Théorie des graphes (2)

Michel Rigo

http://www.discmath.ulg.ac.be/

Année 2015–2016

Sous-graphe

&

Forte connexité

Sous-graphe

Définition

Soit $G = (V, E)$ un graphe (orienté ou non).
Le graphe $G' = (V', E')$ est un sous-graphe de G si

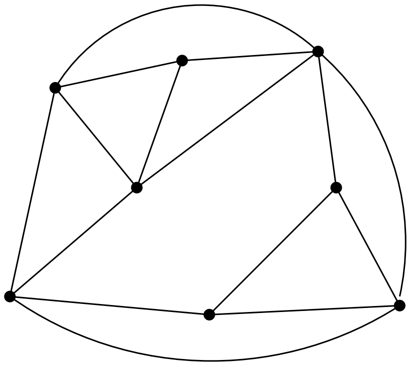
- $V' \subseteq V$,
- $E' \subseteq E \cap (V' \times V')$.

⚠ Si on enlève un sommet v de G,
il faut enlever tous les arcs incidents à v.

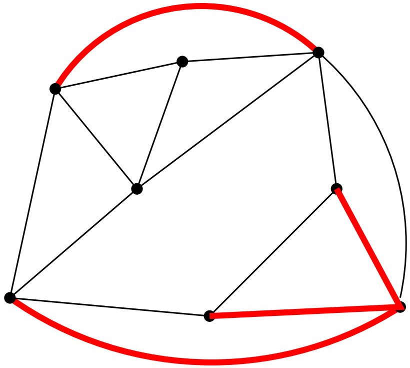
Définition

G' est un sous-graphe propre de G si $E' \subsetneq E$ ou $V' \subsetneq V$.

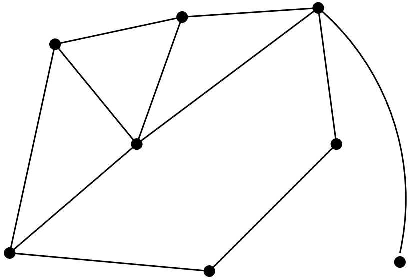
Sous-graphe



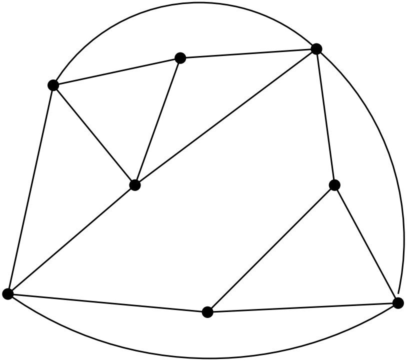
Sous-graphe



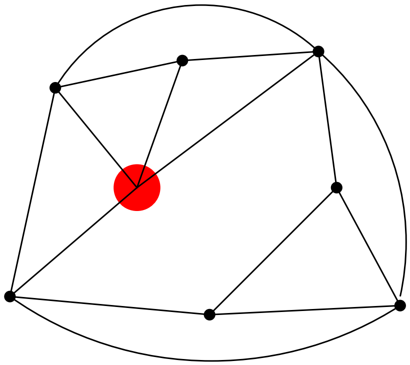
Sous-graphe



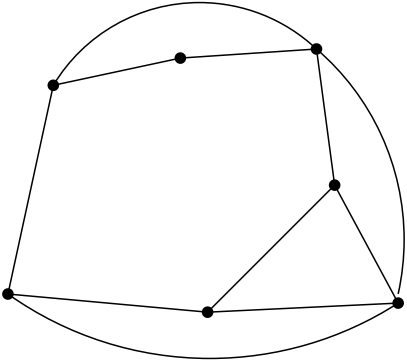
Sous-graphe



Sous-graphe



Sous-graphe



Sous-graphe

Notation

- $G - e$ sous-graphe G' de G obtenu en supprimant l'arc e,
- $G - v$ sous-graphe G' obtenu en supprimant le sommet v et les arcs adjacents,
- $G = G' + e$ graphe obtenu par adjonction à G' d'une arête,
- $G = G' + v$ graphe obtenu par adjonction à G' d'un sommet.

Extension : si $W = \{v_1, \ldots, v_k\} \subseteq V$, alors $G - W$ est le sous-graphe

$$(\cdots((G - v_1) - v_2) \cdots - v_{k-1}) - v_k := G - v_1 - \cdots - v_k.$$

On procède de même pour un ensemble fini d'arcs et on introduit la notation $G - F$ pour $F \subset E$.

Sous-graphe

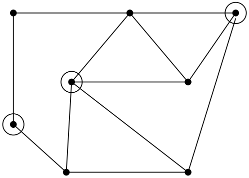
Définition

$W \subseteq V$. Le sous-graphe de G induit par W est le sous-graphe $G' = (W, E')$ avec $E' = E \cap (W \times W)$.

Définition

Si $W \subseteq V$ est tel que le sous-graphe induit par W ne contient aucune arête, alors les sommets de W sont dits indépendants.

$\alpha(G)$ = nombre maximal de sommets indépendants de G



Sous-graphe

Définition

Soient $G = (V, E)$ un graphe et $G' = (V', E')$ un sous-graphe.

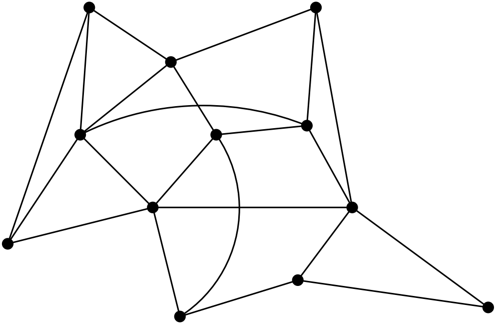
G' est un sous-graphe couvrant G, si $V' = V$ et si

$$\forall v \in V, \exists z \in V : \{z, v\} \in E'.$$

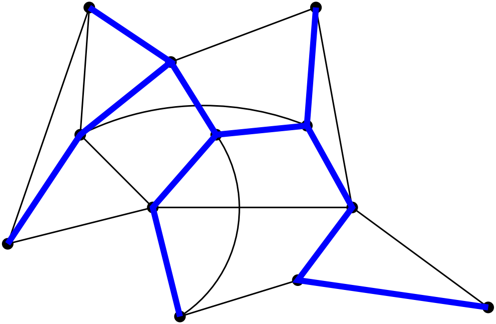
On dira que E' est une couverture (par des arêtes) de G
i.e., tout sommet de G est une extrémité d'une arête de E'.

Notion duale : $W \subset V$ est une couverture (par des sommets) si toute arête a une extrémité dans W.

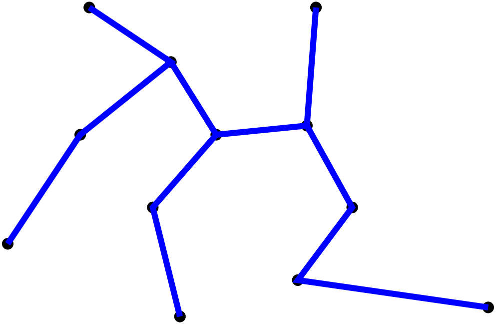
Sous-graphe



Sous-graphe



Sous-graphe



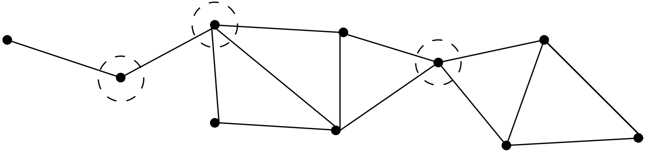
Définition

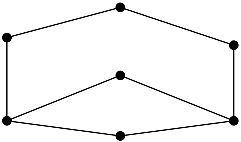
Soit $G = (V, E)$ un multi-graphe non orienté.

v est un point d'articulation, (cut vertex)
si $\#$ comp. connexes de $G - v >$ $\#$ comp. connexes de G.

Si G connexe et n'a aucun point d'articulation,
alors G est au moins 2-connexe (pour les sommets).

