Introduction aux graphes

Retour sur les parcours

Mathilde Vernet
mathilde.vernet@univ-avignon.fr

Licence Informatique, CERI
Licence Mathématiques, Institut AgES
Avignon Université

Automne 2025

Plan

1. Rappels sur les parcours
   - Parcours en largeur
   - Parcours en profondeur

2. Algorithme d'identification des composantes fortement connexes
   - DFS avec ordre de pré et post-visite
   - Graphes renversés et méta-graphes de CFC
   - Propriétés
   - Algorithme

Parcours en largeur

Procedure `BFS`(G)
  v.visited $\leftarrow$ false $\forall v \in V$
  Pour $(v) \in V$ **Faire**
    Si $\neg v$.visited **Alors**
      `exploreBFS`(G,v)
    FinSi
  FinPour
FinProcedure

Procedure `exploreBFS`(G,s)
  s.visited $\leftarrow$ true
  open.add(s)
  TantQue $\neg$open.empty() **Faire**
    $v \leftarrow$ open.peek()
    $u \leftarrow$ sommet tel que $(v, u) \in A$
        et u.visited $=$ false
    Si u existe **Alors**
      u.visited $\leftarrow$ true
      open.add(u)
    Sinon
      open.remove()
    FinSi
  FinTantQue
FinProcedure

Remarques

- Procédure générique où `open` est une file
- Opérations de la file :
  - f.`empty()` : la file f est-elle vide ?
  - f.`add(`v`)` : ajoute le sommet v à la fin de la file f
  - f.`peek()` : retourne le premier élément de la file f
  - f.`remove()` : retire et retourne le premier élément de la file f

Principe

- Commencer par un sommet s
- Visiter tous les voisins de s d'abord
- *Avancer dans le graphe* ensuite

Parcours en profondeur

```
Procedure DFS(G)
    v.visited ← false ∀v ∈ V
    Pour (v) ∈ V Faire
        Si ¬v.visited Alors
            exploreDFS(G,v)
        FinSi
    FinPour
FinProcedure
```

```
Procedure exploreDFS(G,s)
    s.visited ← true
    open.push(s)
    TantQue ¬open.empty() Faire
        v ← open.peek()
        u ← sommet tel que (v, u) ∈ A
                et u.visited = false
        Si u existe Alors
            u.visited ← true
            open.push(u)
        Sinon
            open.pop()
        FinSi
    FinTantQue
FinProcedure
```

Remarques

- Procédure générique où `open` est une pile
- Opérations de la pile :
  - p.`empty()` : la pile p est-elle vide ?
  - p.`push(`v`)` : ajoute le sommet v sur le dessus de la pile p
  - p.`peek()` : retourne l'élément sur le dessus de la pile p
  - p.`pop()` : retire et retourne l'élément sur le dessus de la pile p

Principe

- Commencer par un sommet s
- *Avancer dans le graphe* le plus loin possible d'abord
- Revenir sur les voisins de s ensuite

Remarque

- Le DFS peut s'écrire de façon récursive
  - Visiter s
  - Puis relancer la procédure sur un voisin

Parcours en profondeur avec récursivité

Procedure `DFSRec`(G)
  v.visited $\leftarrow$ false $\forall v \in V$
  Pour $(v) \in V$ **Faire**
    Si $\neg v$.visited **Alors**
      `exploreDFSRec`(G,v)
    FinSi
  FinPour
FinProcedure

Procedure `exploreDFSRec`(G,s)
  s.visited $\leftarrow$ true
  Pour $(s,u) \in E$ **Faire**
    Si $\neg u$.visited **Alors**
      `exploreDFSRec`(G,u)
    FinSi
  FinPour
FinProcedure

Noter les dates de visites

- Utilisation des procédure `previsit` et `postvisit`

Parcours en profondeur avec dates de pré et post-visite

Procedure `DFS`(G)
  v.visited $\leftarrow$ false $\forall v \in V$
  clock $\leftarrow 1$
  Pour $(v) \in V$ **Faire**
    Si $\neg v$.visited **Alors**
      `exploreDFS`(G,v)
    FinSi
  FinPour
FinProcedure

Procedure `exploreDFS`(G,s)
  s.visited $\leftarrow$ true
  `previsit`(s)
  Pour $(s, u) \in E$ **Faire**
    Si $\neg u$.visited **Alors**
      `exploreDFS`(G,u)
    FinSi
  FinPour
  `postvisit`(s)
FinProcedure

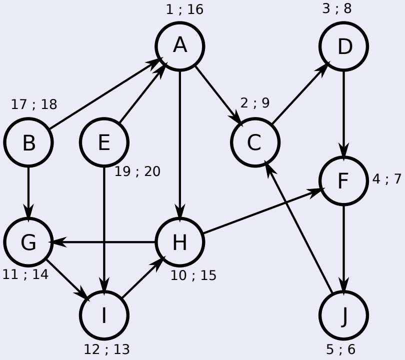
Procedure `previsit`(v)
  v.pre $\leftarrow$ clock
  clock $\leftarrow$ clock $+ 1$
FinProcedure

Procedure `postvisit`(v)
  v.post $\leftarrow$ clock
  clock $\leftarrow$ clock $+ 1$
FinProcedure

