Alain Cournier

ALGORITHMES DE PARCOURS EN LARGEUR

contenu

- Idées du parcours en largeur
- Premier algorithme
- Rappel sur les graphes non orientés
- Second algorithme
- Un exemple d'exécution

Idées du parcours en largeur

- À partir du sommet de départ x, on visite les sommets depuis les plus proches de x vers les plus éloignés.
- On visitera x puis ses successeurs puis les successeurs des successeurs…

Premier algorithme

- Pour réaliser cette idée de parcours, nous allons partager atteint en 2 parties :
  – AtteintAG : Les sommets découverts auparavant
  – AtteintNG : Les nouveaux sommets découverts
- Au fur et à mesure que l’on choisit des sommets x dans AtteintAG, on insère les sucesseurs de x inexplorés dans AteintNG.

Premier algorithme

- Quand l'ensemble AtteintAG est devenu vide, on y transfèrera tous les sommets de l'ensemble AtteintNG (Nouvelle Génération) afin de continuer la visite du graphe.

Premier algorithme

- Nous allons gérer un entier d (initialement nul) pour maintenir les 2 propriétés suivantes :
- Propriété 1 : Si un sommet y est dans l'ensemble AtteintAG alors le plus court chemin reliant x et y est de longueur d.
- Propriété 2 : Si un sommet y est dans l'ensemble AtteintNG alors le plus court chemin reliant x et y est de longueur d+1.

Algorithme de Visite (En-tête)

- Algorithme VisiteGrapheLarg
  - Données :
    - G = (X,U) un graphe
    - x un sommet de G
  - Donnée/Résultat
    - Exploré : ensemble de sommets
  - Variables
    - AtteintAG, AtteintNG : ensembles de sommets
    - u, v : Sommets de G; d : entier

Algorithme de Base (Code)

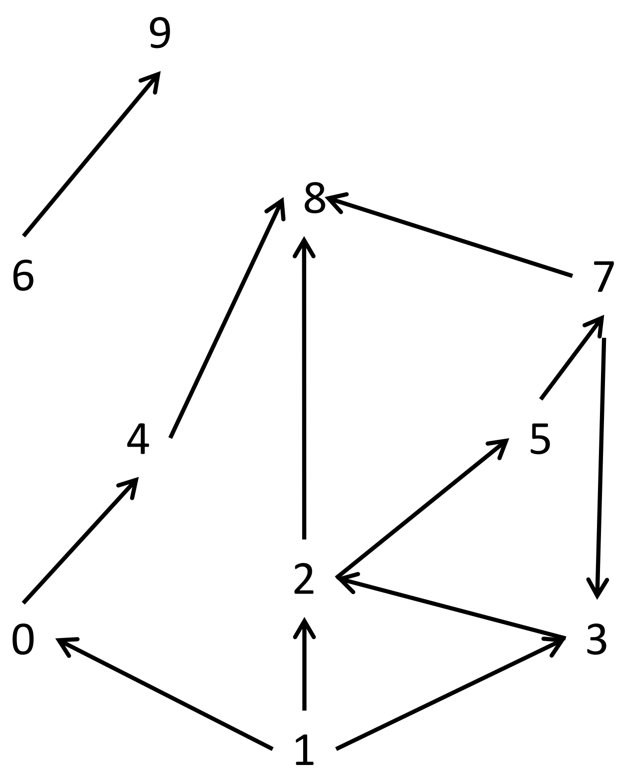
```
DébutCode
    Si x ϵ Exploré alors AtteintAG ← {}
    Sinon AtteintAG ← {x}; AtteintNG ←{}; d←0;
    Finsi
    Tant que AtteintAG ≠ {} faire
        Choisir u ϵ AtteintAG; AtteintAG ← AtteintAG – {u};
        Exploré ← Exploré U {u}; TraiterSommet (u);
        Pour chaque v ϵ Succ(u) faire
            TraiterArc(uv)
            Si non (v ϵ (Exploré U AtteintAG)) alors AtteintNG ← AtteintNG U {v} finsi
        FinPour
        Si AtteintAG ={} alors AtteintAG ← AtteintNG; AtteintNG ←{}; d ← d+1 Finsi
    FinTQ
FinCode
```

Algorithme de parcours

- L'algorithme de ParcoursGraphe reste quasi inchangé.
  - Il suffit remplacer l'appel de l'algorithme VisitGraph par l'appel de VisitGrapheLarg

Exemple

Le graphe

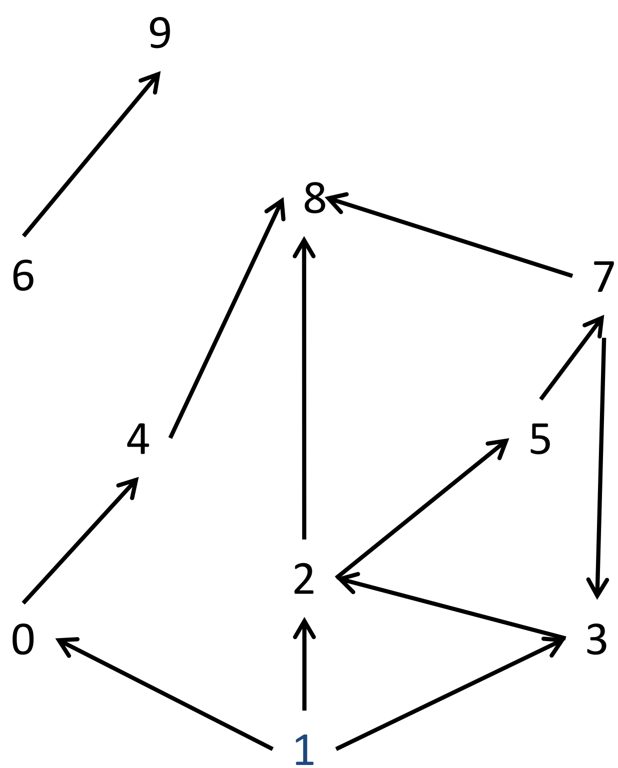


Les explications

- On exécute ParcoursGraphes
- Exploré = {}
- x = 1 (Pour changer)
- Appel de VisitGrapheLarg à partir du sommet x = 1

Exemple

Le graphe

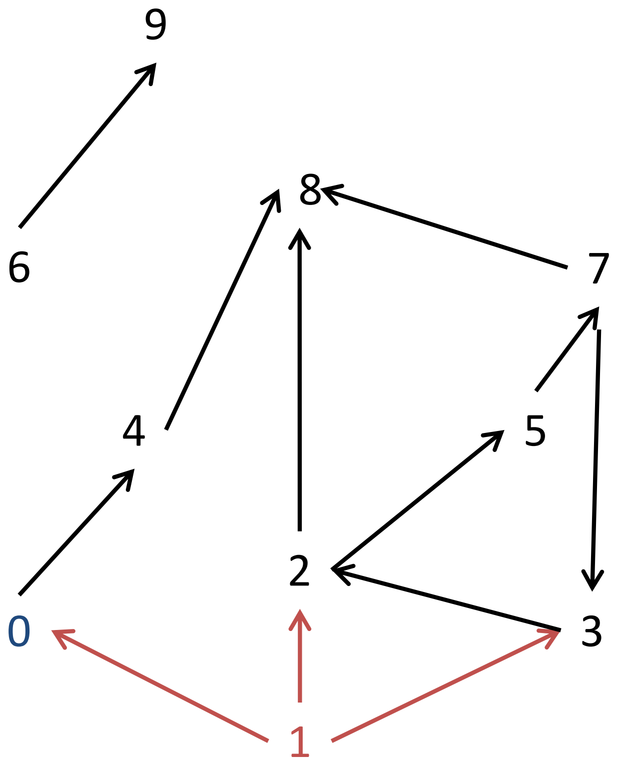


Les explications

- Exploré = {}
- AtteintAG = {1}
- AtteintNG = {}
- On choisi 1,
- Traitement de 1
- Traitement des arcs (1,0), (1,2) et (1, 3)
- Insertion de 0, 2 et 3 dans AtteintNG
- Attention AtteintAG est vide donc AtteintAG ←AtteintNG

