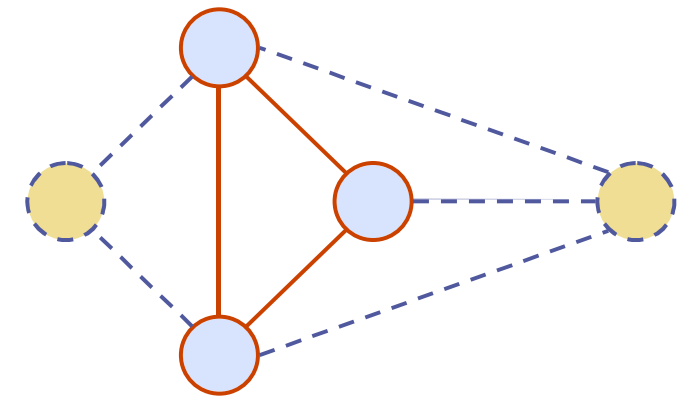


Recherche en profondeur (Depth-First Search) et recherche en largeur (Breadth-First Search) (§12.3)

Sous-graphes

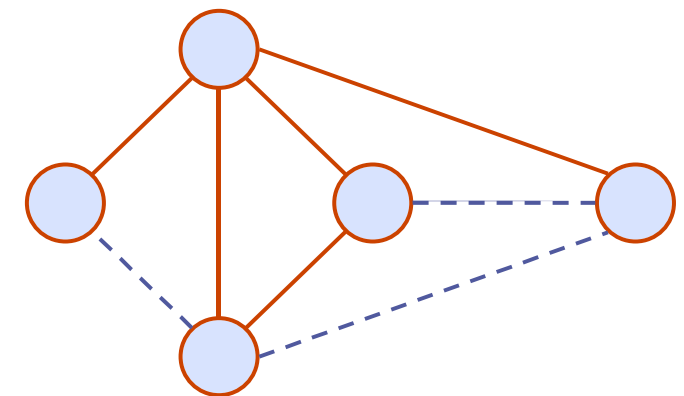
● Un sous-graphe S d'un graphe G est un graphe tel que

- les sommets de S forment un sous-ensemble des sommets de G
- les arêtes de S forment une sous-collection des arêtes de G



Sous-graphe

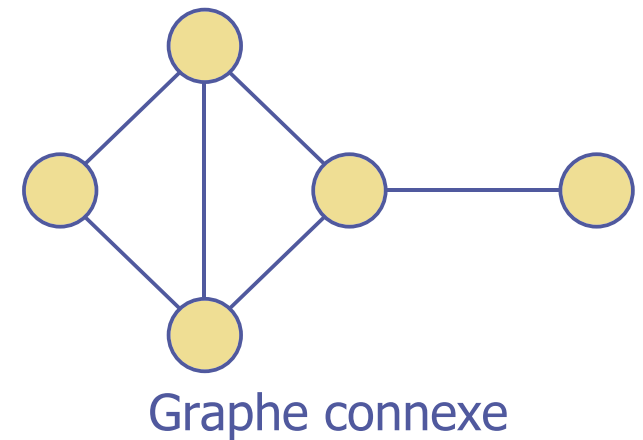
● Un sous-graphe couvrant d'un graphe G est sous-graphe contenant tous les sommets de G



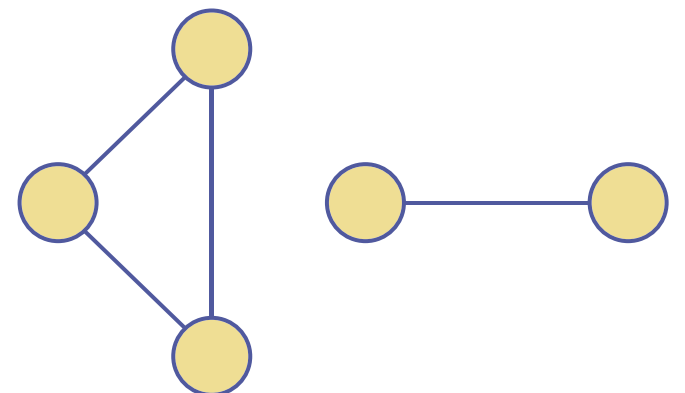
Sous-graphe couvrant

Graphes connexes

- Un graphe est connexe s'il existe un chemin entre chaque paire de sommets du graphe