Remarques sur les opérations de pré et post-visite

- Ici, ces opérations marquent des dates
- La date de pré-visite correspond au moment où on *arrive* sur le sommet pour le visiter
- La date de post-visite correspond au moment où on ne peut plus exploiter ce sommet ni les sommets auxquels il peut accéder

Exemple

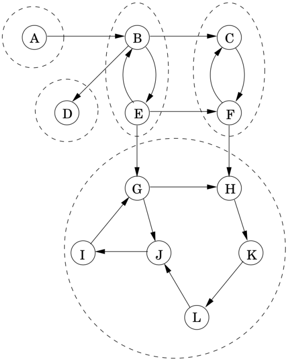


Graphe renversé

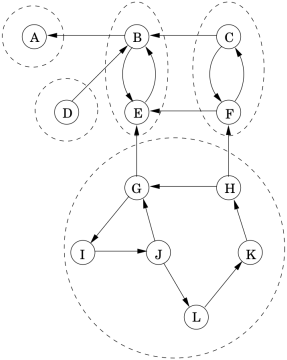
On appelle *graphe renversé* d'un graphe orienté $G = (V, A)$ le graphe orienté $G^R = (V, A^R)$ tel que :

- $A^R = \{(u, v) | (v, u) \in A\}$

Exemple : graphe initial G



Exemple : graphe renversé G^R



Propriété

Un graphe G et son graphe renversé G^R possèdent les mêmes composantes fortement connexes

Pourquoi G et G^R ont-ils les mêmes composantes fortement connexes ?

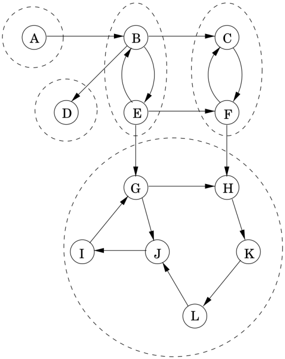
- u et v sont dans la même composante fortement connexe si et seulement si il existe :
  - un chemin de u à v dans le graphe
  - un chemin de v à u dans le graphe
- Un chemin de u à v dans G devient un chemin de v à u dans G^R
- S'il y a un chemin de u à v et un chemin de v à u dans G alors il y a un chemin de v à u et un chemin de u à v dans G^R

Méta-graphe des composantes fortement connexes

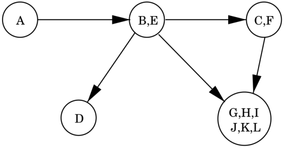
On appelle *méta-graphe* des composantes fortement connexes d'un graphe orienté G le graphe orienté dans lequel :

- chaque sommet C_i correspond à une composante fortement connexe de G
- chaque arc (C_i, C_j) correspond à la présence dans G d'un arc d'un sommet de la composante C_i vers un sommet de la composante C_j

Exemple : graphe initial G



Exemple : méta-graphe des CFC de G



Objectif

Nous souhaitons concevoir un algorithme afin d'identifier les composantes fortement connexes d'un graphe orienté

Comment faire ?

Pour cela, montrons les propriétés suivantes :

1. Le méta-graphe des composantes-fortement connexes est un DAG
2. Chaque DAG a au moins un sommet *source* (sommet avec un degré entrant nul) et au moins un sommet *puits* (sommet avec un degré sortant nul)
3. Si la procédure `explore` commence par un sommet qui se trouve dans une composante *puits*, alors elle va parcourir exactement cette composante
4. Le sommet avec la date de post-visite la plus grande se trouve forcément dans une composante *source*

Propriété 1

Démonstration.

Si le méta-graphe contenait un cycle entre deux composantes, alors il existerait :

- un chemin de n'importe quel sommet de la première composante vers n'importe quel sommet de la deuxième composante

ET

- un chemin de n'importe quel sommet de la deuxième composante vers n'importe quel sommet de la première composante

$\Rightarrow$ Et donc tous ces sommets seraient dans la même composante fortement connexe.
$\Rightarrow$ Ces composantes ne formeraient alors qu'un seul sommet dans le méta-graphe.
Donc le méta-graphe des composantes-fortement connexes est un DAG. $\square$

Propriété 2

Démonstration.

- Un DAG admet un ordre topologique sur les sommets
- Le premier sommet de cet ordre n'a pas de voisins entrants
- Le dernier sommet de cet ordre n'a pas de voisins sortants

Donc chaque DAG a au moins un sommet *source* et au moins un sommet *puits*

□

Propriété 3

Démonstration.

- Une composante *puits* n'a pas d'arc sortant
- La procédure `explore` à partir d'un sommet s visite tous les sommets accessibles depuis s
- Si s se trouve dans une composante *puits* alors les seuls sommets accessibles depuis s sont ceux de sa composante fortement connexe

Donc si on lance la procédure `explore` depuis un sommet d'une composante *puits*, on va parcourir exactement les sommets de cette composante $\square$

Propriété 4

Démonstration.

Soient C et C' deux composantes sommets du méta-graphe telles qu'il existe un arc d'un sommet de C vers un sommet de C'. Deux cas :

- Si on visite C avant C' alors la procédure `explore` aura visité tous les sommets de C et de C' avant de s'arrêter. Donc le sommet par lequel la procédure a commencé dans C aura une date date de post-visite plus grande que n'importe quel sommet de C'.
- Si on visite C' avant C, la procédure `explore` s'arrêtera avant d'avoir visité les sommets de C puisqu'ils ne sont pas accessible depuis C'. Donc il faudra relancer la procédure sur C et tous les sommets de C auront une date de post-visite plus grande que les sommets de C'.