Exemple

Le graphe

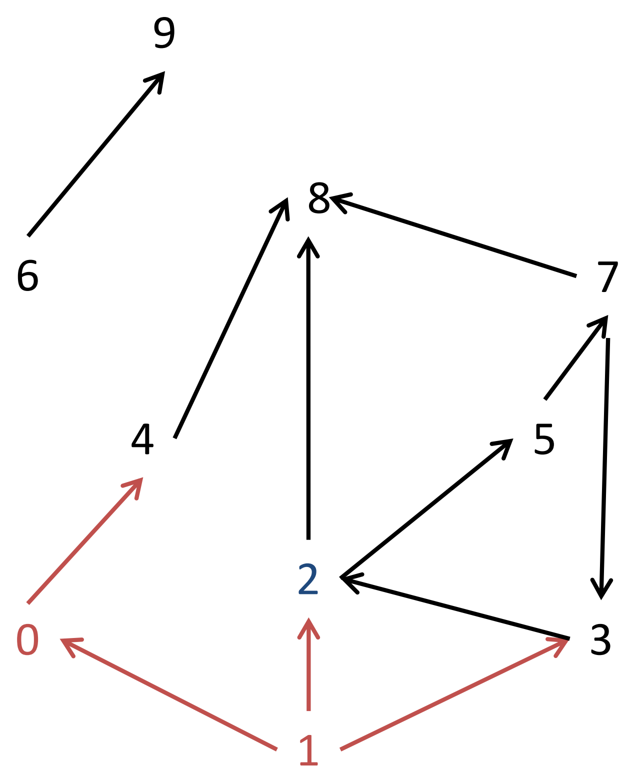


Les explications

- Exploré = {1}
- AtteintAG = {0,2, 3}
- AtteintNG = {}
- On choisi 0,
- Traitement de 0
- Traitement de l'arc (0,4)
- Insérer 4 dans AtteintNG

Exemple

Le graphe

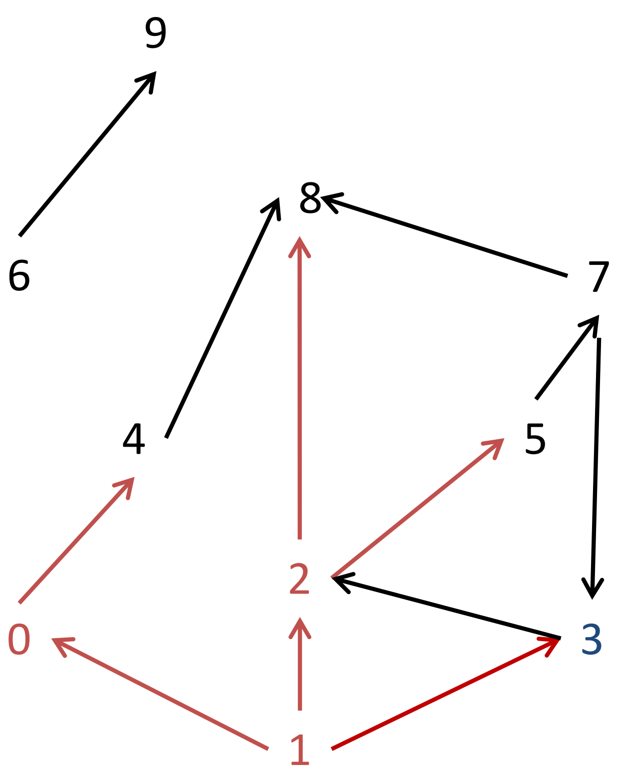


Les explications

- Exploré = {0,1}
- AtteintAG = {2, 3}
- AtteintNG = {4}
- On choisi 2,
- Traitement de 2
- Traitement des arcs (2,8) et (2,5)
- Insérer 5 et 8 dans AtteintNG

Exemple

Le graphe

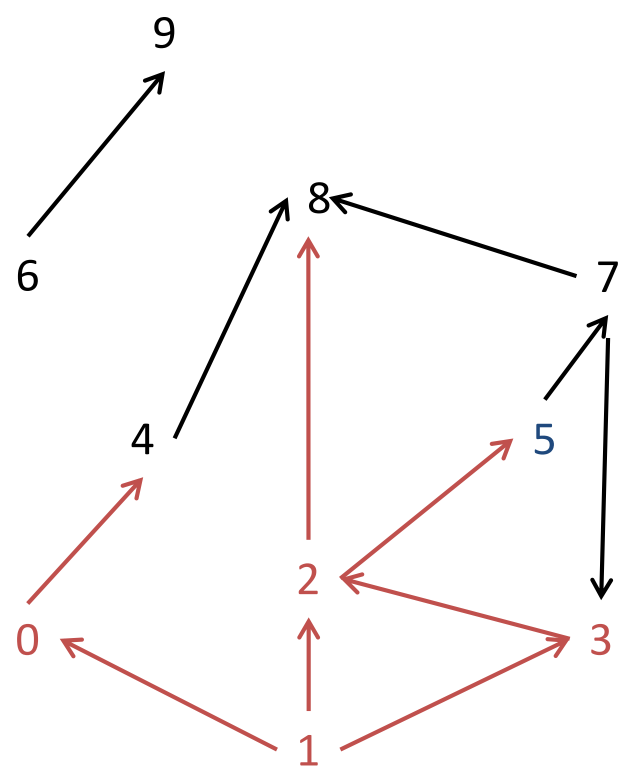


Les explications

- Exploré = {0,1,2}
- AtteintAG = {3}
- AtteintNG = {4,5,8}
- On choisi 3,
- Traitement de 3
- Traitement de l'arcs (3,2)
- Attention AtteintAG est vide donc AtteintAG ←AtteintNG

Exemple

Le graphe

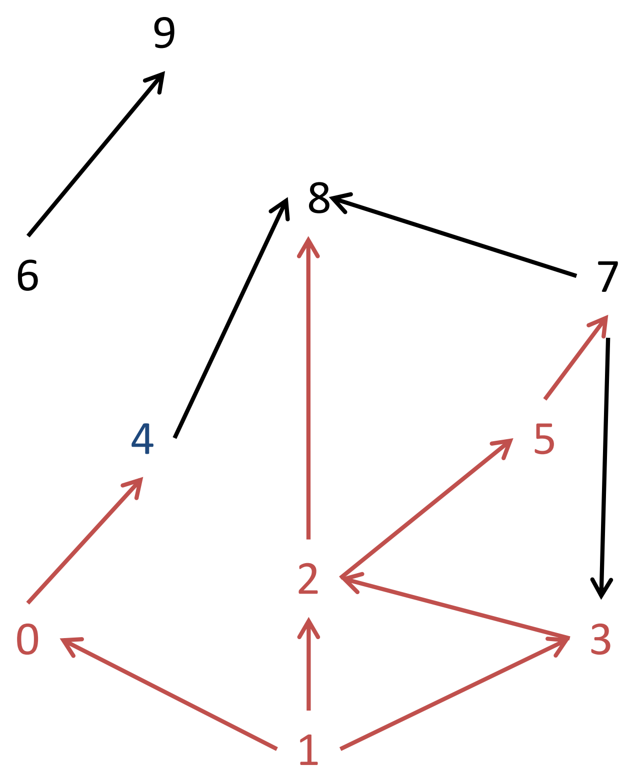


Les explications

- Exploré = {0,1,2,3}
- AtteintAG = {4,5,8}
- AtteintNG = {}
- On choisi 5,
- Traitement de 5
- Traitement de l’arcs (5,7)
- Insérer 7 dans AtteintNG

Exemple

Le graphe

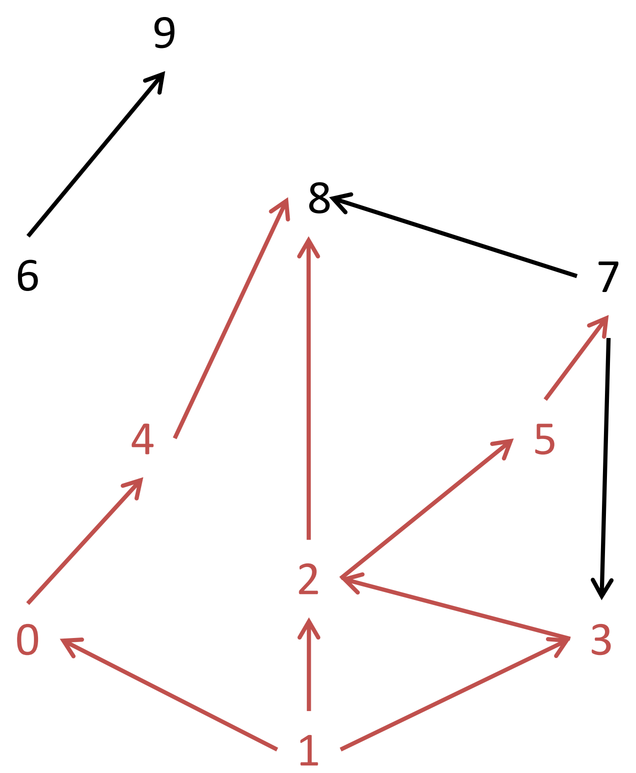


Les explications

- Exploré = {0,1,2,3,5}
- AtteintAG = {4,8}
- AtteintNG = {7}
- On choisi 4,
- Traitement de 4
- Traitement de l'arcs (4,8)