- Une composante connexe d'un graphe G est un sous-graphe connexe maximal de G

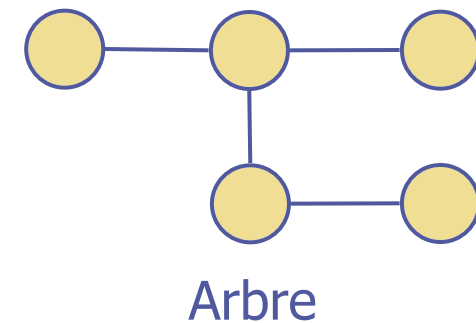


Un graphe non-connexe ayant deux composantes connexes

Arbres et forêts

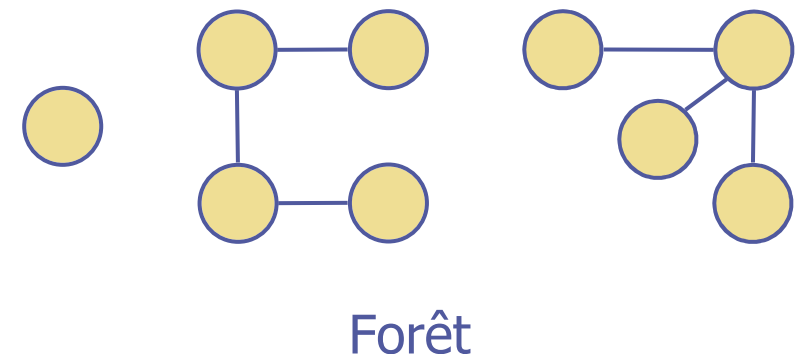
○ Un arbre est un graphe non-orienté T tel que

- T est connexe
- T n'a pas de cycles



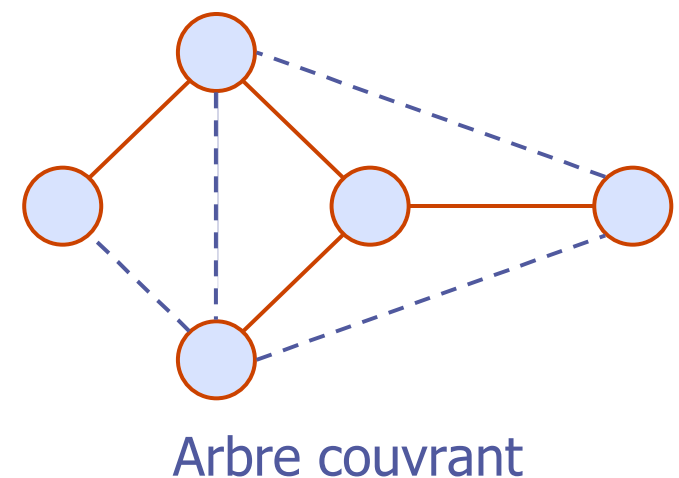
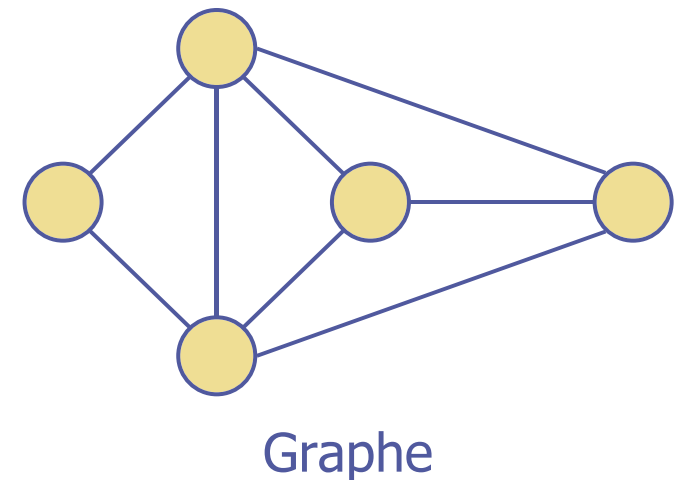
○ Une forêt est un graphe non-orienté qui n'a pas de cycles

○ Les composantes connexes d'une forêt sont des arbres



Arbres et forêts couvrants






- Un arbre couvrant d'un graphe connexe est un sous-graphe couvrant qui est un arbre.
- Un arbre couvrant n'est pas unique sauf si le graphe est un arbre.
- Une forêt couvrante d'un graphe est un sous-graphe couvrant qui est une forêt.