$\Rightarrow$ On en déduit que si un sommet s' se trouve dans une composante C' qui n'est pas une composante source alors un sommet s d'une composante C telle qu'il existe un arc d'un sommet de C vers un sommet de C' aura une plus grande date de post-visite que s'.
Donc le sommet avec la date de post-visite la plus grande se trouve forcément dans une composante *source*. □

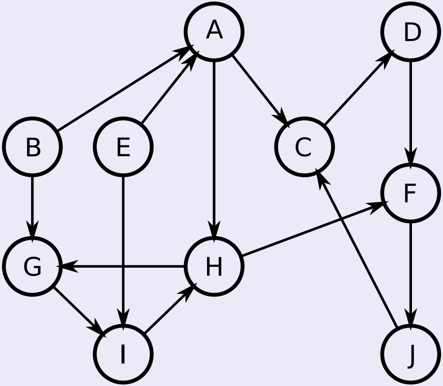
Comment exploiter ces propriétés ?

- On sait que si on lance un DFS depuis un sommet s dans une composante *puits*, on parcourra exactement les sommets de la composante fortement connexe de s
- On sait comment identifier un sommet d'une composante *source*
  - Comment identifier un sommet d'une composante *puits* ?
- On sait que le graphe renversé possède les mêmes composantes fortement connexes que le graphe d'origine

Idée de l'algorithme d'identification

1. Exécuter un DFS sur le graphe renversé G^R
   On identifie ainsi un sommet d'un composante puits grâce aux dates de post visite
2. Exécuter l'algorithme d'identification des composantes connexes pour les graphes non-orientés sur G en sélectionnant les sommets dans l'ordre décroissant des dates de post-visite sur G^R
   On parcours ainsi d'abord toute la composante puits, puis les composantes les unes après les autres

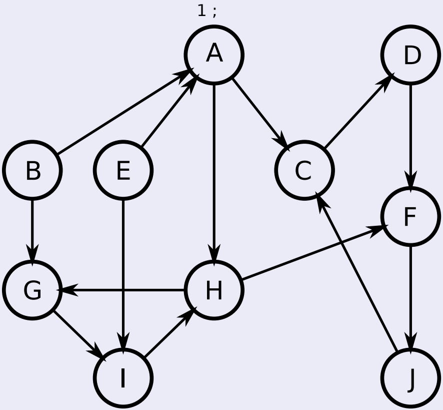
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

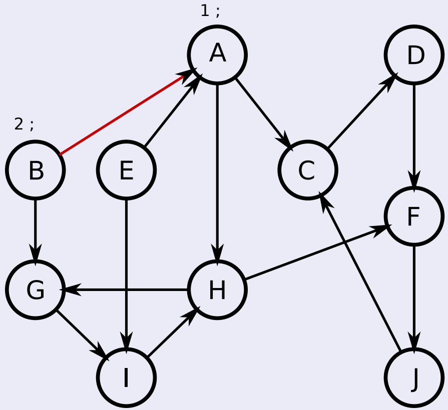
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

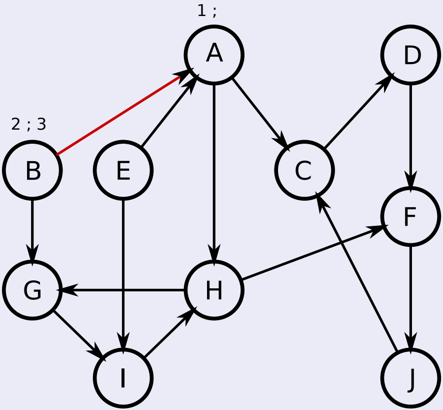
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

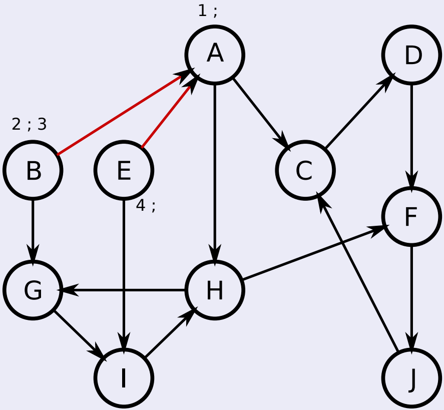
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

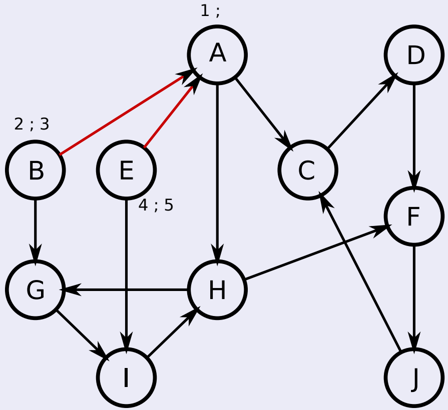
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

