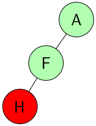


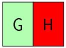




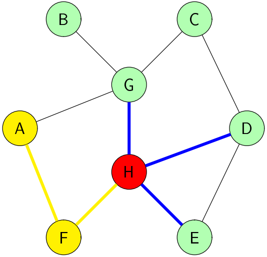
Exemple de parcours en profondeur d'un graphe

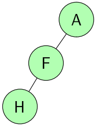


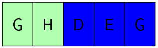




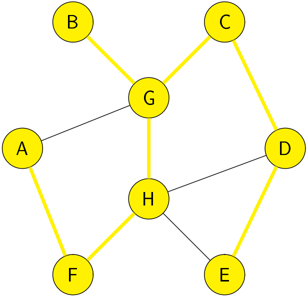
Exemple de parcours en profondeur d'un graphe

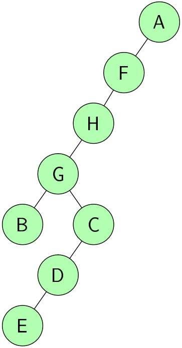




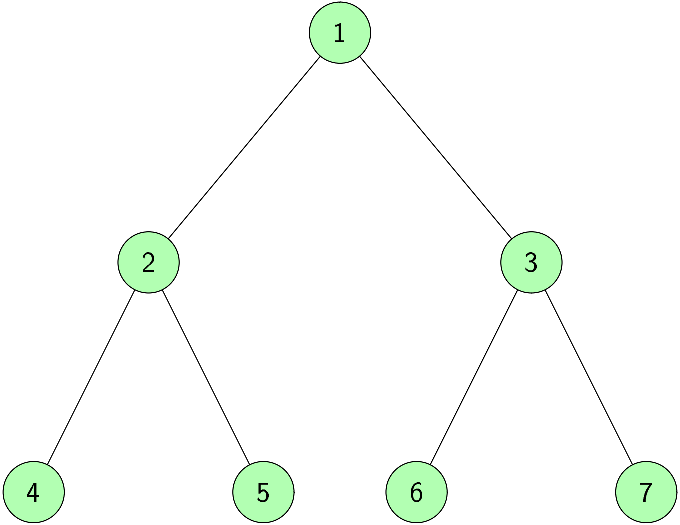


Exemple de parcours en profondeur d'un graphe





Approche intuitive



Définition

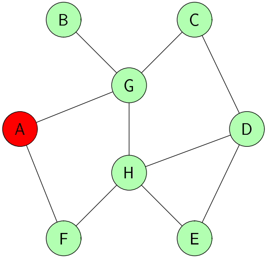
Définition : parcours en largeur d'abord

Le parcours en largeur d'abord consiste à parcourir le graphe en développant chaque noeud fils non encore développé du sommet courant.

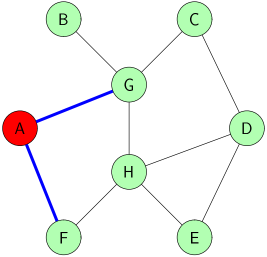
Implantation

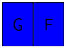
On utilise une pile FIFO (First In First Out). On ajoute pour chaque noeud développé tous ses fils non développés dans la pile, on dépile le sommet empilé le plus ancien et on réapplique le raisonnement.

Exemple de parcours en largeur d'un graphe

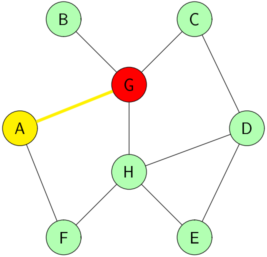


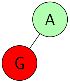
Exemple de parcours en largeur d'un graphe





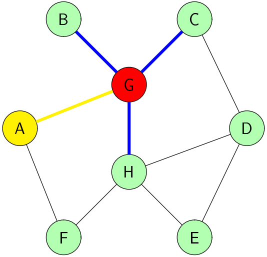
Exemple de parcours en largeur d'un graphe

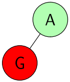




G	F

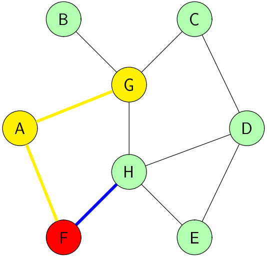
Exemple de parcours en largeur d'un graphe

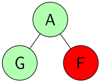




F	B	C	H

Exemple de parcours en largeur d'un graphe

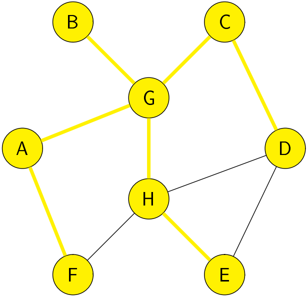


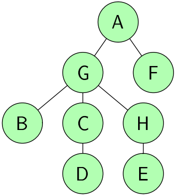


B	C	H

Inutile de mettre H dans la pile !

Exemple de parcours en largeur d'un graphe





Idée générale

Définition : parcours en meilleur d'abord

Le parcours en meilleur d'abord d'un graphe consiste, étant donné un critère de préférence sur les sommets, à développer le sommet le plus prometteur à chaque itération. Le critère le plus souvent utilisé est une borne inférieure de la fonction que l'on veut minimiser (respectivement borne supérieure de la fonction que l'on veut maximiser).

Implantation

L'implantation d'un tel algorithme peut être réalisée grâce à une pile ordonnée selon le critère de préférence. Ainsi, à chaque itération on dépilera le sommet le plus prometteur.