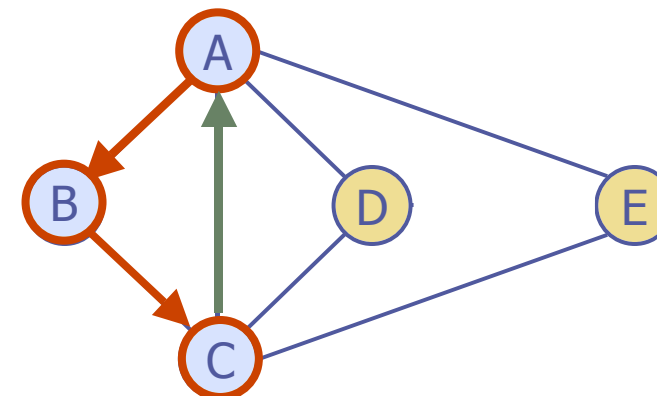
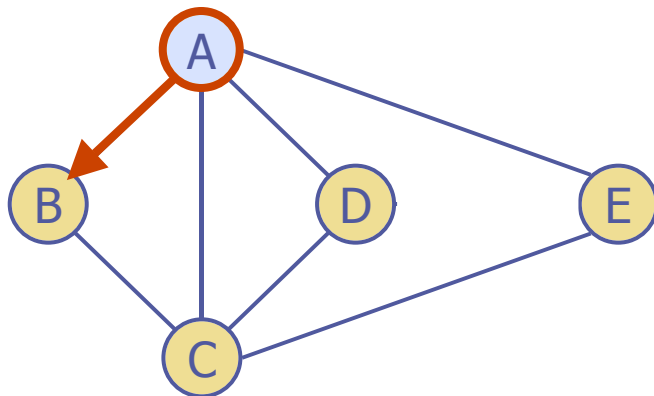
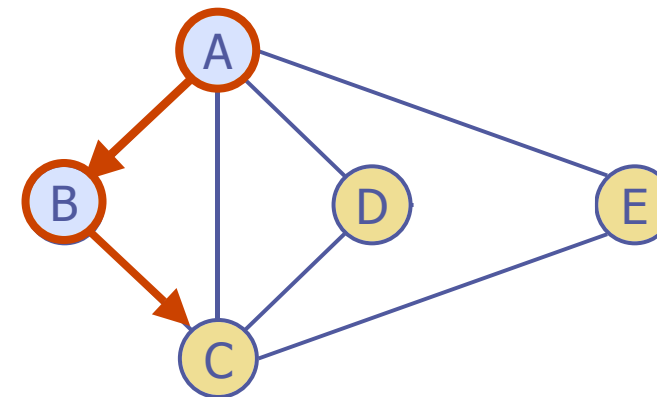


Parcours en profondeur (Depth-First Search)