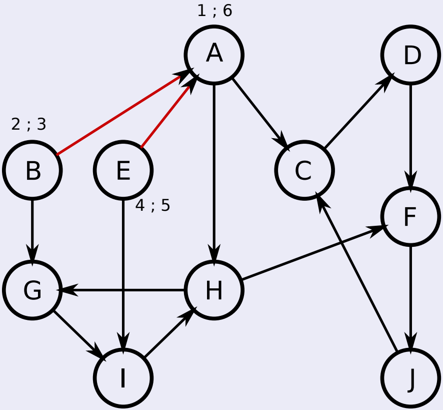
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

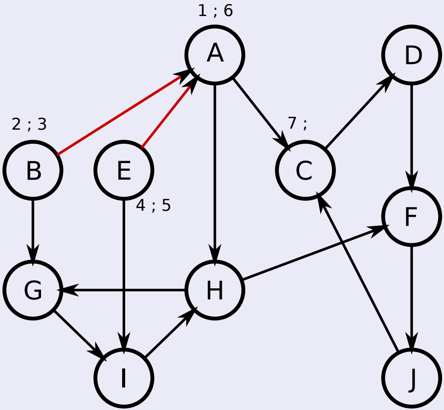
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

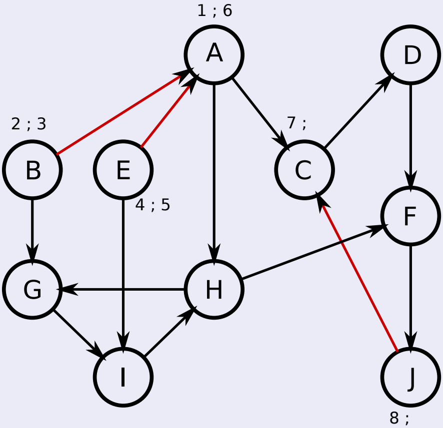
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

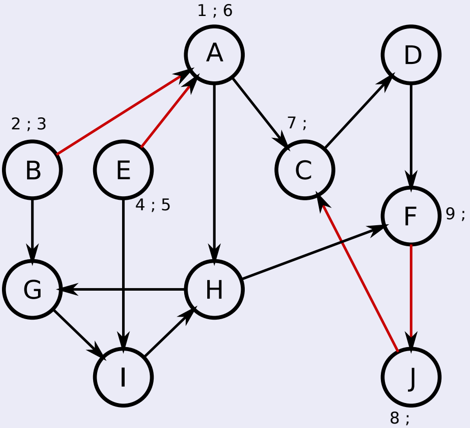
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

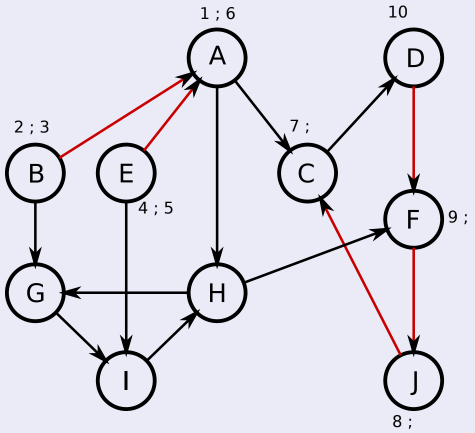
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

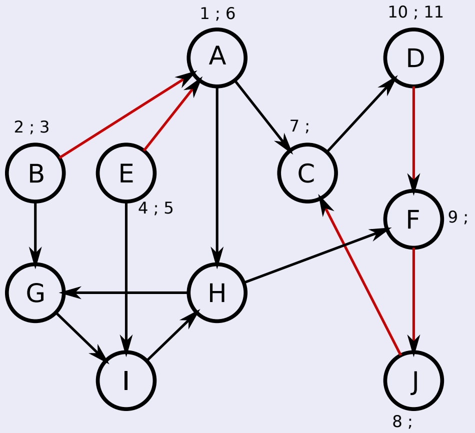
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

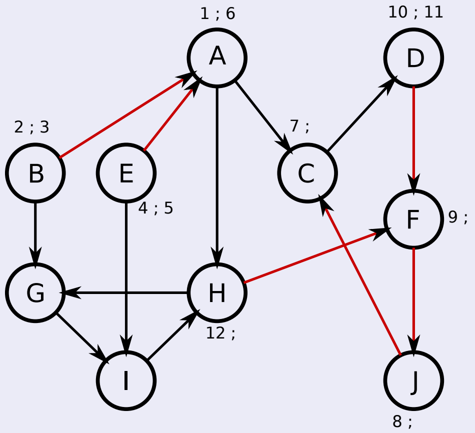
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

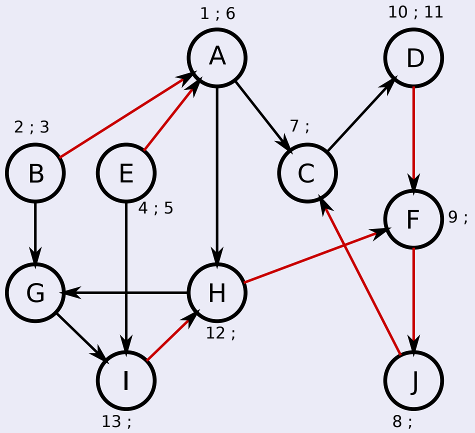
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