Exemple

Le graphe

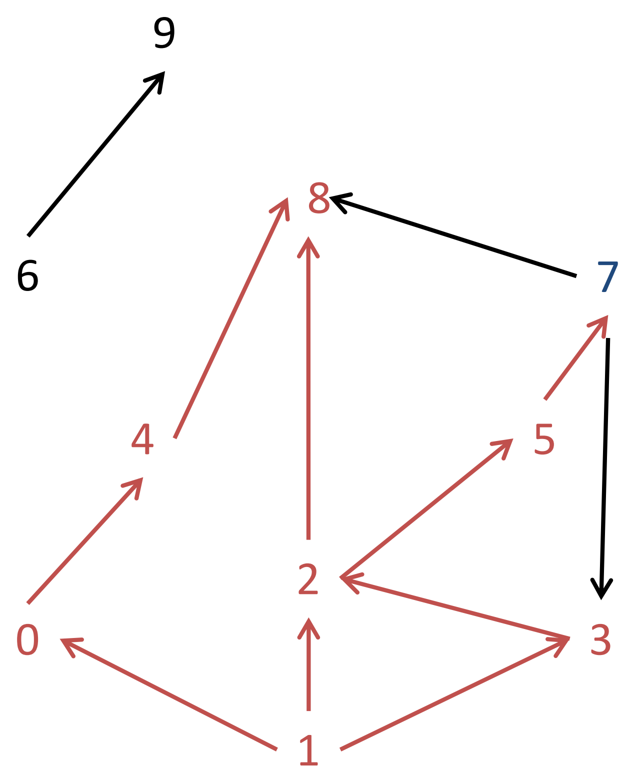


Les explications

- Exploré = {0,1,2,3,4,5}
- AtteintAG = {8}
- AtteintNG = {7}
- On choisi 8,
- Traitement de 8
- AtteintAG est vide

Exemple

Le graphe



Les explications

- Exploré = {0,1,2,3,4,5,8}
- AtteintAG = {7}
- AtteintNG = {}
- On choisi 7,
- Traitement de 7
- Traiter les arcs (7,8) et (7,3)
- AtteintAG et AtteintNG sont vides
- Je vous laisse finir

Parcours sur un graphe non orienté

- Un graphe non orienté est un graphe dans lequel le sens de la relation n'a pas d'importance.
- On ne parle plus alors de l'arc (x,y) mais de l'arête {x,y}.
- L'orientation n'ayant pas d'importance on notera que l'arête {x,y} est égale à l'arête {y,x}.

Représentation

- Un graphe non orienté peut se représenter comme un graphe symétrique.
- Nous ne parlerons plus de successeurs et de s, mais de voisins.

- Rappel : un arbre est un graphe connexe sans cycles.

Parcours en largeur : Algorithme 2

- Un parcours en largeur privilégie l'exploration des sommets du graphe en utilisant les plus courtes chaînes.
- On peut ainsi construire l'arbre des plus courtes chaînes du graphe G à partir d'un sommet x donné.
- On est ainsi capable de calculer la distance entre deux sommets dans un graphe.

Parcours en largeur : Algorithme 2

- Permets en outre de calculer le diamètre et un centre du graphe.

- Nous utiliserons une structure de donnée FIFO pour gérer notre ensemble Atteint : la file. De plus Atteint sera un ensemble de couples (sommet, Distance).

Algorithme ParcLarg (En-tête)

- Algorithme ParcLarg
  - Données :
    - G = (X_G,U_G) un graphe non orienté;
    - x : un sommet de G;
  - Résultats :
    - Exploré : ensemble de couples (sommet, entier)
    - Total : entier
  - Variables :
    - Atteint : une file de couples (sommet; entier)
    - u,v : deux sommets; i : entier

Algorithme ParcLarg (Code)

DebutCode

Atteint ← {(x,0)}; Exploré ← {};

Tq Non(TestFileVide(Atteint)) faire

(u,i) ← Premier(Atteint); Défiler (Atteint);

Insérer (u,i) dans Exploré

Pour tout v $\in V_G(u)$ faire

Si v n’est ni dans Atteint ni dans Exploré alors

Enfiler (v,i+1) dans Atteint Finsi

FinPour

FinTq

CalculTotal (Exploré,Total)

FinCode

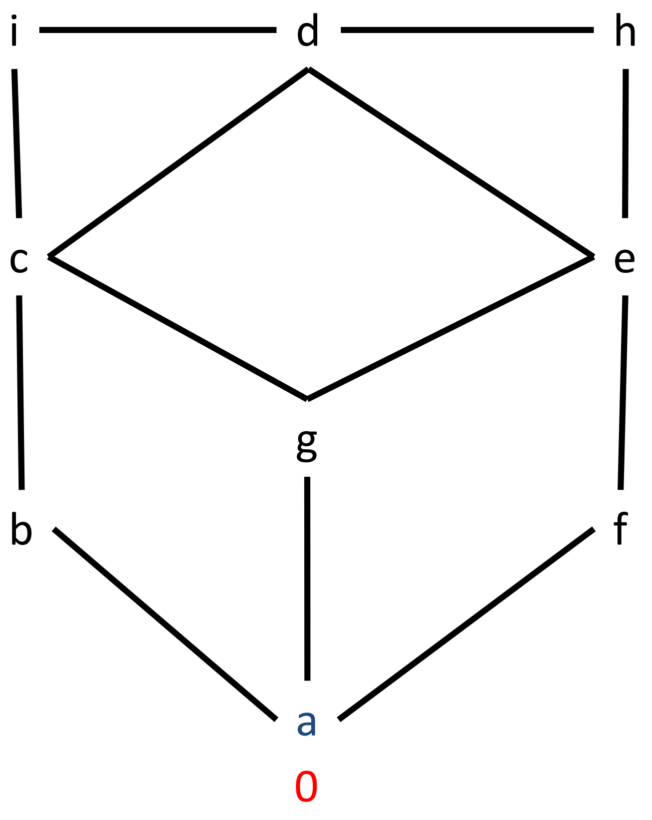
Algorithme CalculTotal (En-tête)

- Algorithme CalculTotal
  - Donnée
    - Exploré : ensemble de couples (sommets, entier)
  - Résultat
    - Total : entier
  - Variables
    - u : sommet; i entier

Algorithme CalculTotal (Code)

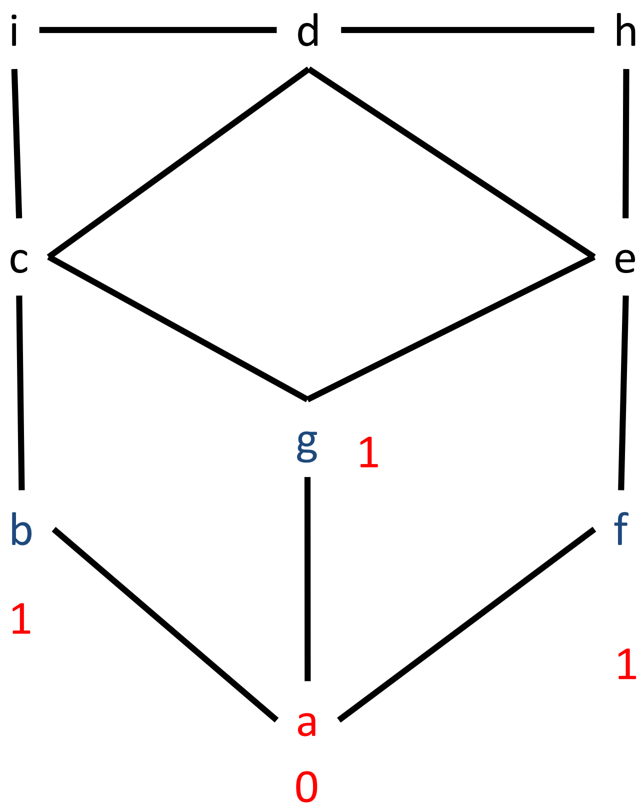
- DébutCode
  - Total ← 0;
  - Pour chaque couple (u,i) de Exploré faire
    - Total ← Total + i;
  - FinPour
  - Renvoyer(Total)
- FinCode