Coupes, points d'articulation, k-connexité





Coupes, points d'articulation, k-connexité

Définition

multi-graphe *connexe* $G = (V, E)$
$W \subset V$ ensemble d'articulation de G
si $G - W$ n'est plus connexe ou réduit à un sommet.

$\kappa(G)$= taille minimale d'un ensemble d'articulation de G,
$\kappa(K_n) = n - 1$.

Définition

Pour un multi-graphe G *non connexe*,
W est un ensemble d'articulation
si # comp. connexes de $G - W$ > # comp. connexes de G.

$\kappa(G) = 0$ car non connexe.

TERMINOLOGIE

Si $\kappa(G) = k \geq 1$, H est k-connexe (pour les sommets).

$\kappa(G) = k$:

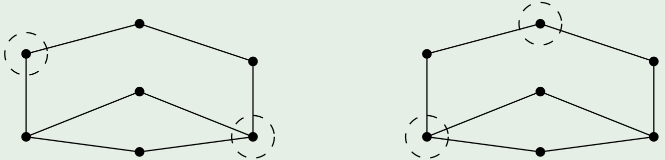
- quels que soient les $k-1$ sommets supprimés, G reste connexe
- il est possible de supprimer k sommets "bien choisis" pour disconnecter G (ou le rendre trivial).

G est au moins k-connexe : $\kappa(G) \geq k$.

REMARQUE

Si G est au moins $(k+1)$-connexe, alors il est au moins k-connexe

Un graphe 2-connexe

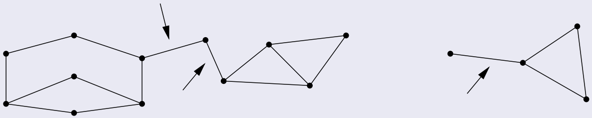


notion analogue pour les arêtes

Définition

Soit $G = (V, E)$ un multi-graphe non orienté.

e est une arête de coupure (bridge, cut-edge, isthmus)
si # comp. connexes de $G - e >$ # comp. connexes de G.



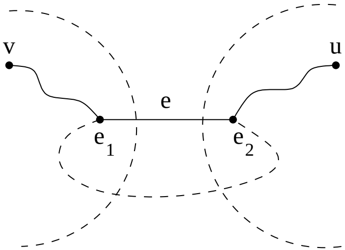
Proposition

Une arête e est une arête de coupure du graphe $H = (V, E)$
SSI e n'appartient à aucune piste fermée de H.

$\Rightarrow$ Si e est une arête de coupure
$\exists$ sommets u et v connectés dans H
mais qui ne sont plus connectés dans $H - e$.

Coupes, points d'articulation, k-connexité

Il existe un chemin joignant u et v qui passe par e.



Dans $H - e$,
une partie de ce chemin joint u à une extremité de e : e_2
l'autre partie du chemin joint v à l'autre extrémité de e : e_1.

P.A. Si e appartient à une piste fermée, il existe un chemin joignant e_1 à e_2 et ne passant pas par e.
u et v sont encore connectés dans $H - e$, impossible !

Réciproque. Supposons que $e = \{e_1, e_2\}$ n'est pas une arête de coupure.

H et $H - e$ ont les mêmes composantes connexes.

$\rightsquigarrow$ les extrémités de e restent connectées dans $H - e$.

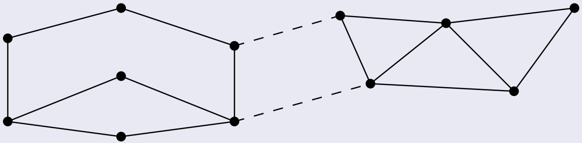
Ainsi, dans H, e appartient à une piste fermée.

Coupes, points d'articulation, k-connexité

Définition

Soit $G = (V, E)$ un multi-graphe non orienté *connexe*.

$F \subset E$ est un ensemble de coupure, une coupe, une coupure
si F est un ensemble **minimal** (pour l'inclusion)
tel que $G - F$ n'est pas connexe.



$\lambda(G)$ = taille minimale d'une coupe

On rencontre aussi la notation $\kappa'(H)$.

Définition

Soit $G = (V, E)$ un multi-graphe non orienté *non connexe*.

$F \subset E$ est un ensemble de coupure, une coupe, une coupure si F est un ensemble **minimal** (pour l'inclusion) tel que $\#$ comp. connexes de $G - F > \#$ comp. connexes de G.

On pose $\lambda(G) = 0$ car G non connexe.

Coupes, points d'articulation, k-connexité

Définition

Si $\lambda(G) = k \geq 1$: H est k-connexe (pour les arêtes).

$\lambda(G) = k$:

- quelles que soient les $k-1$ arêtes supprimées, G reste connexe
- il est possible d'enlever k arêtes "bien choisies" pour le disconnecter.

G est au moins k-connexe (pour les arêtes) : $\lambda(G) \geq k$.

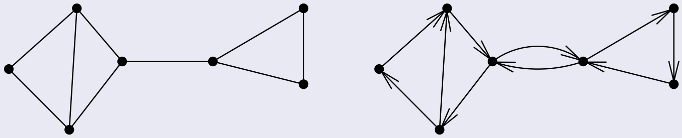
⚠ k-connexité pour les sommets $\neq$ k-connexité pour les arêtes

Problème des sens uniques

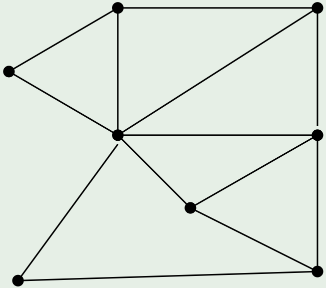
Graphe non orienté connexe. On désire orienter ses arêtes pour obtenir un graphe f. connexe.

Les arêtes de coupure doivent nécessairement être remplacées par deux arcs (pas de sens unique).

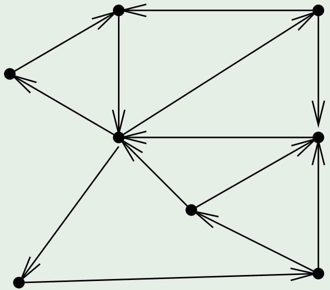
Les autres arêtes appartiennent toutes à une piste fermée qu'il est aisé d'orienter (création de sens uniques).



Un autre exemple



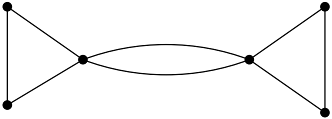
Un autre exemple



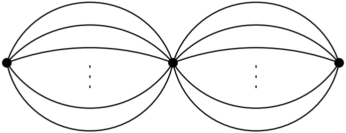
Théorème de Robbins

On peut orienter un graphe connexe pour le rendre f. connexe SSI ce graphe est au moins 2-connexe pour les arêtes.

$\kappa(G) = 1$ et $\lambda(G) = 2$:

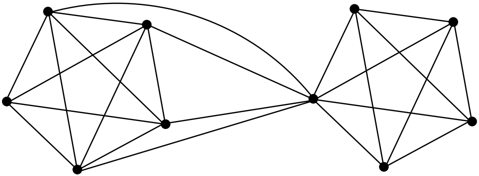


$\kappa(G) = 1$ et $\lambda(G) = k$:



Coupes, points d'articulation, k-connexité

Même avec un graphe simple : pas de lien entre $\lambda(G)$ et $\kappa(G)$.
Ici, $\lambda(G) = 4$ et $\kappa(G) = 1$:



Remarque

Si $\deg v = k$, supprimer les k arêtes incidentes à v isole v

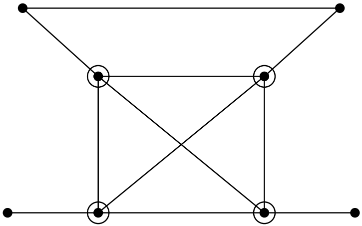
$$\lambda(G) \leq \min_{v \in V} \deg(v).$$

Définition

Une clique d'un graphe non orienté et simple $G = (V, E)$ est un sous-graphe complet de G.

La taille d'une clique est le nombre de sommets qui la composent.

$\omega(G)$ = taille maximale d'une clique de G.