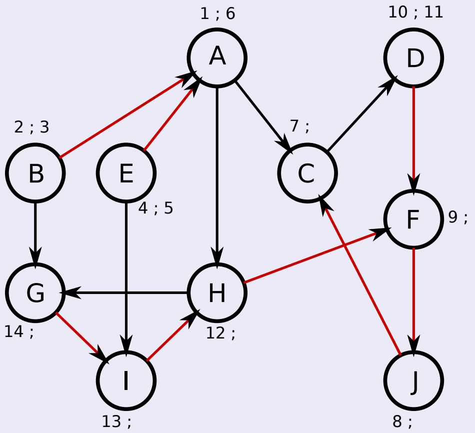
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

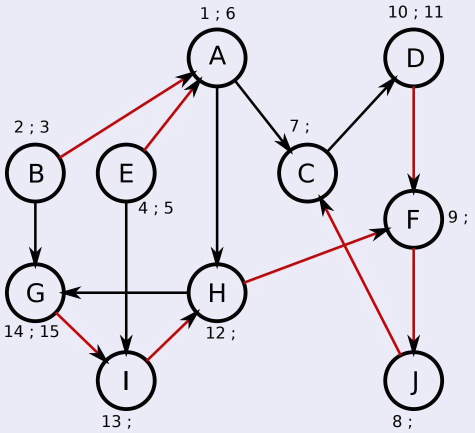
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

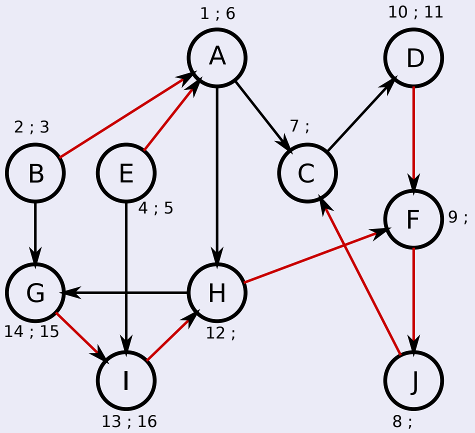
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

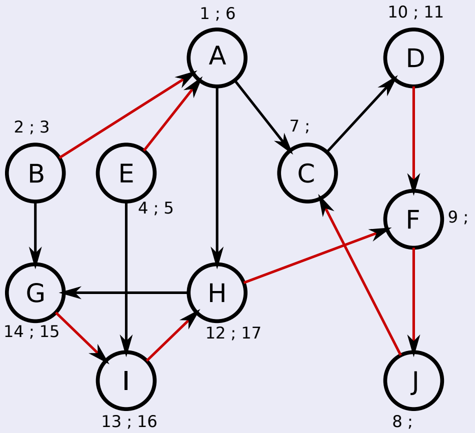
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

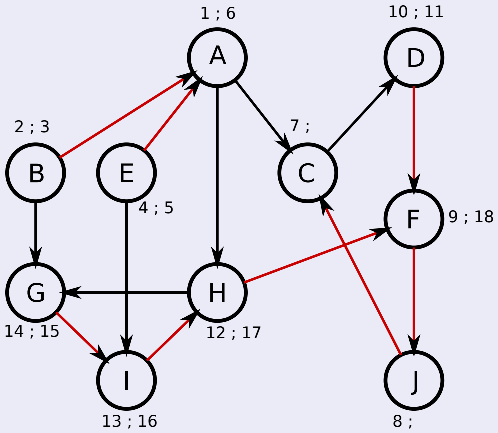
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

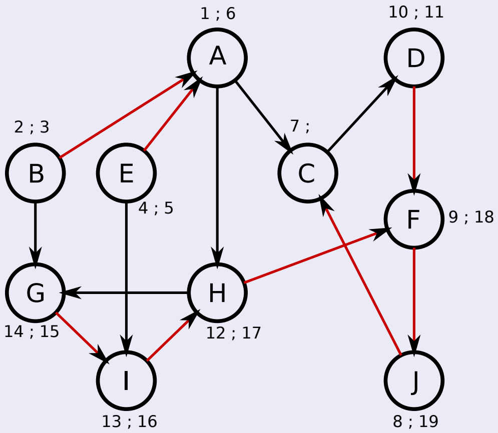
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

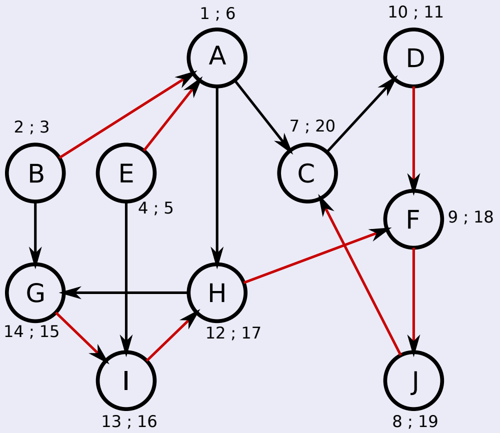
Exemple : Appliquer DFS sur le graphe renversé



Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

Exemple : Appliquer DFS sur le graphe renversé



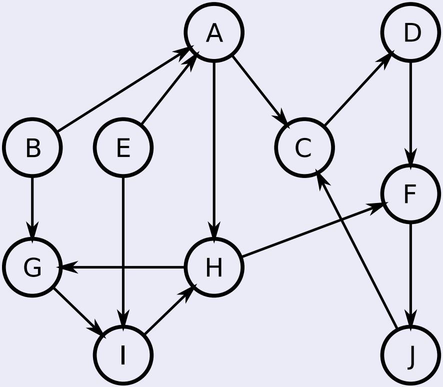
Remarque

Appliquer DFS sur le graphe renversé est équivalent à appliquer DFS en considérant les arcs *à l'envers*