Exemple



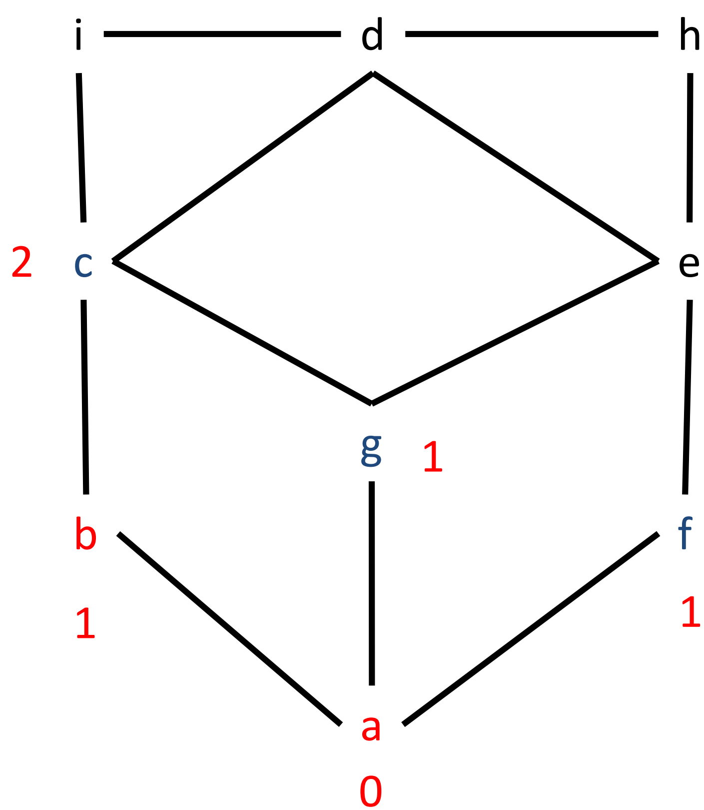
- Appel de ParcLarg à partir de a.
- Les valeurs entières données par l'algorithme seront écrites en Rouge
- Atteint = {(a,0)}

Exemple



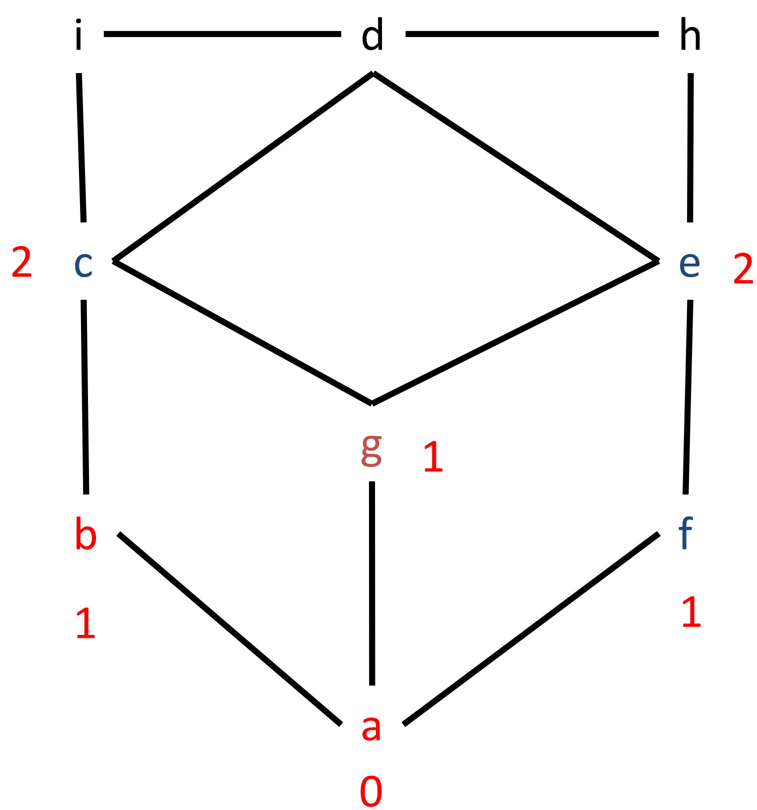
- On prend le couple (a,0)
- Insertion de (a,0) dans Exploré
- Exploré = {(a,0)}
- Chaque voisin b, g, f est introduit dans atteint
- Atteint={(b,1),(g,1),(f,1)}

Exemple



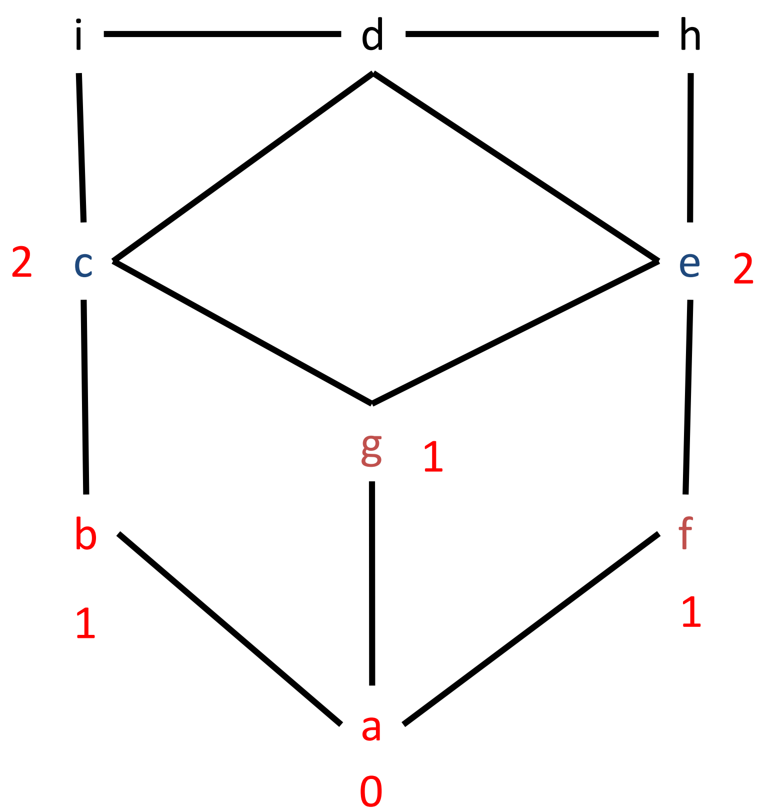
- On prend le couple (b,1)
- Insertion de (b,1) dans Exploré
- Exploré = {(a,0),(b,1)}
- Le voisin c est introduit dans Atteint
- Atteint={(g,1),(f,1),(c,2)}

Exemple



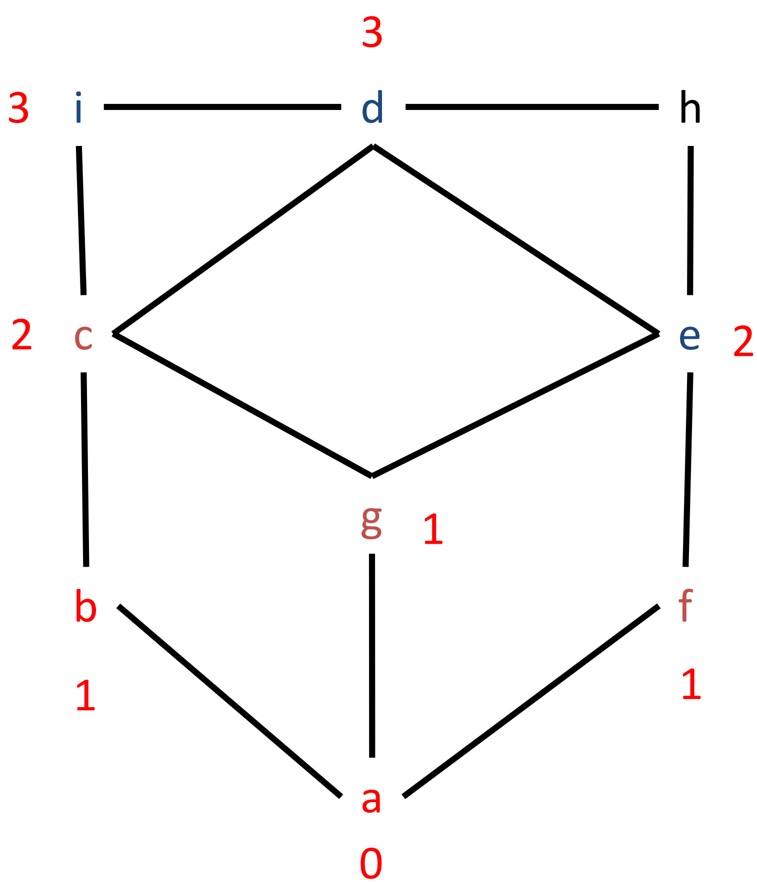
- On prend le couple (g,1)
- Insertion de (g,1) dans Exploré
- Exploré = {(a,0),(b,1),(g,1)}
- Le voisin e est introduit dans Atteint
- Atteint={(f,1),(c,2),(e,2)}