Théorème(s) de Menger

Un autre résultat, pour information, sans preuve.

On considère des graphes simples non orientés
— aucune différence pour des multi-graphes.

Définition

Soient un graphe $G = (V, E)$ et u, v 2 sommets distincts de G.
Un sous-ensemble $S \subseteq V \setminus \{u, v\}$ sépare u et v s'il n'existe aucun chemin joignant u et v dans le sous-graphe de G induit par $V \setminus S$.

Définition

Deux chemins joignant u et v sont indépendants
si les seuls sommets qu'ils ont en commun sont u et v.

Théorème(s) de Menger

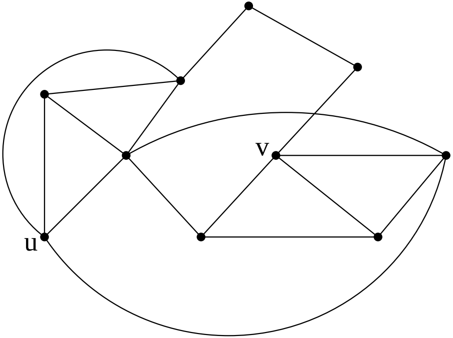
Théorème de Menger (1927)

Soient u, v deux sommets non adjacents d'un graphe connexe.
La taille minimum d'un ensemble de sommets séparant u et v

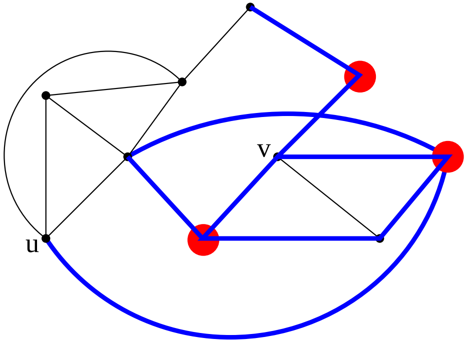
$$=$$

le nombre max. de chemins 2 à 2 indépendants joignant u et v.

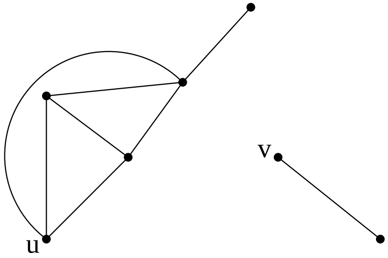
Théorème(s) de Menger

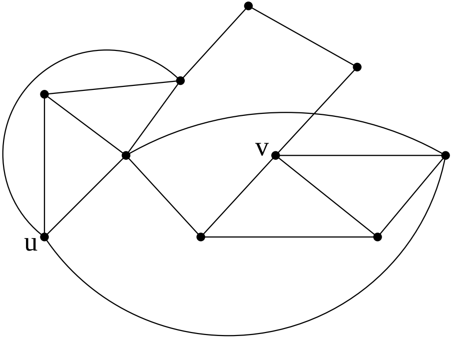


Théorème(s) de Menger

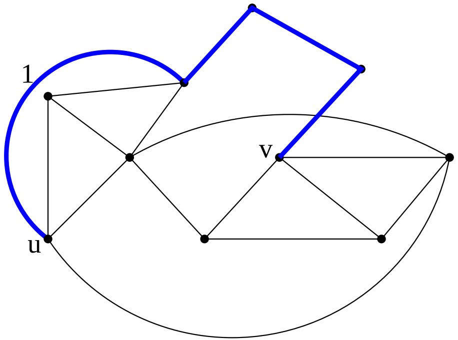


Théorème(s) de Menger

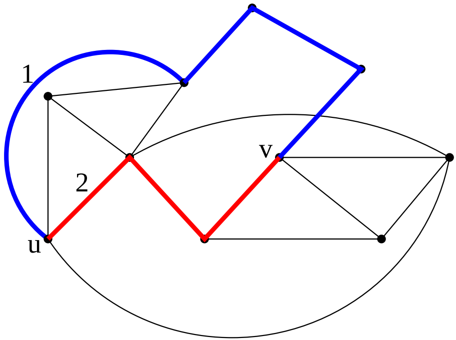



v
u

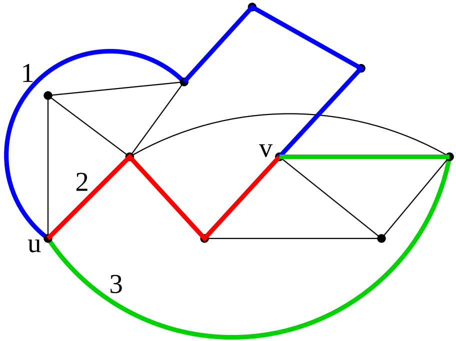
Théorème(s) de Menger



Théorème(s) de Menger



Théorème(s) de Menger



Corollaire, Menger (1927)

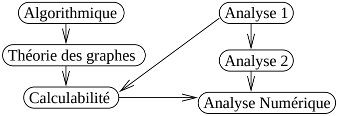
Soit $k \geq 2$.
Un graphe $G = (V, E)$ est au moins k-connexe (pour les sommets)

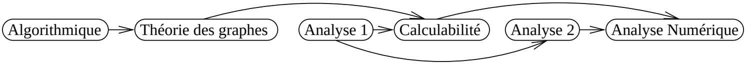
SSI

toute paire de sommets distincts de G est connectée par au moins k chemins indépendants.

Tri topologique

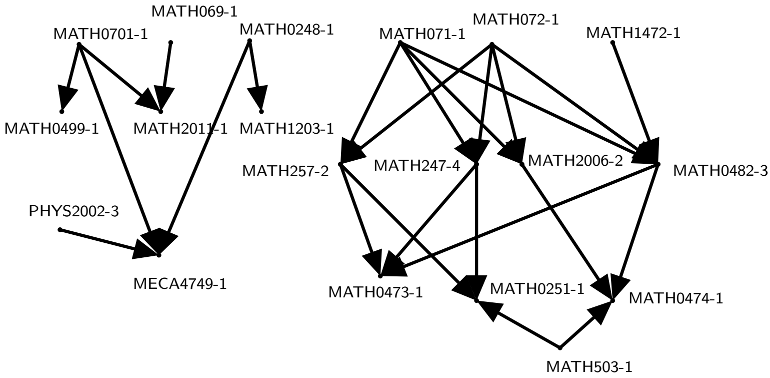
Tri topologique





déterminer une indexation des sommets d'un graphe orienté sans cycle de manière telle que s'il existe un arc de v_i *à* v_j*, alors* $i < j$.

Tri topologique



sous-graphe des prérequis, bachelier sc. math., 2015–2016

Tri topologique

On peut se limiter au cas des graphes simples.

Lemme

Si un graphe simple (fini) orienté $G = (V, E)$ est sans cycle, alors $\exists\ v$ tel que $d^-(v) = 0$ (resp. $d^+(v) = 0$).

Considérer un chemin simple $(x_1, \ldots, x_k)$ de G de longueur maximale déterminé par des sommets de G.

Tri topologique

Proposition

Un graphe simple (fini) orienté $G = (V, E)$ est sans cycle

SSI

$\exists v \in V$ tel que $d^-(v) = 0$ et
$\forall v$ tel que $d^-(v) = 0$, le graphe $G - v$ est sans cycle.

$\Rightarrow$ Découle du lemme précédent.

$\Leftarrow$ Soit v un sommet tel que $d^-(v) = 0$.
Par hypothèse, $G - v$ est sans cycle.

Si G possède un cycle, ce dernier doit passer par v.
Si un cycle passe par v, $d^-(v) \geq 1$. Impossible !

Tri topologique

Algorithme permettant de décider si un graphe est sans cycle.

```
Tant qu'il existe v ∈ V tel que d⁻(v)=0,
G:=G − v
Si G=∅
alors sortie : "oui, G sans cycle"
sinon sortie : "non, G possède un cycle"
```

Complexité

Une implémentation à l'aide de listes d'adjacence,
pour détecter v t.q. $d^-(v) = 0$, on parcourt l'ensemble du graphe.