Ordre décroissant des post-visite

C ; J ; F ; H ; I ; G ; D ; A ; E ; B

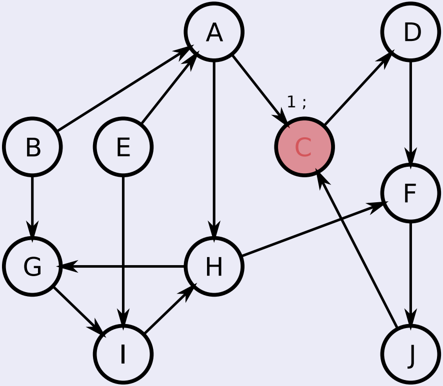
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~; J ; F ; H ; I ; G ; D ; A ; E ; B

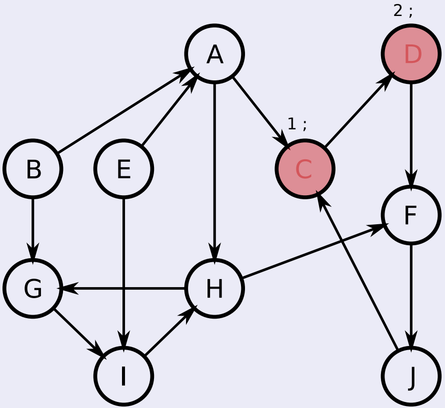
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; J ; F ; H ; I ; G ; ~~D~~ ; A ; E ; B

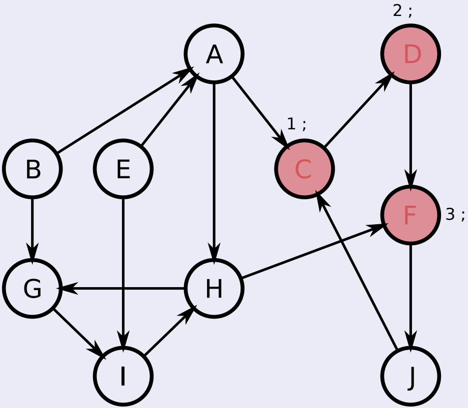
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; J ; ~~F~~ ; H ; I ; G ; ~~D~~ ; A ; E ; B

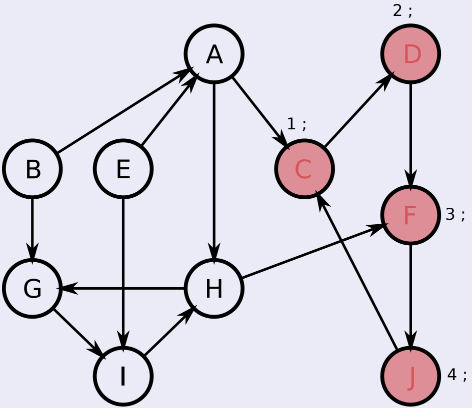
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; H ; I ; G ; ~~D~~ ; A ; E ; B

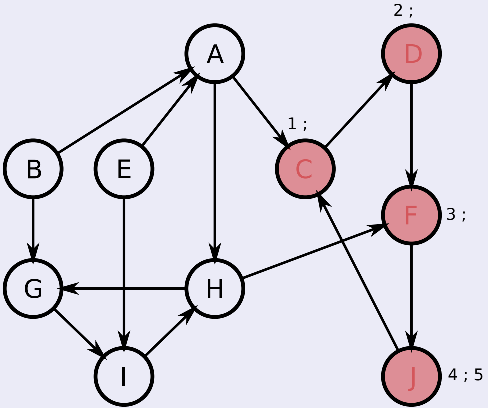
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; H ; I ; G ; ~~D~~ ; A ; E ; B

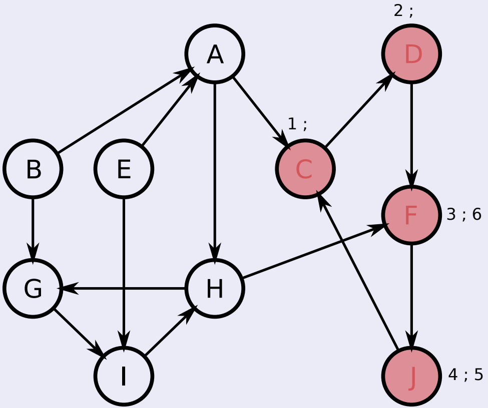
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; H ; I ; G ; ~~D~~ ; A ; E ; B

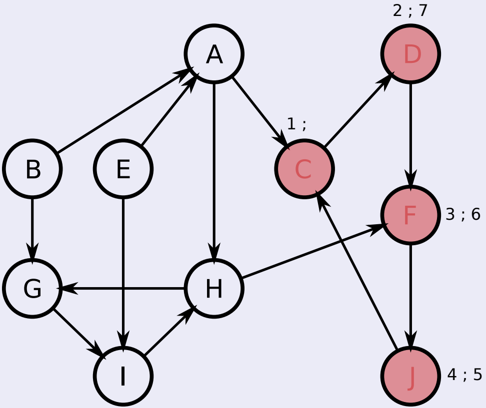
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; H ; I ; G ; ~~D~~ ; A ; E ; B