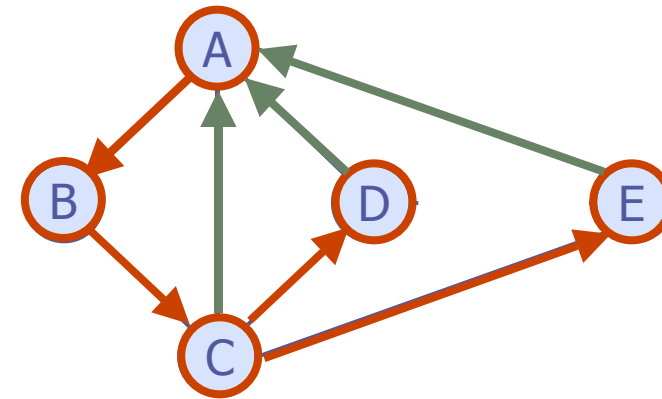
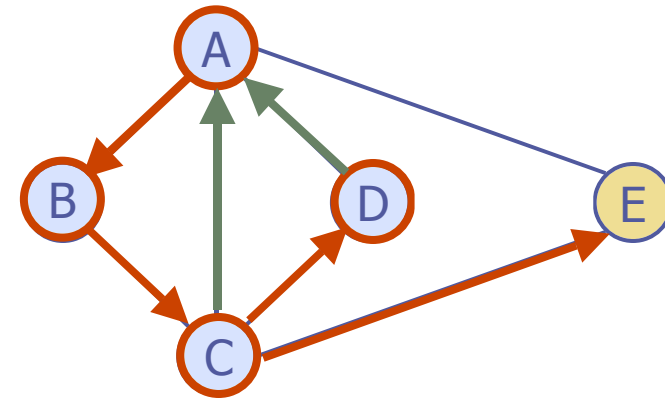
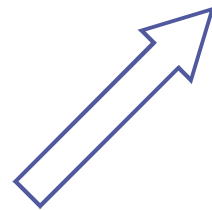
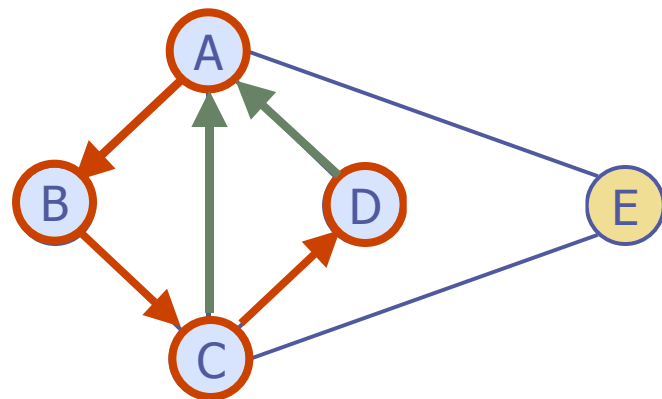
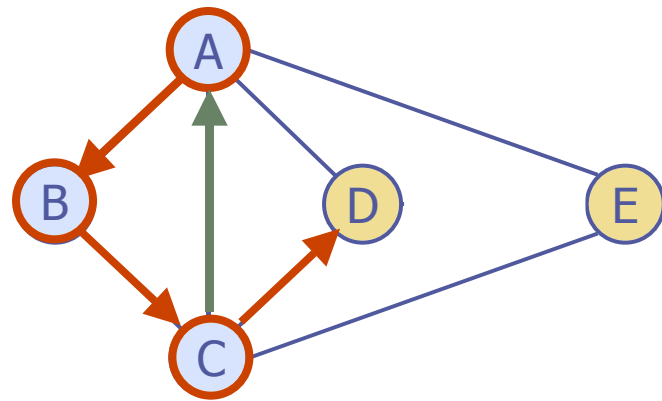
- Un parcours en profondeur d'un graphe G
 - ▣ Visite tous les sommets et toutes les arêtes de G
 - ▣ Détermine si G est connexe ou non
 - ▣ Calcule les composantes connexes de G
 - ▣ Calcule une forêt couvrante pour G
- La complexité en temps d'un parcours en profondeur est $O(n+m)$
- L'algorithme de parcours en profondeur peut être modifié pour résoudre d'autres problèmes sur les graphes:
 - ▣ trouver et retourner un chemin entre deux sommets
 - ▣ trouver un cycle dans un graphe

Parcours en profondeur: Exemple

-  Sommets non-explorés
-  Sommets visités
-  Arêtes non-explorées
-  Arêtes sélectionnées
-  Arêtes de retour



Parcours en profondeur: Exemple



Analyse du parcours en profondeur

- Chaque sommet est étiqueté deux fois
 - ▣ Une fois “non-exploré”
 - ▣ Une fois “visité”
- Chaque arête est étiquetée deux fois
 - ▣ Une fois “non-explorée”
 - ▣ Une fois “sélectionnée” ou “retour”
- L'opération *incidentes(s)* est appelée pour chaque sommet
- La complexité en temps du parcours en profondeur est en $O(n+m)$ si on utilise la structure de données “liste d'adjacence”

Chemin entre deux sommets

```
Algorithme trouverChemin( $G, v, z$ )  
  changerÉtiquette( $v, VISITÉ$ )  
  S.empiler( $v$ )  
  si  $v = z$   
    retourner S.éléments()  
  Pour tout  $e \in G.incidentes(v)$   
    si Étiquette( $e$ ) = NON-EXPLORÉ  
       $w \leftarrow opposé(v, e)$   
      si Étiquette( $w$ ) = NON-EXPLORÉ  
        changerÉtiquette( $e, SÉLECTIONNÉ$ )  
        S.empiler( $e$ )  
        trouverChemin( $G, w, z$ )  
        S.dépiler( $e$ )  
      sinon  
        changerÉtiquette( $e, RETOUR$ )  
  S.dépiler( $v$ )
```