Un tel parcours est effectué à chaque étape de la boucle.

$\leadsto$ Complexité quadratique en $\#E + \#V$.

Tri topologique

Théorème

Un graphe simple (fini) orienté $G = (V, E)$ est sans cycle SSI il est possible d'énumérer les sommets de $V = \{v_1, \ldots, v_n\}$ t.q. $\forall i = 1, \ldots, n$, le demi-degré entrant de v_i restreint au graphe $G_i = G - v_1 - \cdots - v_{i-1}$ soit nul : $d^-_{G_i}(v_i) = 0$.

$\Rightarrow$ Supposons G sans cycle.
Par le lemme ..., $\exists v_1$ de $G = G_1$ tel que $d^-(v_1) = d^-_{G_1}(v_1) = 0$.
Par la prop..., $G_1 - v_1 = G_2$ est sans cycle.
Par le lemme ..., $\exists v_2$ tel que $d^-_{G_2}(v_2) = 0$.

$\vdots$

De proche en proche, on obtient l'énumération proposée.

Tri topologique

$\Leftarrow$ Supposons disposer d'une énumération des sommets ayant les propriétés indiquées. Procédons par récurrence.

G_n est restreint à l'unique sommet v_n : sans cycle.

G_{n-1} contient les sommets v_n et v_{n-1} et $d^-_{G_{n-1}}(v_{n-1}) = 0$.
G_{n-1} possède au mieux un arc de v_{n-1} à v_n : sans cycle.

Appliquons ce raisonnement pour une étape i quelconque.

Si le graphe G_{i+1} est sans cycle,
alors G_i se compose du graphe G_{i+1} auquel on ajoute v_i
et éventuellement des arcs de v_i vers les sommets de G_{i+1}.
On en conclut que G_i est sans cycle.

Tri topologique

Algorithme basé sur le thm.

La variable d associée à chaque sommet stocke le demi-degré entrant (du graphe envisagé au moment de la construction).

```
Pour tout v ∈ V, initialiser d(v)=0
Pour tout v ∈ V,
   Pour tout w ∈ succ(v), d(w)=d(w)+1
aTraiter := ∅
nbSommet := 0
Pour tout v ∈ V,
   Si d(v) = 0, alors
      aTraiter=aTraiter∪{v}
      nbSommet:=nbSommet+1
```

Tri topologique

```
Tant que aTraiter ≠ ∅, faire
   Soit v, le premier élément de aTraiter
   Enumérer/Print v
   aTraiter := aTraiter \{v}
   Pour tout w ∈ Succ(v), faire
      d(w)=d(w)-1
      si d(w)=0, alors
         aTraiter=aTraiter∪{w}
         nbSommet := nbSommet+1
Si nbSommet=#V
   alors sortie : "oui, G sans cycle"
   sinon sortie : "non, G possède un cycle"
```

Tri topologique

Définition

Soit $G = (V, E)$ un graphe simple orienté.
Un tri topologique de G est une énumération $v_1, \ldots, v_n$ de V t.q. si (v_i, v_j) est un arc de G, alors $i < j$.

Théorème

Un graphe simple orienté admet un tri topologique
SSI il est sans cycle.

Si un graphe possède un cycle, alors quelle que soit l'énumération de ses sommets, il ne peut s'agir d'un tri topologique.

Si un graphe est sans cycle, alors une énumération de ses sommets donnant lieu à un tri topologique est donnée par le thm. précédent.

Remarque

Il n'y a pas qu'un seul tri topologique pour un graphe donné $G = (V, E)$. En effet, si on dénote par

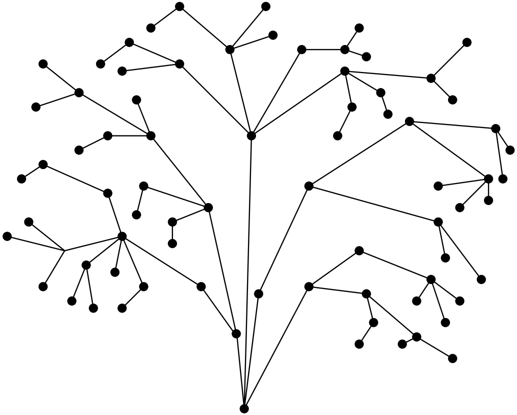
$$S(G) = \{v \in V \mid d^-(v) = 0\}$$

l'ensemble des *sources* de G, alors l'ensemble des tris topologiques de G est donné par la formule récursive suivante

$$\Pi(G) = \bigcup_{v \in S(G)} \{v.\sigma \mid \sigma \in \Pi(G - v)\}$$

où σ est un tri topologique de $G - v$ et où $v.\sigma$ désigne l'énumération des sommets de G en débutant par v puis en suivant l'énumération prescrite par σ.

Arbres



Arbres

Définition

Un arbre est un graphe simple non orienté connexe et sans cycle (i.e., sans circuit simple)

Une forêt est un graphe simple non orienté dont chaque composante connexe est un arbre.

Un arbre $A = (V, E)$ n-aire est t.q. pour tout sommet $v \in V$, $\deg(v) \leq n$.

Arbres

Dans un arbre :

- Toute paire de sommets distincts de G est connectée par exactement un chemin simple.
- Les arêtes d'un arbre sont toutes des arêtes de coupure.
- Soit $e \in (V \times V) \setminus E$ qui n'est pas une boucle. Le graphe $G + e$ contient une piste fermée, c'est-à-dire, $G + e$ n'est plus un arbre.

Réciproquement, un graphe connexe dont toutes les arêtes sont des arêtes de coupure est un arbre.

ARBRES

LEMME

Tout arbre non trivial (non réduit à un sommet) contient un sommet de degré 1.

$\rightsquigarrow$ Si un graphe connexe est tel que tout sommet est de degré ≥ 2, alors ce graphe contient un cycle.

LEMME

Si $A = (G, V)$ est un arbre, alors $\#V = \#E + 1$.

Preuve par récurrence sur $\#V$.
OK pour $\#V = 1$ (graphe trivial) ou $\#V = 2$ (K_2).

Si OK pour n, est-ce OK pour $n + 1$?
A possède un sommet v de degré 1, le graphe $A - v$ est encore un arbre et on applique l'hypothèse de récurrence.

Arbres

Proposition

Un graphe est connexe SSI il possède un sous-arbre couvrant.

$\Leftarrow$ Clair.

$\Rightarrow$ Soient $G = (V, E)$ un graphe connexe et $C = (V, E')$ un sous-graphe couvrant *connexe* minimal (i.e., on ne peut pas remplacer E' par un sous-ensemble strict et garder la connexité).

Vu la minimalité de C, chacune de ses arêtes est une *arête de coupure* de C. $\rightsquigarrow$ C est un arbre.

Corollaire

Si $G = (V, E)$ est un graphe (simple non orienté) connexe, alors $\#E \geq \#V - 1$.

G possède un sous-arbre couvrant $C = (V, E')$. De là, il vient

$$\#E \geq \#E' = \#V - 1$$

SpanningTree(G)

Choose a vertex v ;
$Component \leftarrow \{v\}$; $New \leftarrow \{v\}$; $Edges \leftarrow \emptyset$;
while $New \neq \emptyset$,
    do $Neighbors \leftarrow \emptyset$;
        for all $u \in New$,
            do $Neighbors \leftarrow Neighbors \cup \nu(u)$;
        $New \leftarrow Neighbors \setminus Component$;
        for all $v \in New$,
            do choose an edge $\{u, v\}$ with $u \in Component$;
                $Edges \leftarrow Edges \cup \{\{u, v\}\}$;
        $Component \leftarrow Component \cup New$;
if $Component \neq V(G)$
    then return 'Error : G is not connected' ;
    else return $(Component, Edges)$;

Si on dispose d'un graphe connexe pondérée

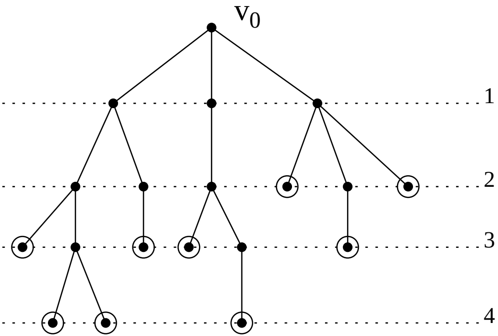
$\rightsquigarrow$ recherche d'un sous-arbre couvrant de poids minimum

- Algorithme de Prim (même philosophie que Dijkstra)
  Jarník–Prim–Dijkstra algorithm
- Algorithme de Kruskal
- ...