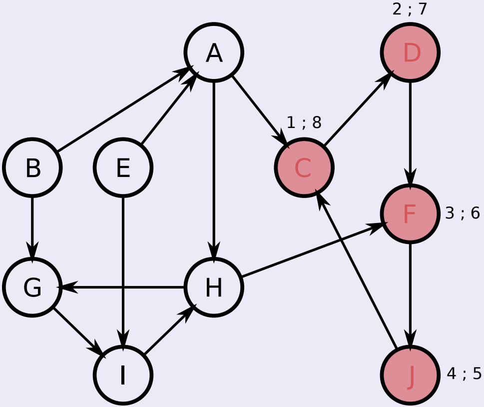
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; H ; I ; G ; ~~D~~ ; A ; E ; B

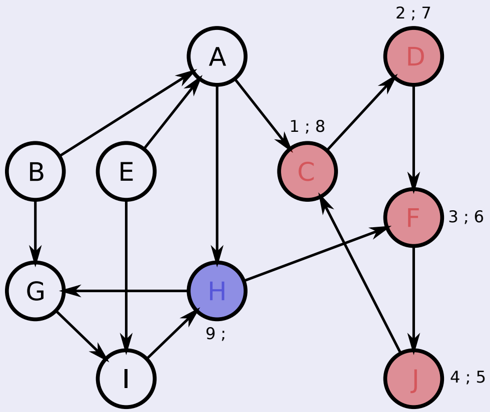
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; I ; G ; ~~D~~ ; A ; E ; B

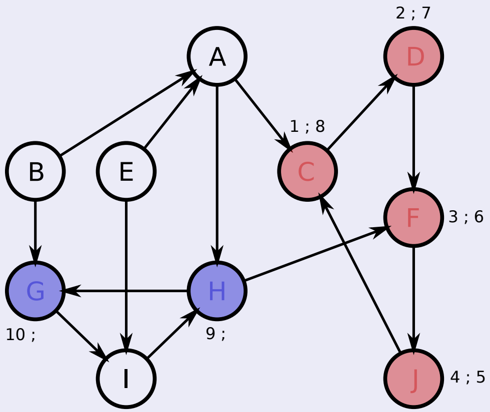
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; I ; ~~G~~ ; ~~D~~ ; A ; E ; B

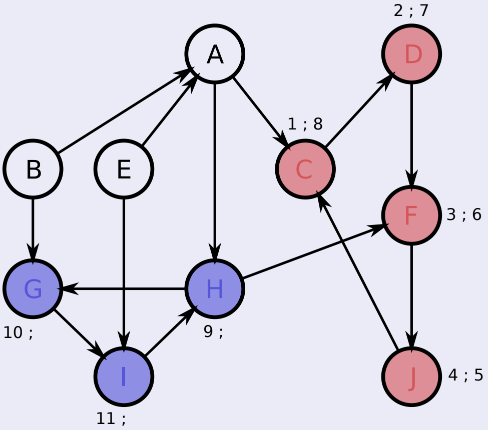
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; A ; E ; B

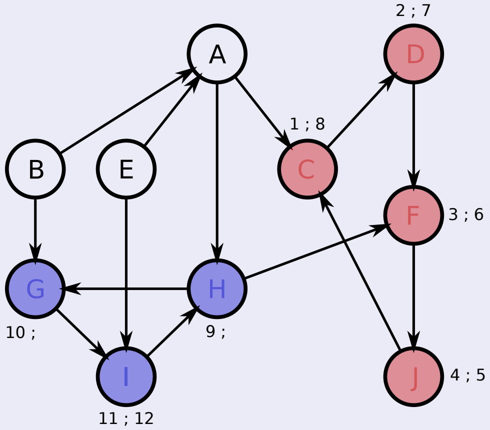
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; A ; E ; B

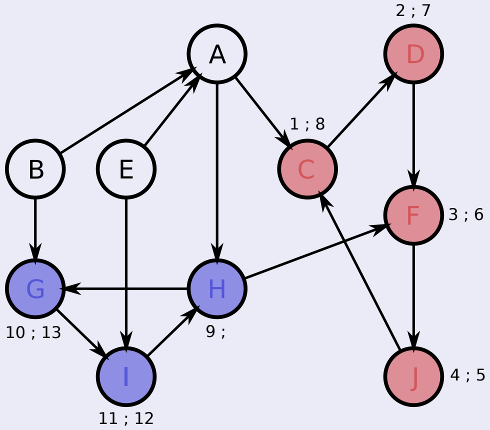
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; A ; E ; B

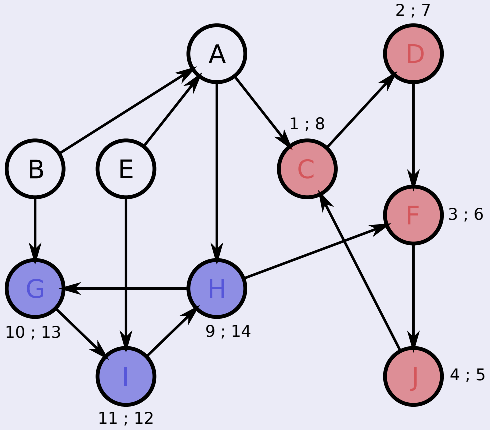
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; A ; E ; B