Exemple



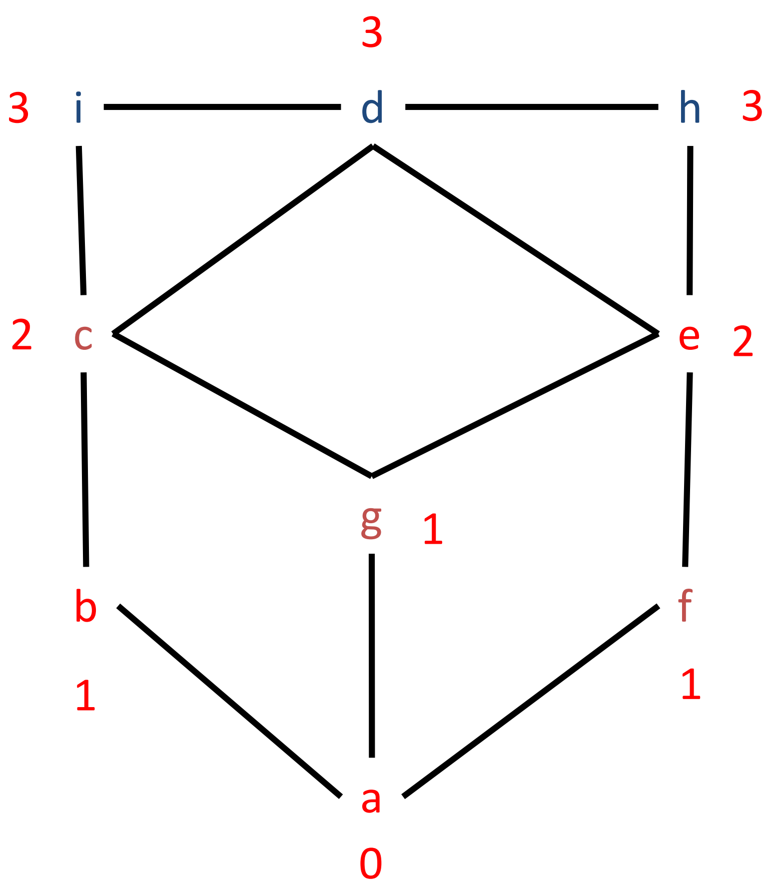
- On prend le couple (f,1)
- Insertion de (f,1) dans Exploré
- Exploré = {(a,0),(b,1),(g,1),(f,1)}
- Aucun sommet n'est introduit dans Atteint
- Atteint={(c,2),(e,2)}

Exemple



- On prend le couple (c,2)
- Insertion de (c,2) dans Exploré
- Exploré = {(a,0),(b,1),(g,1),(f,1),(c,2)}
- Les Voisins i et d sont introduits dans Atteint
- Atteint={(e,2),(i,3),(d,3)}

Exemple



- On prend le couple (e,2)
- Insertion de (e,2) dans Exploré
- Exploré = {(a,0),(b,1),(g,1),(f,1),(c,2),(e,2)}
- Le voisins h est introduit dans Atteint
- Atteint={(i,3),(d,3),(h,3)}
- … Total 16

Remarques

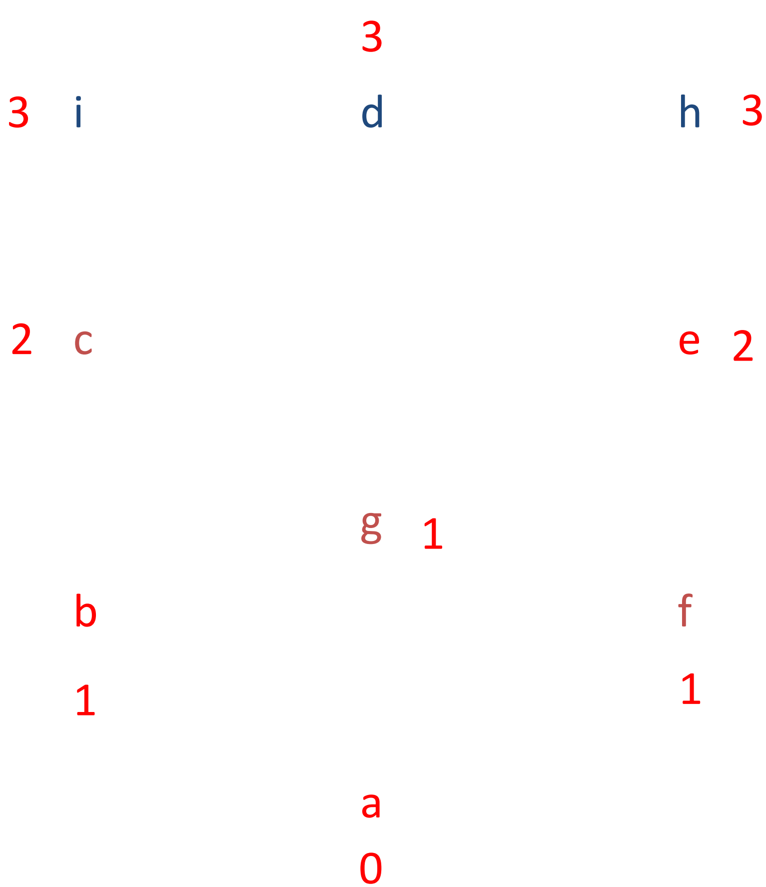
- Si un sommet u a reçu la valeur entière val alors :
  - il existe un chaîne de longueur val entre le point de départ du parcours (a dans l'exemple) et notre sommet u.
  - Il n'existe pas dans le graphe de chaîne de longueur strictement inférieure à val entre a et notre sommet u.
- Le parcours en largeur permet de calculer la plus courte chaîne entre deux sommets.

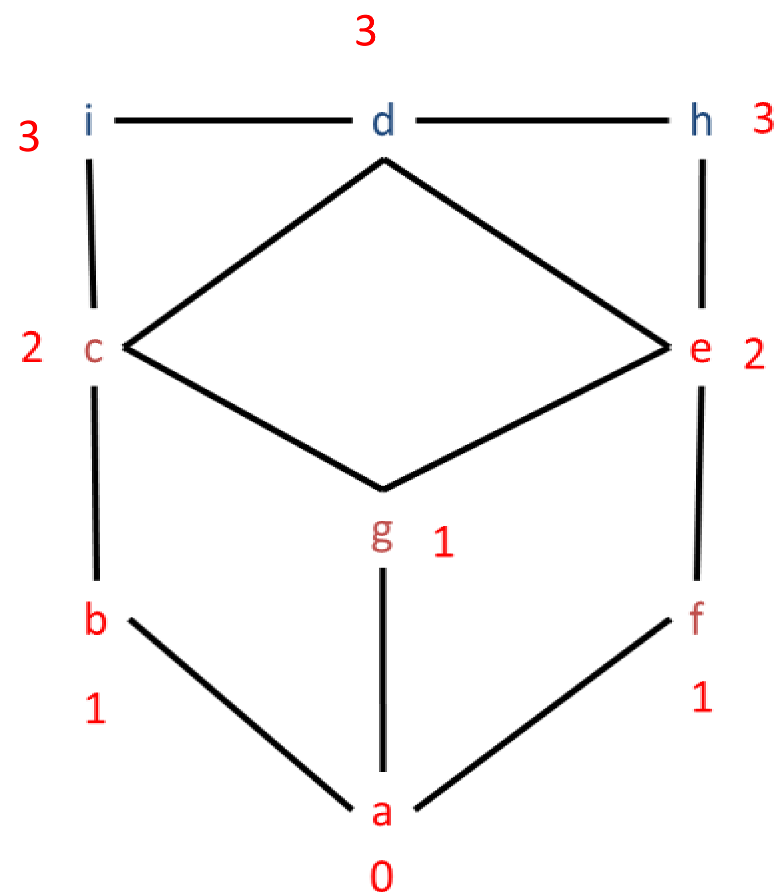
Remarques

- La longueur d’une chaîne est égale au nombre d’arêtes qui la compose. C’est aussi le nombre de sommets de la chaîne moins un.