DÉFINITION

Un arbre $A = (V, E)$ avec un sommet privilégié v_0 est un arbre pointé : (A, v_0) et v_0 est la racine de l'arbre.

sommets de A ordonnés selon leur distance à v_0.
Si $d(v_0, v) = i$ v est un sommet de niveau i.



Arbres

Définition

Si v est un sommet de niveau i et si tous ses voisins sont de niveau $i-1$, on dit alors que v est une feuille de l'arbre.

La hauteur d'un arbre est le niveau maximal de ses feuilles.

Définition

Pointer un arbre définit *une orientation* des arêtes :
orienter les arcs de façon à ce qu'ils joignent les sommets de niveau i aux sommets de niveau $i+1$.

fils (resp. père) d'un noeud v : ses successeurs
(resp. son unique prédécesseur).

descendants (resp. ancêtres) de v : les éléments de $\mathrm{succ}^*(v)$
(resp. $\mathrm{pred}^*(v)$).

Définition

Un arbre pointé est k-aire si tout sommet a au plus k fils.

Si $k = 2$, on parle d'arbre binaire.

Un arbre k-aire de hauteur n possède au plus

$$1 + k + \cdots + k^n = \frac{k^n - 1}{k - 1}$$

sommets. S'il en possède exactement ce nombre, on parle d'arbre k-aire complet.

Parcours d'arbres

ordonner les noeuds : on suppose que les fils d'un noeud v_i sont ordonnés $v_{i,1}, \ldots, v_{i,k_i}$.
Cet ordre est connu et fixé une fois pour toutes.

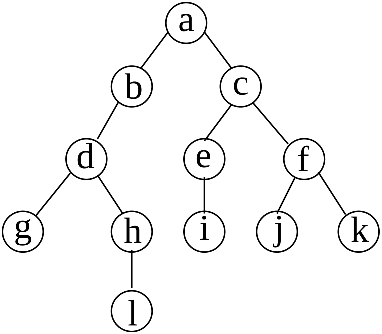
Parcours en profondeur

- parcours préfixe : d'abord la racine puis, de manière récursive, les sous-arbres pointés de racine respective $v_{0,1}, \ldots, v_{0,k_0}$.
- parcours suffixe : d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.
- parcours infixe (arbre binaire) : d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

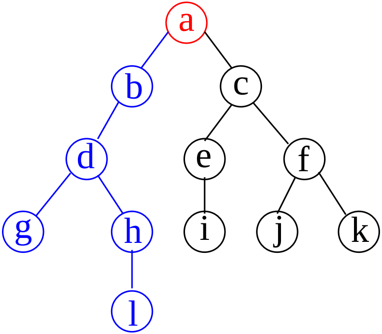


.

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

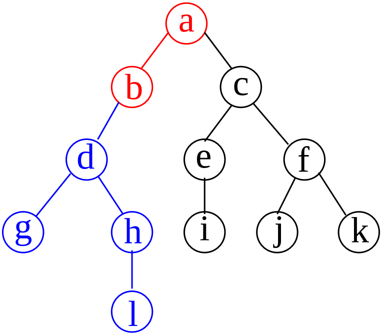


a

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

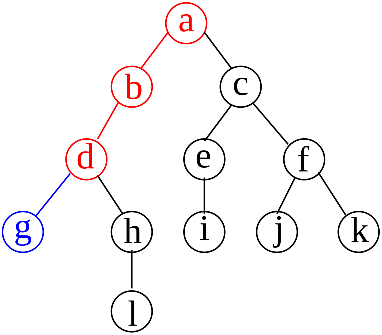


a, b

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

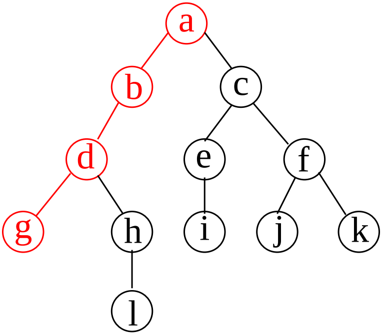


a, b, d

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

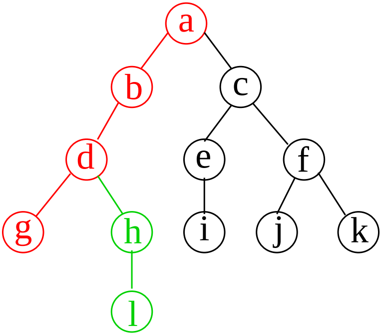


a, b, d, g

Parcours d'arbres

parcours préfixe

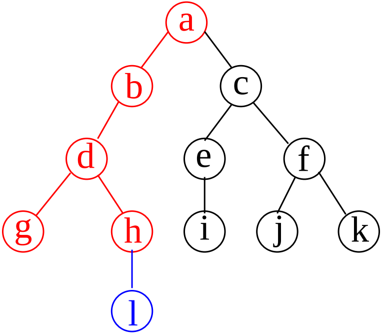
d'abord la racine puis, de manière récursive, les sous-arbres pointés



a, b, d, g

PARCOURS PRÉFIXE

d'abord la racine puis, de manière récursive, les sous-arbres pointés

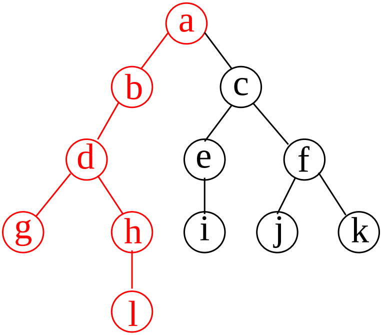


a, b, d, g, h

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

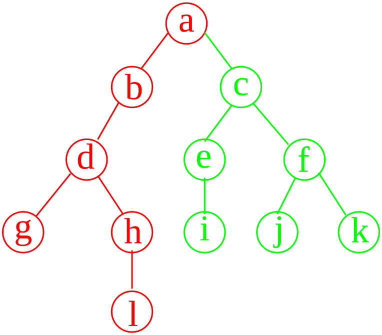


a, b, d, g, h, l

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

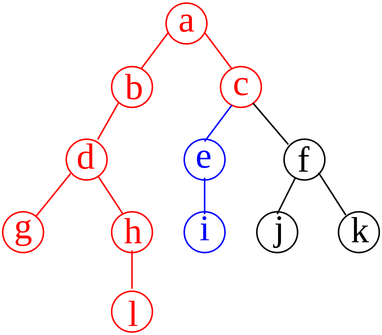


a, b, d, g, h, l

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

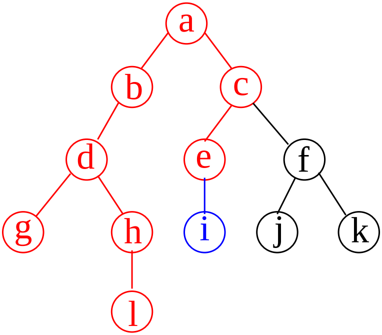


a, b, d, g, h, l, c

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

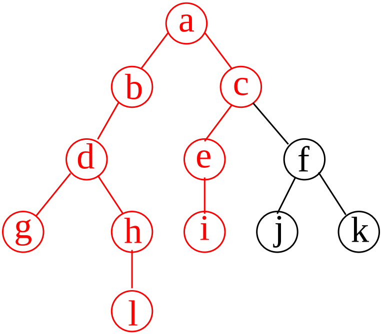


a, b, d, g, h, l, c, e

Parcours d'arbres

parcours préfixe

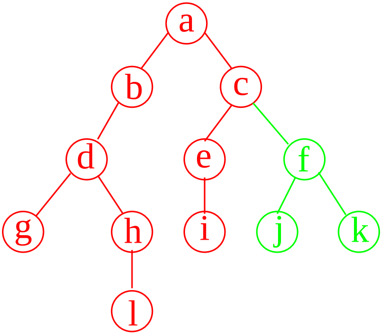
d'abord la racine puis, de manière récursive, les sous-arbres pointés