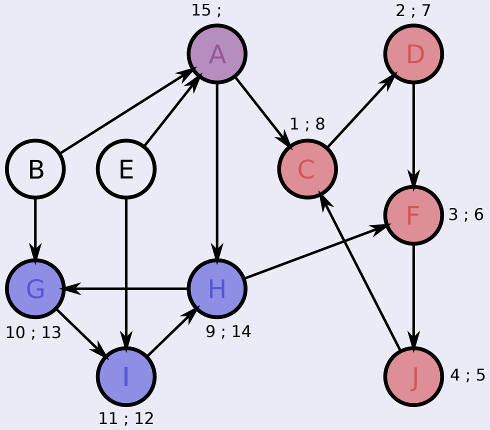
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; ~~A~~ ; E ; B

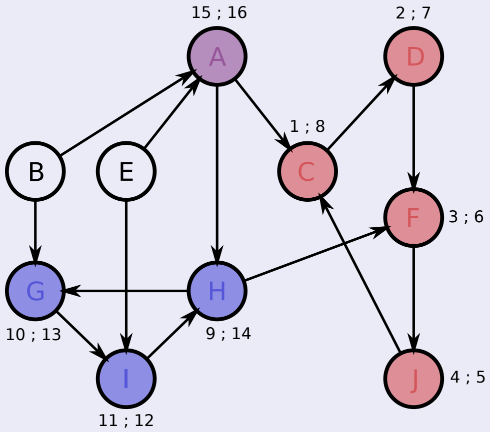
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; ~~A~~ ; E ; B

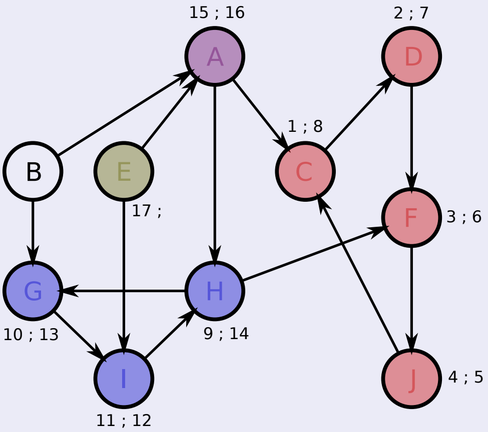
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; ~~A~~ ; ~~E~~ ; B

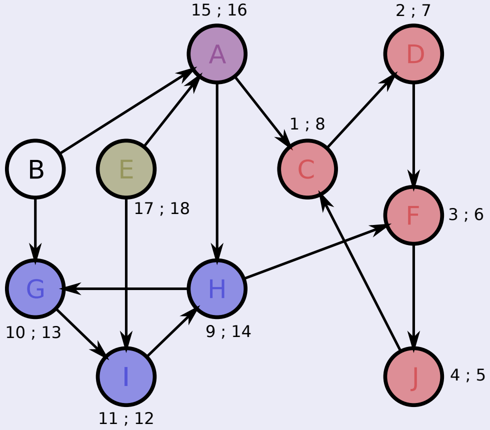
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; ~~A~~ ; ~~E~~ ; B

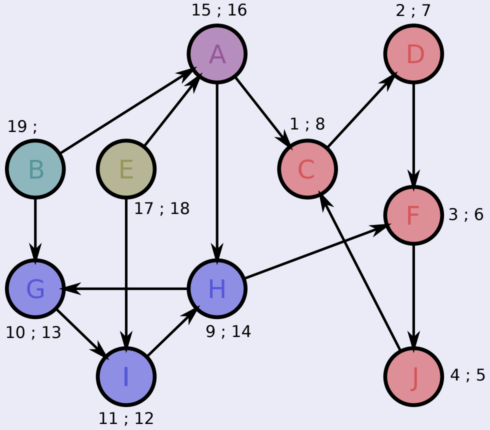
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; ~~A~~ ; ~~E~~ ; ~~B~~

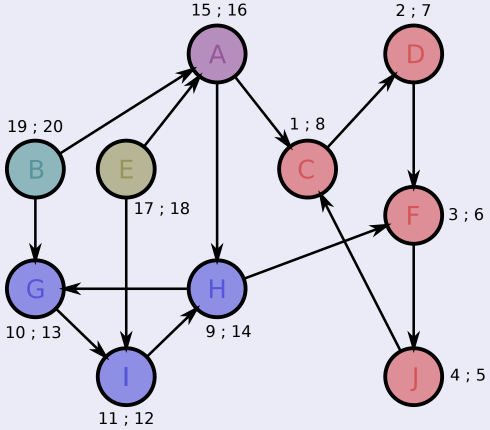
Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Ordre décroissant des post-visite

~~C~~ ; ~~J~~ ; ~~F~~ ; ~~H~~ ; ~~I~~ ; ~~G~~ ; ~~D~~ ; ~~A~~ ; ~~E~~ ; ~~B~~

Exemple : Appliquer DFS en suivant l'ordre des dates de post-visite



Exemple : Représentation du méta-graphe des composantes fortement connexes