Trouver un cycle dans un graphe

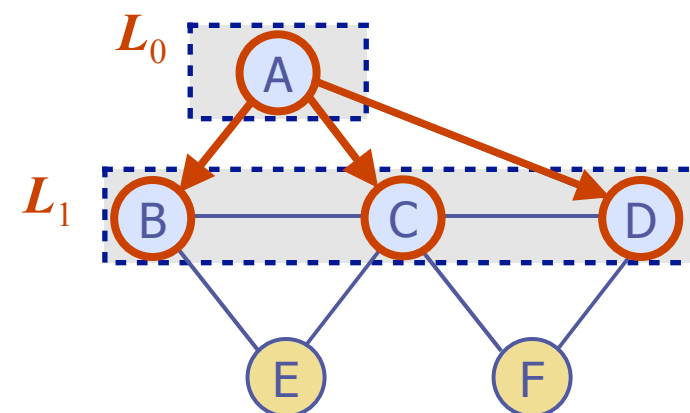
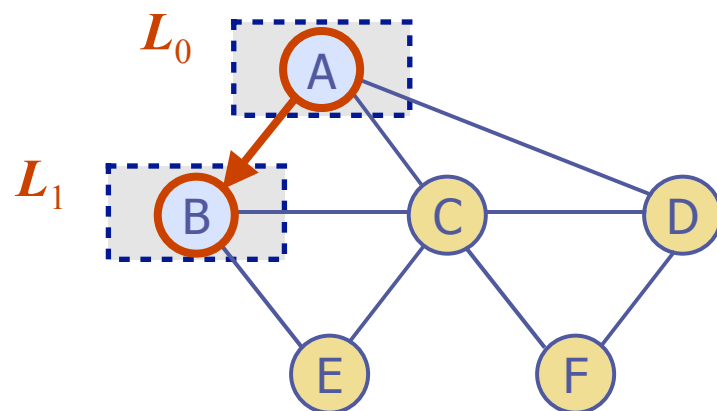
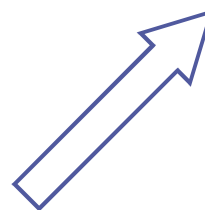
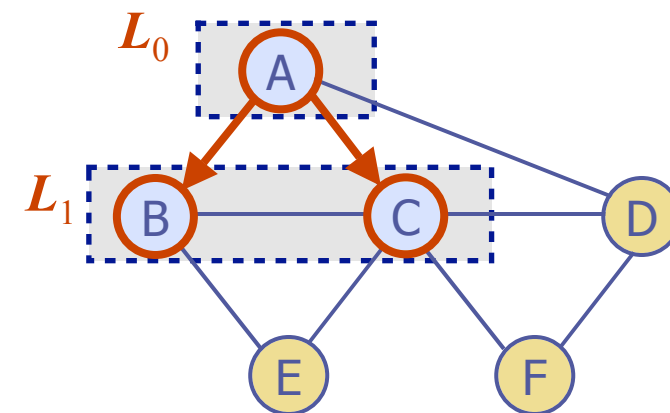
```
Algorithme trouverCycle( $G, v, z$ )  
  changerÉtiquette( $v, VISITÉ$ )  
  S.empiler( $v$ )  
  Pour tout  $e \in G.incidentes(v)$   
    si Étiquette( $e$ ) = NON-EXPLORÉ  
       $w \leftarrow opposé(v, e)$   
      S.empiler( $e$ )  
      si Étiquette( $w$ ) = NON-EXPLORÉ  
        changerÉtiquette( $e, SÉLECTIONNÉ$ )  
        trouverChemin( $G, w, z$ )  
        S.dépiler( $e$ )  
      sinon  
         $T \leftarrow$  nouvelle pile vide  
        répéter  
           $o \leftarrow S.dépiler()$   
          T.empiler( $o$ )  
        tant que  $o = w$   
        retourner T.éléments()  
  S.dépiler( $v$ )
```

Parcours en largeur (Breadth-First Search)