$a, b, d, g, h, l, c, e, i$

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

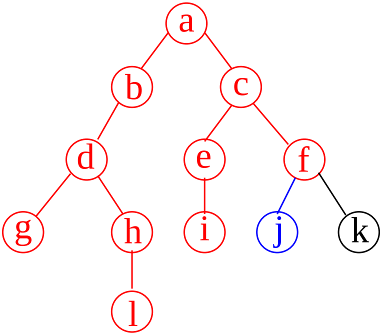


$a, b, d, g, h, l, c, e, i$

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

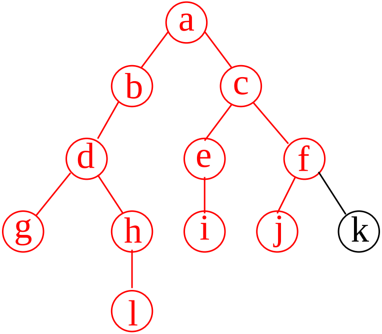


$a, b, d, g, h, l, c, e, i, f$

Parcours d'arbres

parcours préfixe

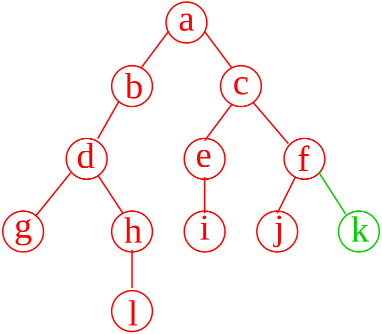
d'abord la racine puis, de manière récursive, les sous-arbres pointés



$a, b, d, g, h, l, c, e, i, f, j$

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

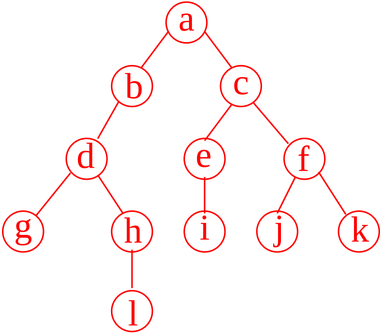


$a, b, d, g, h, l, c, e, i, f, j$

Parcours d'arbres

parcours préfixe

d'abord la racine puis, de manière récursive, les sous-arbres pointés

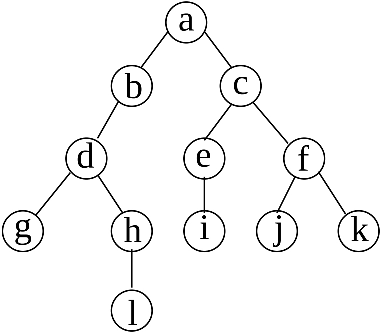


$a, b, d, g, h, l, c, e, i, f, j, k$

Parcours d'arbres

Parcours suffixe

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

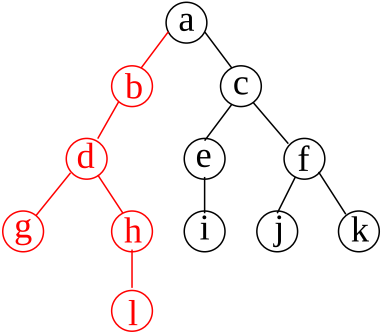


.

Parcours d'arbres

Parcours suffixe

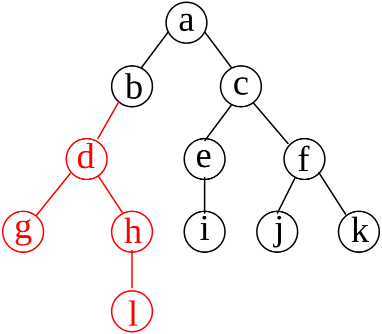
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



.

PARCOURS SUFFIXE

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

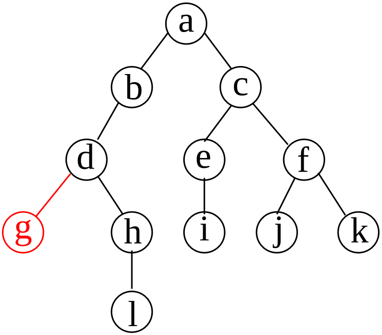


.

Parcours d'arbres

Parcours suffixe

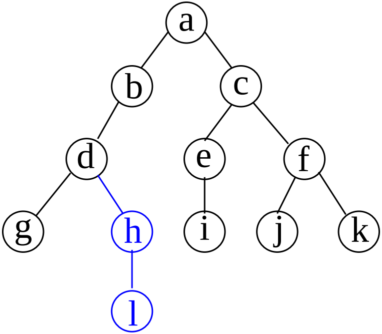
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g

Parcours suffixe

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

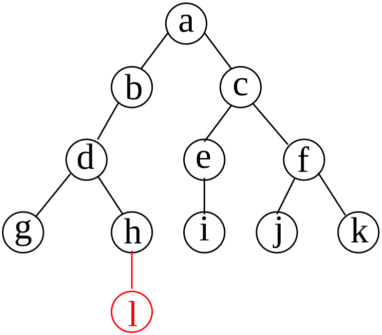


g

Parcours d'arbres

PARCOURS SUFFIXE

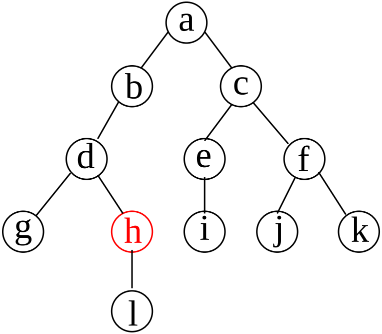
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g, l

Parcours suffixe

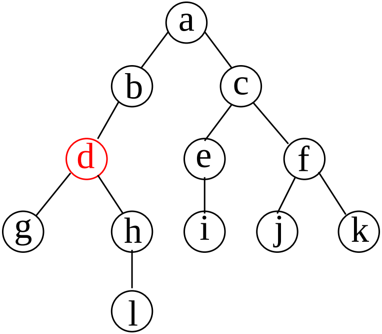
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g, l, h

Parcours suffixe

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

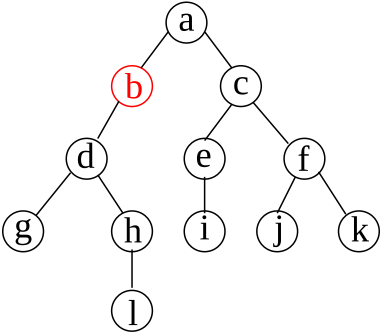


g, l, h, d

Parcours d'arbres

PARCOURS SUFFIXE

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

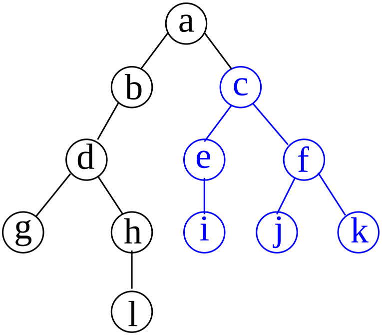


g, l, h, d, b

Parcours d'arbres

PARCOURS SUFFIXE

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

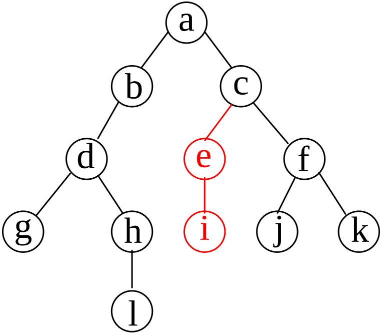


g, l, h, d, b

Parcours d'arbres

PARCOURS SUFFIXE

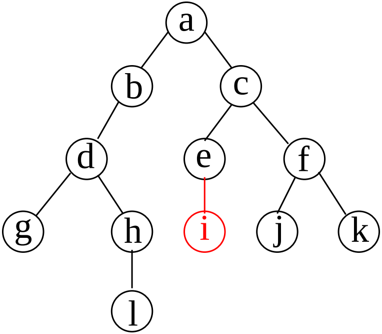
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g, l, h, d, b