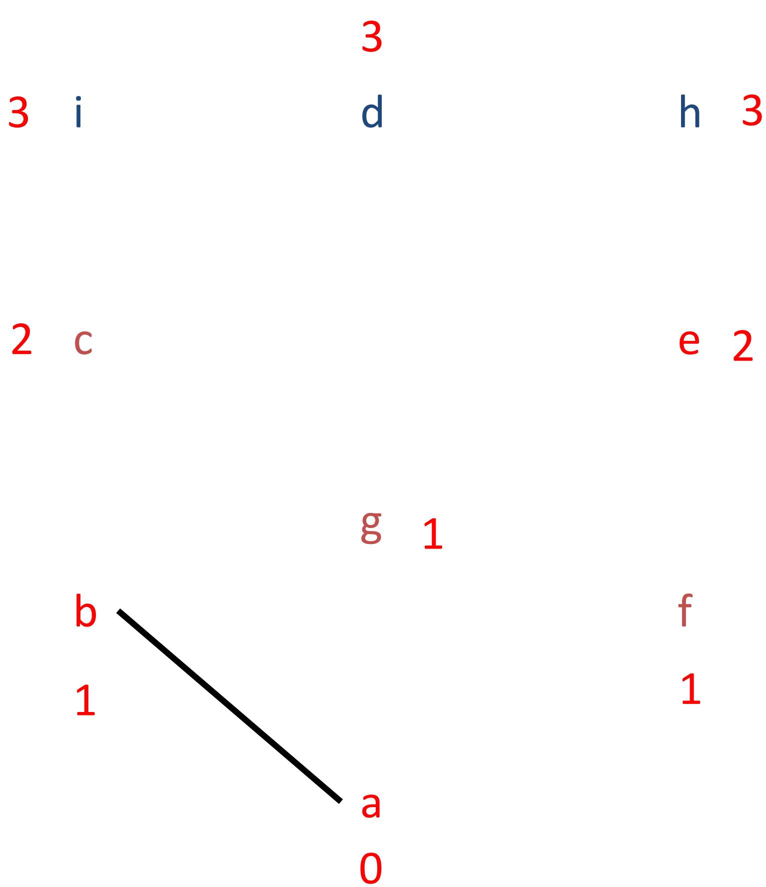
- La distance entre deux sommets x et y dans un graphe G est égale à la longueur de la plus courte chaîne liant x et y. On la note $d_G(x,y)$