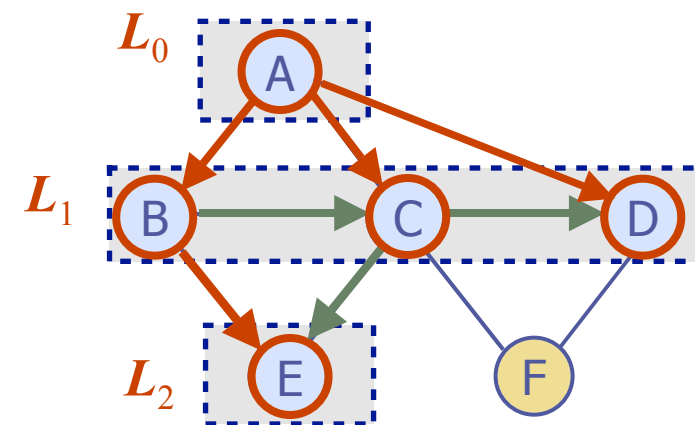
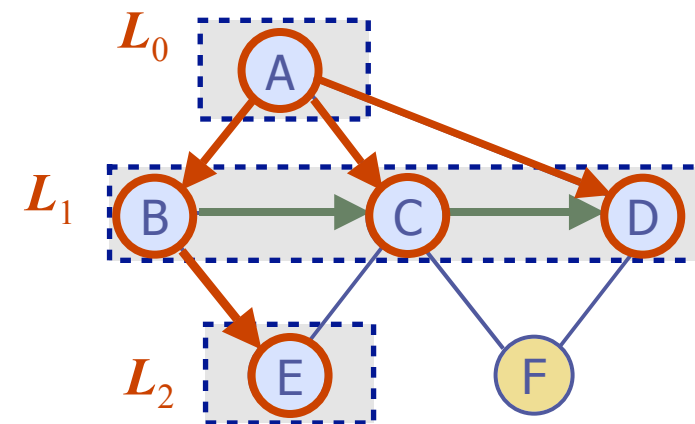
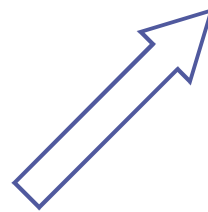
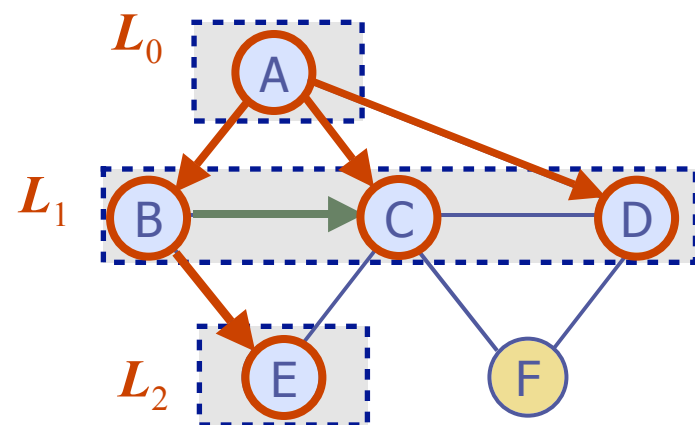
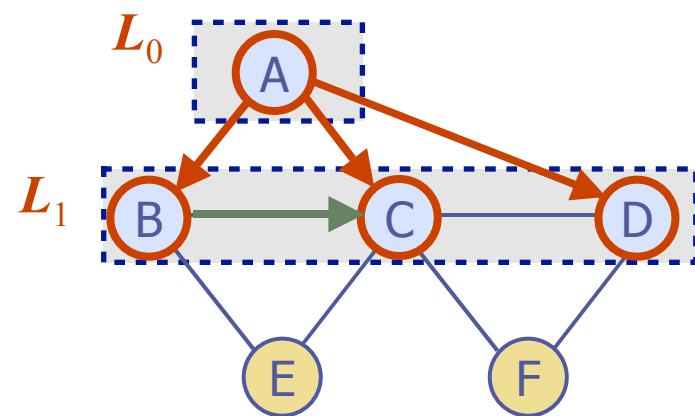
- Un parcours en largeur d'un graphe G
 - ▣ Visite tous les sommets et toutes les arêtes de G
 - ▣ Détermine si G est connexe ou non
 - ▣ Calcule les composantes connexes de G
 - ▣ Calcule une forêt couvrante pour G
- La complexité en temps d'un parcours en largeur est $O(n+m)$
- L'algorithme de parcours en profondeur peut être modifié pour résoudre d'autres problèmes sur les graphes:
 - ▣ trouver et retourner un chemin de longueur **minimale** entre deux sommets
 - ▣ trouver un cycle simple, s'il y en a un