Parcours suffixe

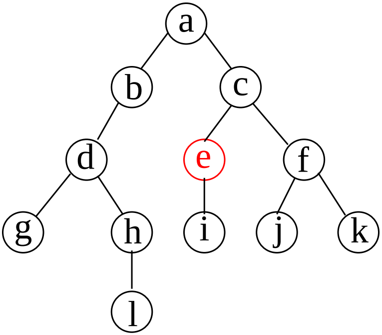
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g, l, h, d, b, i

Parcours suffixe

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

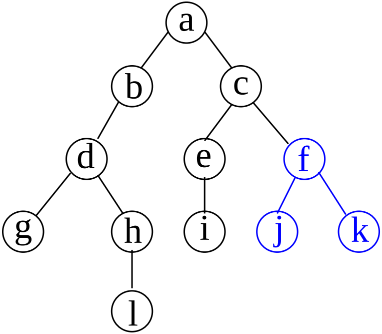


g, l, h, d, b, i, e

Parcours d'arbres

PARCOURS SUFFIXE

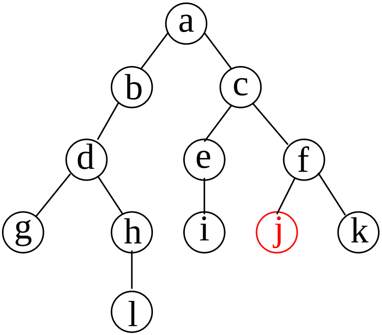
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g, l, h, d, b, i, e

Parcours suffixe

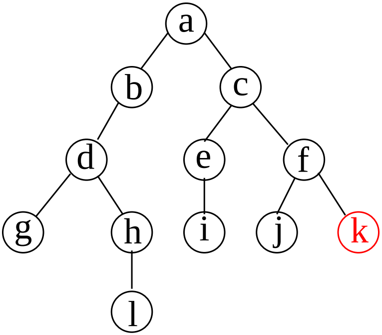
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



g, l, h, d, b, i, e, j

Parcours suffixe

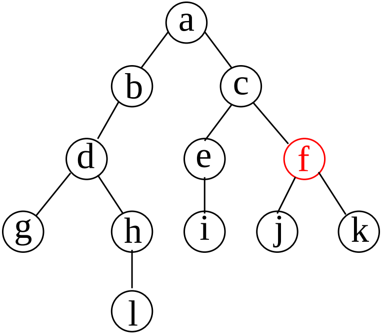
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



$g, l, h, d, b, i, e, j, k$

PARCOURS SUFFIXE

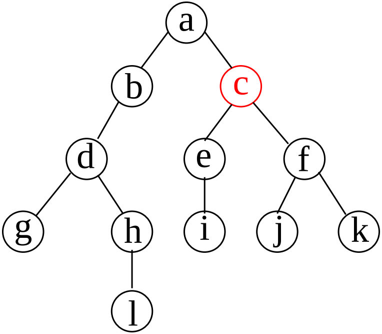
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



$g, l, h, d, b, i, e, j, k, f$

PARCOURS SUFFIXE

d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.

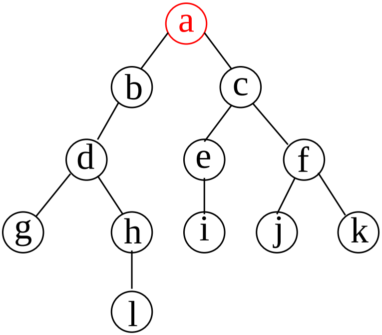


$g, l, h, d, b, i, e, j, k, f, c$

Parcours d'arbres

PARCOURS SUFFIXE

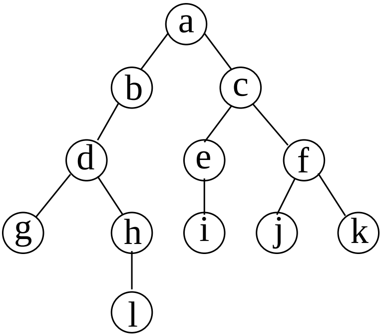
d'abord, de manière récursive, les sous-arbres pointés de racine $v_{0,1}, \ldots, v_{0,k_0}$, puis la racine v_0.



$g, l, h, d, b, i, e, j, k, f, c, a$

Parcours infixe

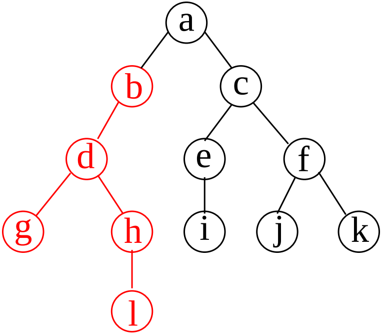
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



.

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

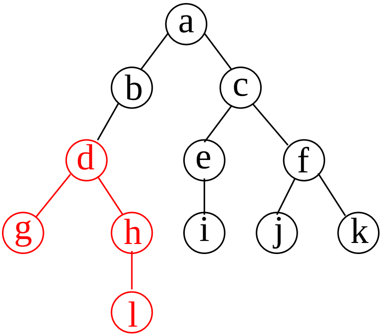


.

Parcours d'arbres

Parcours infixe

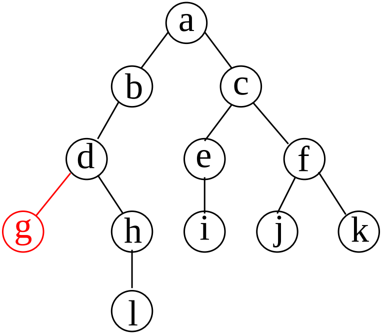
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



.

PARCOURS INFIXE

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

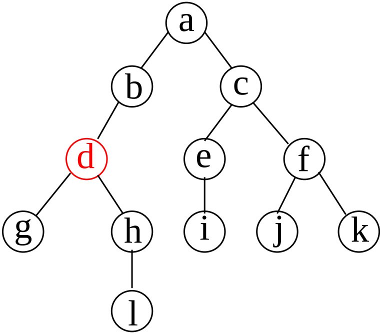


g

Parcours d'arbres

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

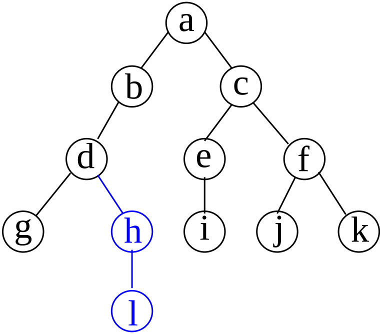


g, d

Parcours d'arbres

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

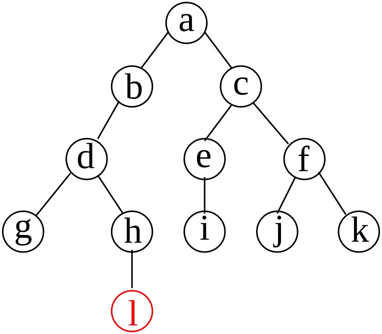


g, d

Parcours d'arbres

PARCOURS INFIXE

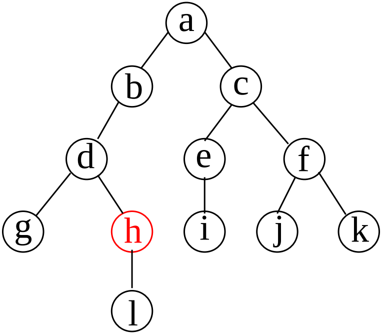
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