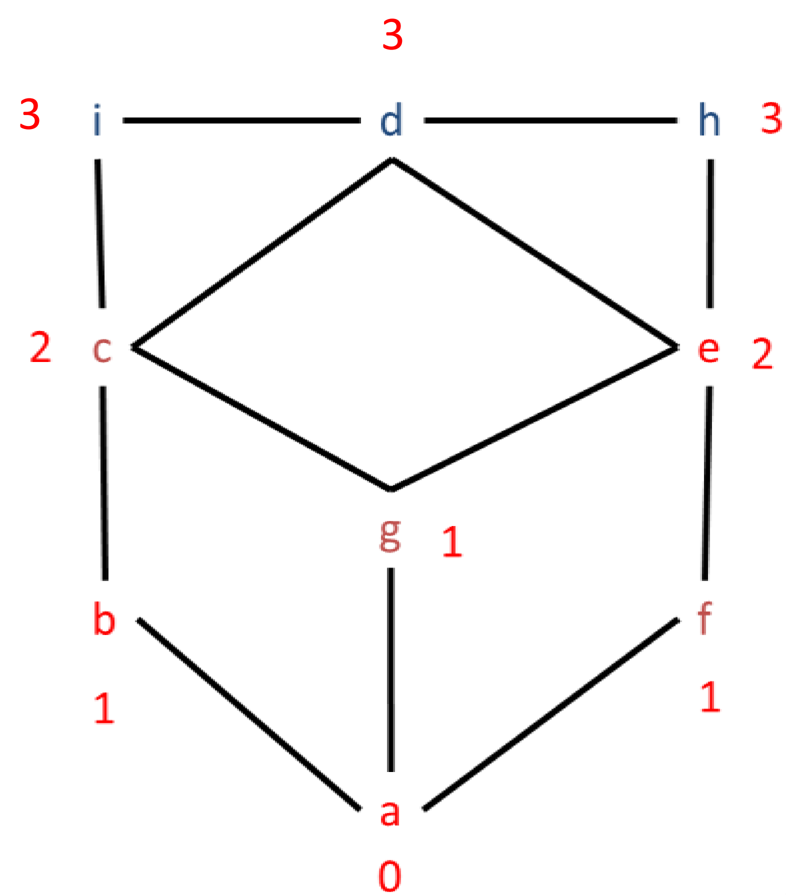
Exemple



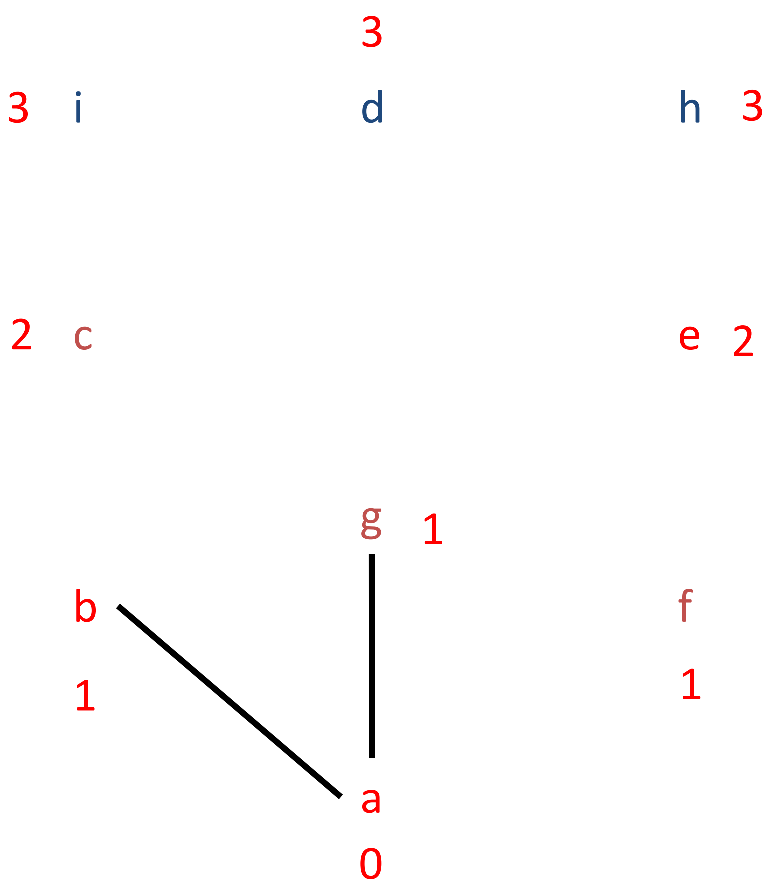


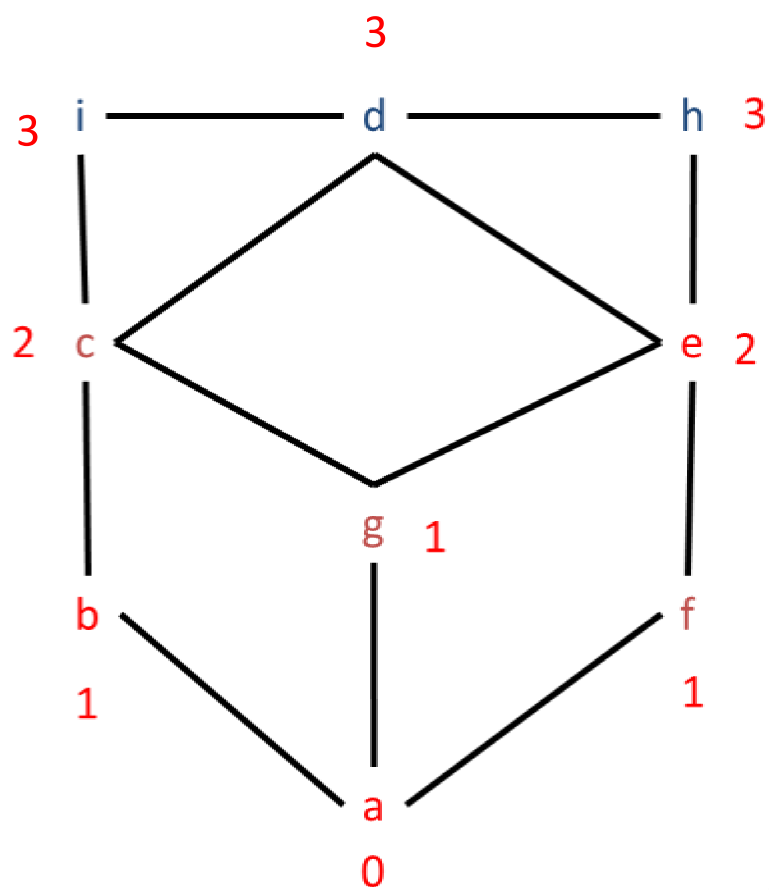
Exemple : Construire l’arbre



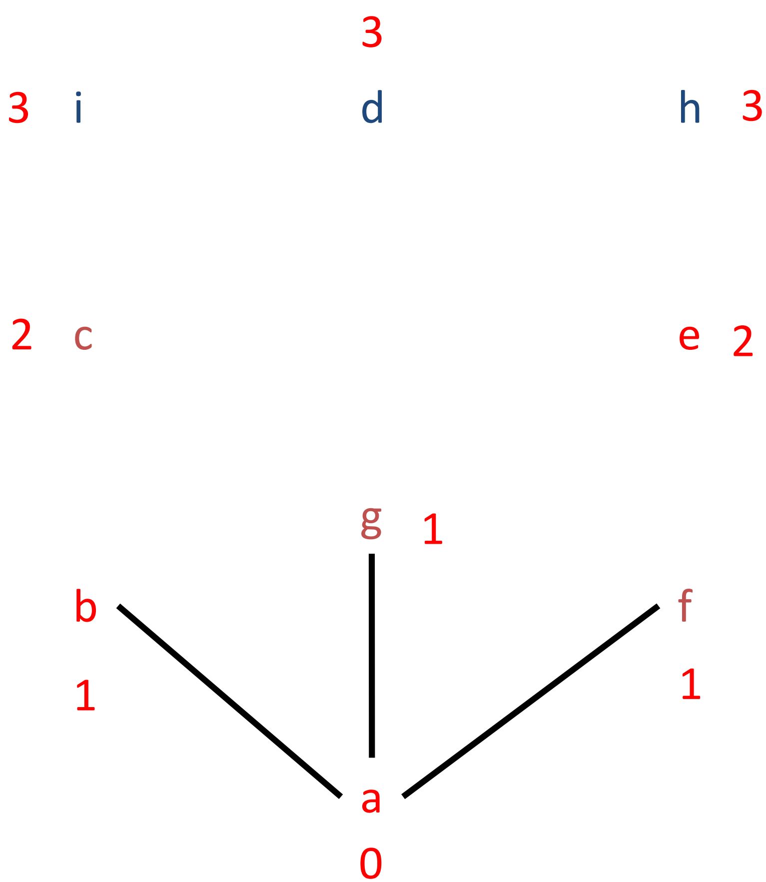


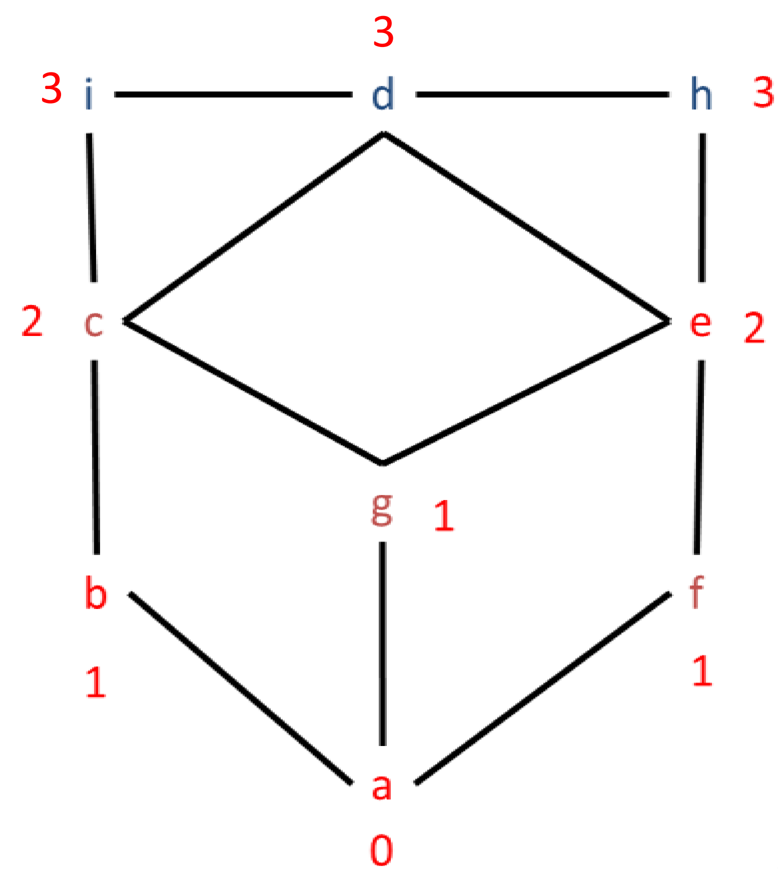
Exemple : Construire l'arbre



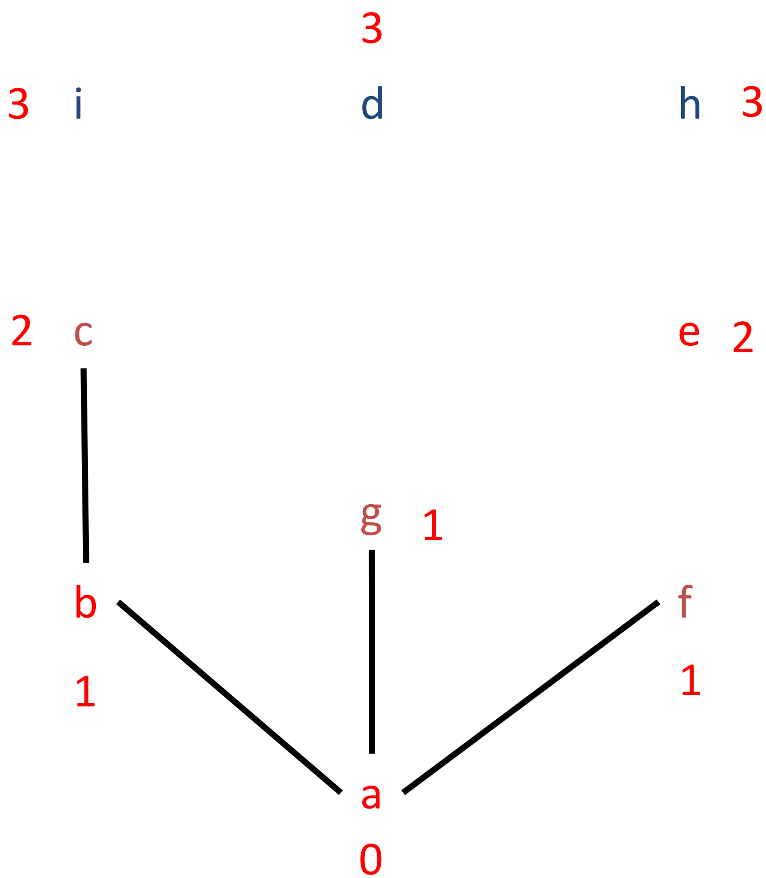


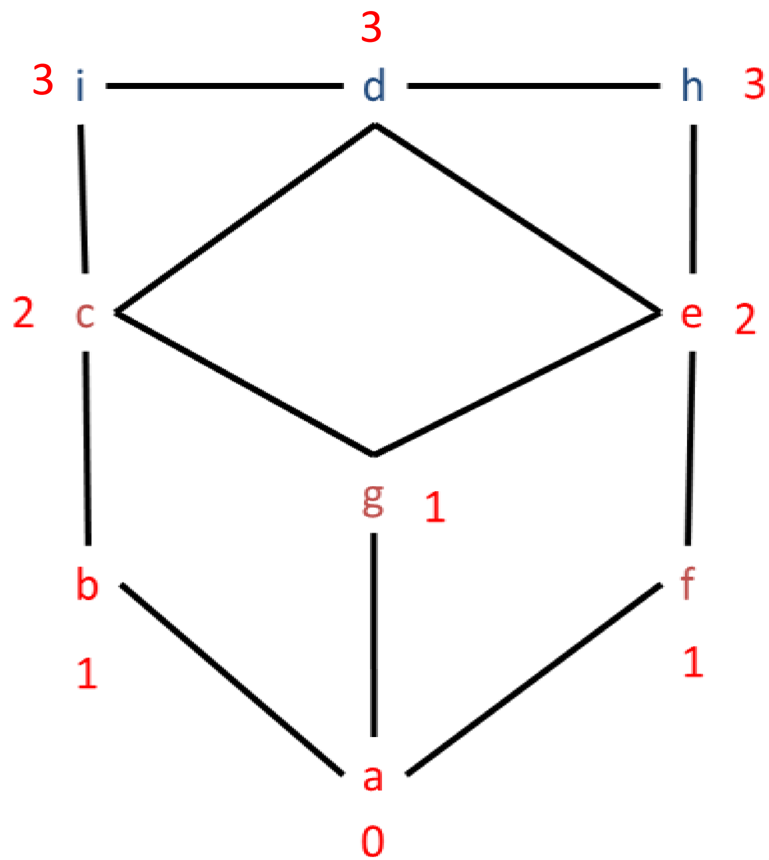
Exemple : Construire l'arbre