Parcours en largeur: Exemple

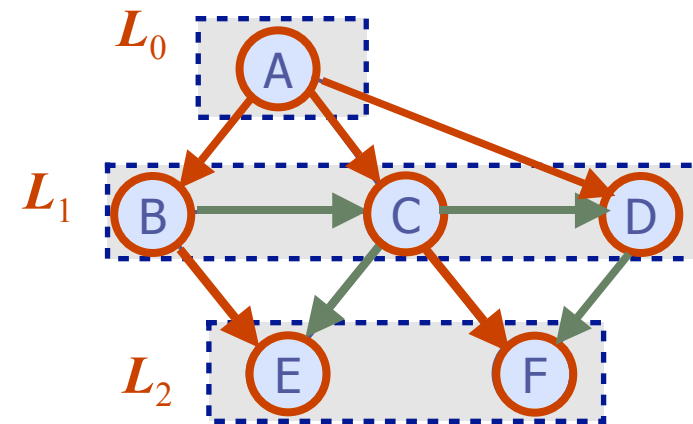
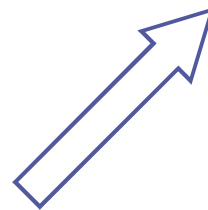
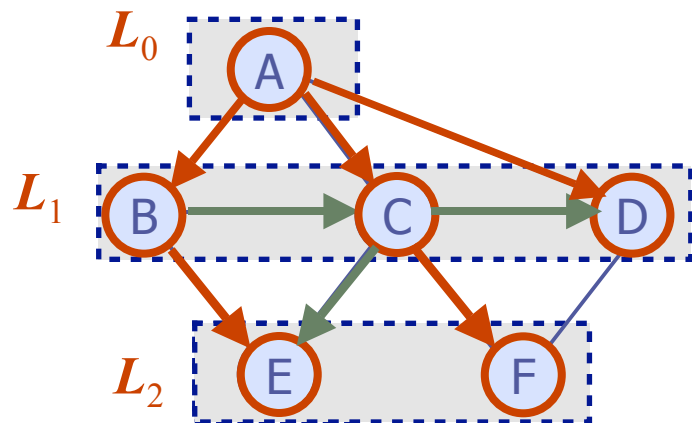
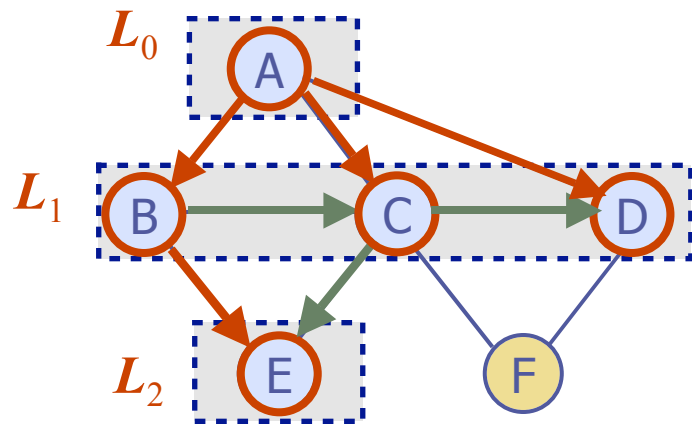
-  Sommets non-explorés
-  Sommets visités
-  Arêtes non-explorées
-  Arêtes sélectionnées
-  Arêtes de retour