g, d, l

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

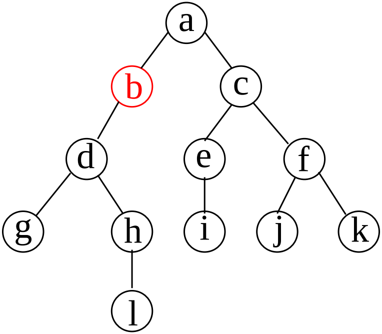


g, d, l, h

Parcours d'arbres

Parcours infixe

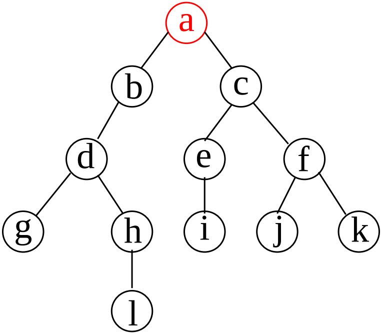
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



g, d, l, h, b

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

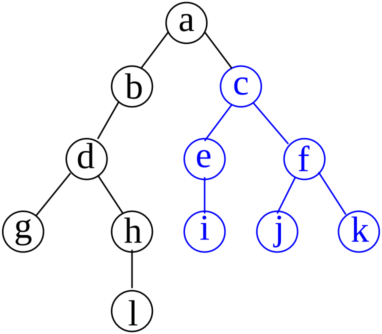


g, d, l, h, b, a

Parcours d'arbres

PARCOURS INFIXE

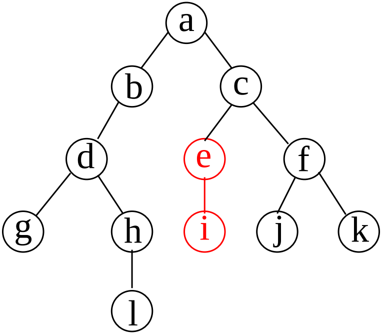
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



g, d, l, h, b, a

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

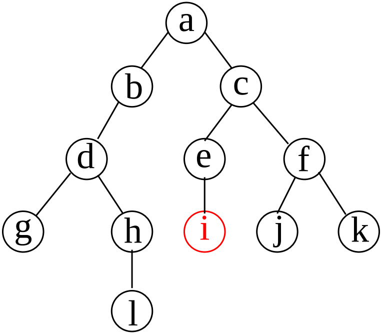


g, d, l, h, b, a

Parcours d'arbres

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

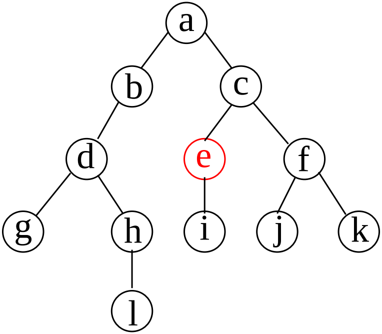


g, d, l, h, b, a, i

Parcours d'arbres

Parcours infixe

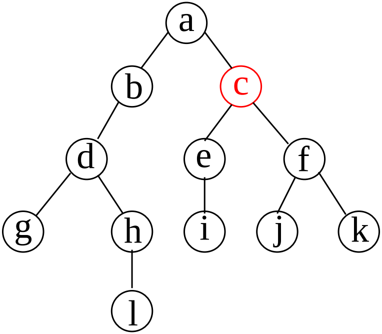
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



g, d, l, h, b, a, i, e

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

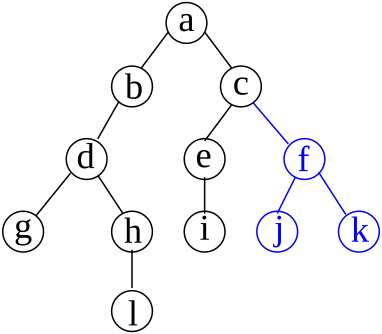


$g, d, l, h, b, a, i, e, c$

Parcours d'arbres

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

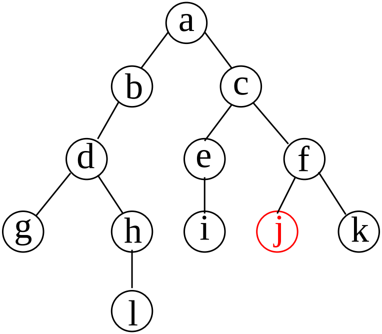


$g, d, l, h, b, a, i, e, c$

Parcours d'arbres

Parcours infixe

d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).

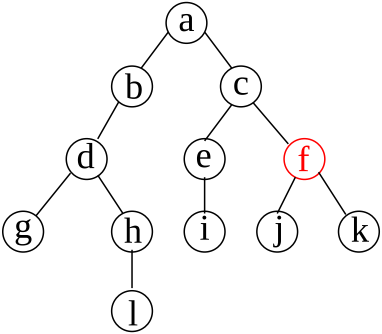


$g, d, l, h, b, a, i, e, c, j$

Parcours d'arbres

Parcours infixe

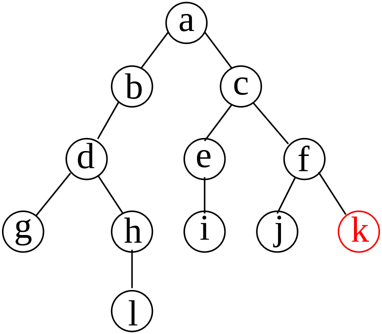
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



$g, d, l, h, b, a, i, e, c, j, f$

Parcours infixe

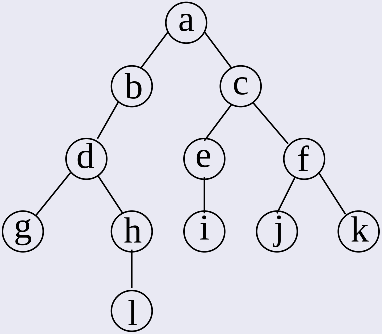
d'abord, de manière récursive, le sous-arbre de gauche, puis la racine, et enfin le sous-arbre de droit (Si un sommet n'a qu'un seul fils : sous-arbre de droite vide).



$g, d, l, h, b, a, i, e, c, j, f, k$

Parcours en largeur

parcours en largeur : parcours des noeuds de l'arbre pointé par niveau croissant.

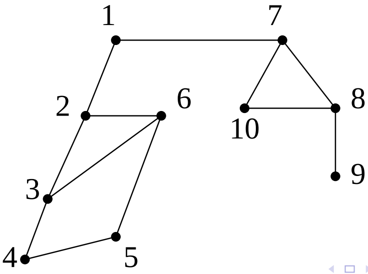


$a, b, c, \ldots, k, l.$

Remarque

Soit $G = (V, E)$ un graphe (orienté ou non) simple et connexe.
Un parcours en profondeur de G est défini récursivement.

Sélectionner un sommet v_0.
A l'étape $k \geq 1$, choisir un voisin de v_{k-1} qui n'a pas encore été sélectionné.
Si un tel voisin n'existe pas, on cherche dans l'ordre, un voisin non sélectionné de $v_{k-2}, \ldots, v_0$.



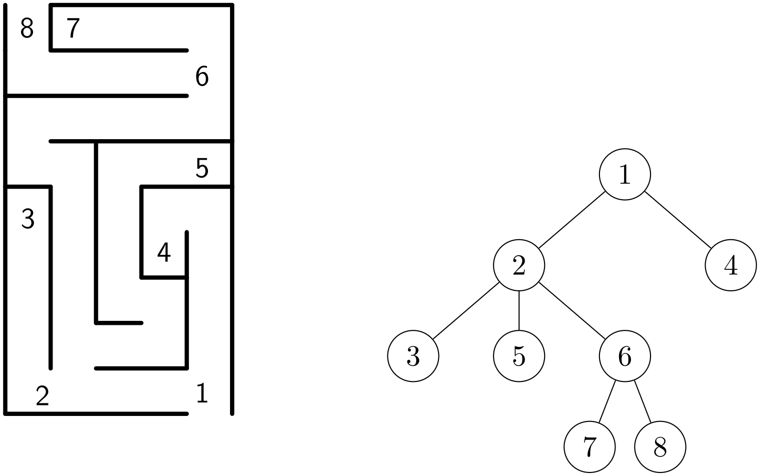


Figure : Un labyrinthe