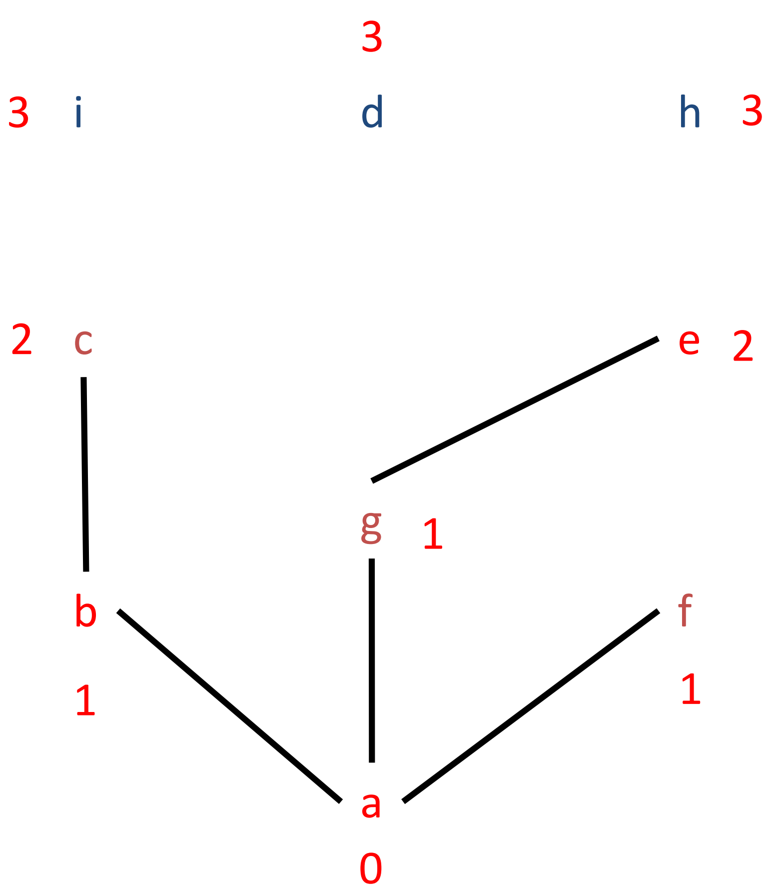


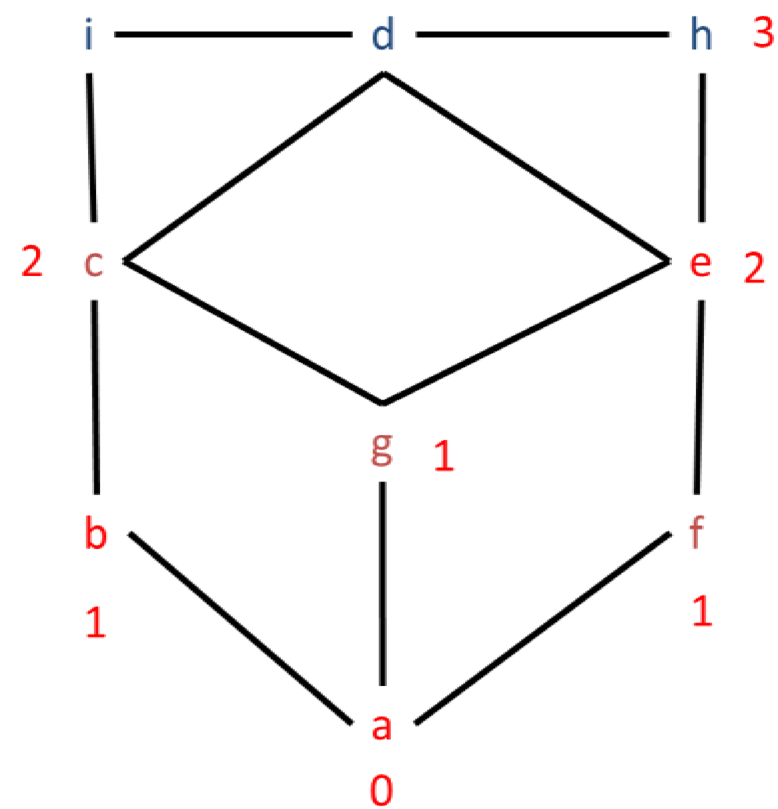
Exemple : Construire l’arbre



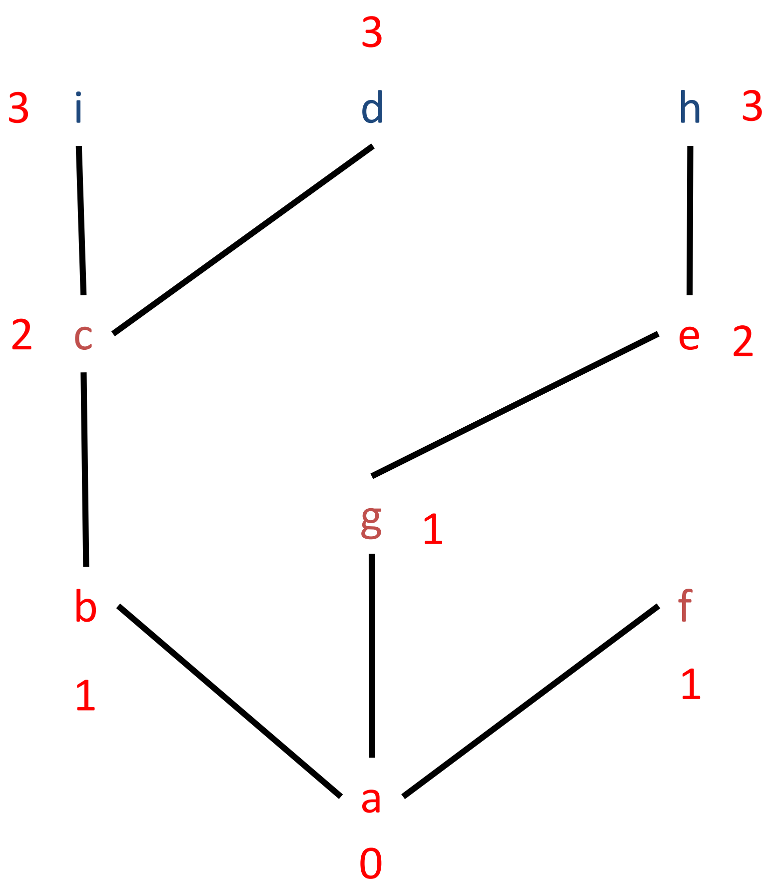


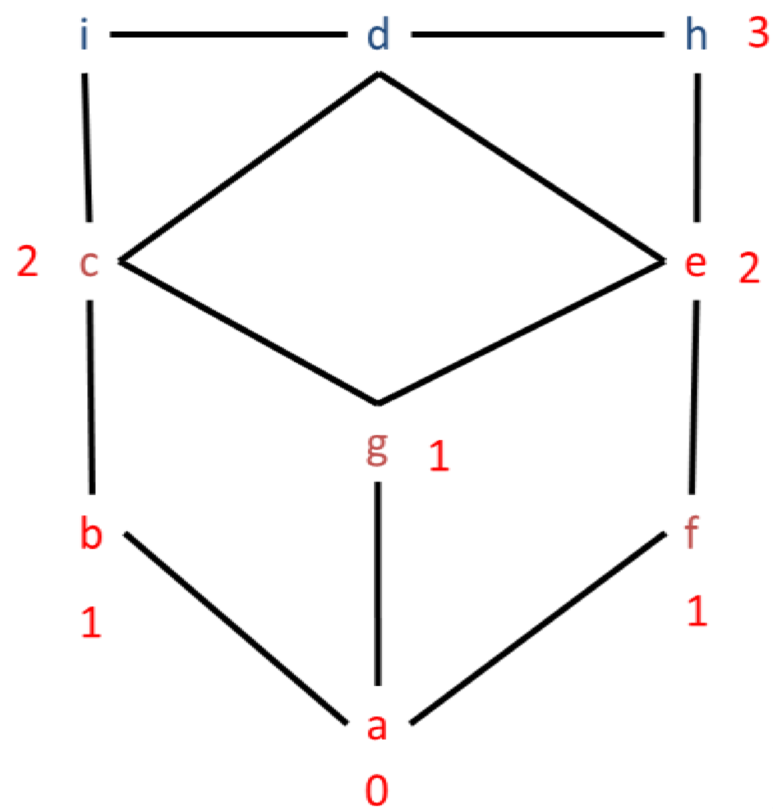
Exemple : Construire l'arbre





Exemple : Construire l'arbre





Remarque

- Les plus courtes chaînes issues de notre sommet a (Sommet de départ de notre algorithme) forment une structure d'arbre couvrant de notre graphe.