Parcours en largeur: Exemple



Parcours en largeur: Exemple

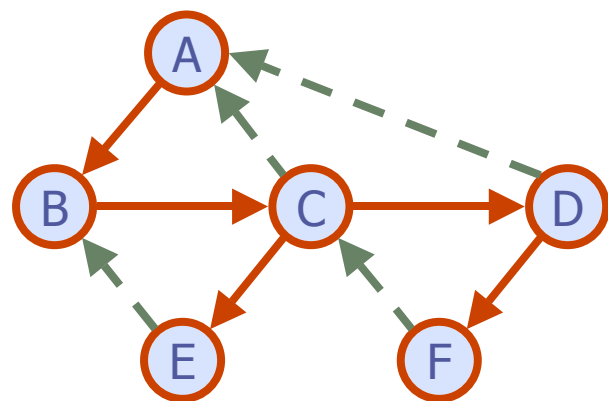


Analyse du parcours en largeur

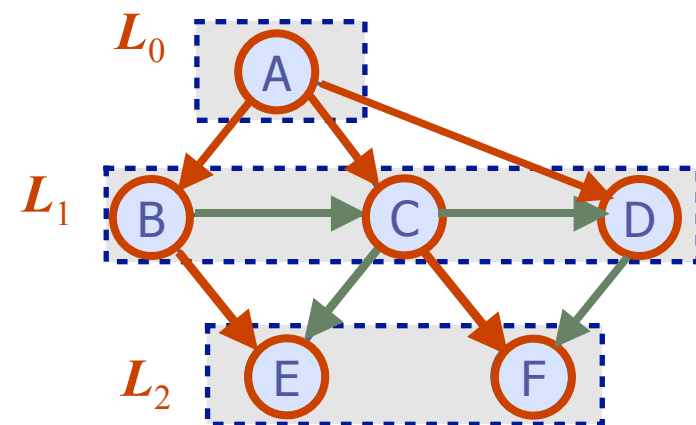
- Chaque sommet est étiqueté deux fois
 - ▣ Une fois “non-exploré”
 - ▣ Une fois “visité”
- Chaque arête est étiquetée deux fois
 - ▣ Une fois “non-explorée”
 - ▣ Une fois “sélectionnée” ou “de traverse”
- Chaque sommet est inséré une fois dans une séquence L_i
- L'opération *incidentes(s)* est appelée pour chaque sommet
- La complexité en temps du parcours en profondeur est en $O(n+m)$ si on utilise la structure de données “liste d'adjacence”

Profondeur versus largeur

Applications	Parcours en profondeur	Parcours en largeur
Forêt couvrante, composantes connexes, chemins, cycles	✓	✓
Plus court chemin		✓



Parcours en profondeur

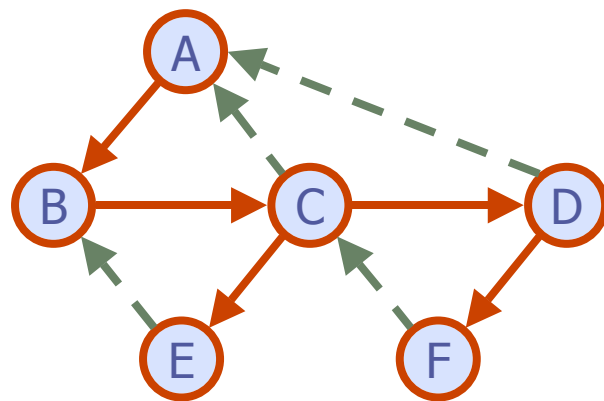


Parcours en largeur

Profondeur versus largeur (suite)

Arête de retour (v, w)

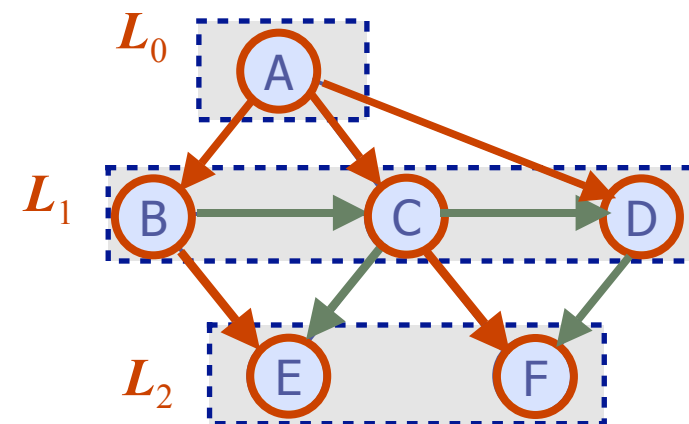
- w est un ancêtre de v dans l'arbre des arêtes sélectionnées



Parcours en profondeur

Arête de traverse (v, w)

- w est sur le même niveau que v ou dans le prochain niveau dans l'arbre des arêtes sélectionnées



Parcours en largeur