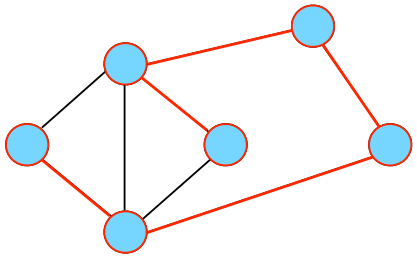
Parcours de graphes

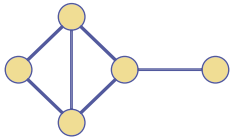
Quelques définitions

- 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 un sous-ensemble des arêtes de G
  - Un sous-graphe est dit couvrant (spanning) s'il contient tous les sommets de G

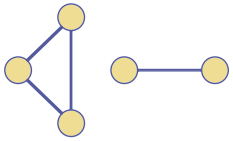


Quelques définitions (suite)

- Un graphe G est dit connexe s'il existe un chemin reliant chaque pair de sommets de G

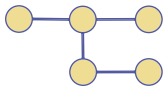


- Une composante connexe d'un graphe G est un sous-graphe connexe maximal de G

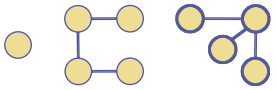


Quelques définitions (suite)

- Un arbre A (non raciné) est un graphe non orienté tel que
  - A est connexe
  - A ne contient pas de cycles

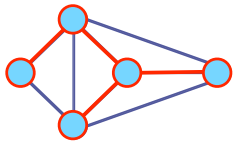


- Une forêt est un graphe non orienté ne contenant pas de cycles
  - Les composantes connexes d'une forêt sont donc des arbres



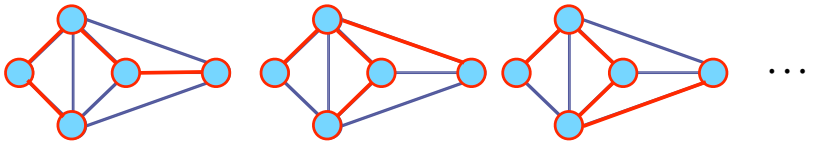
Quelques définitions (suite)

- Un arbre couvrant pour un graphe connexe G est un sous-graphe couvrant qui est un arbre

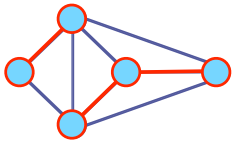


© Goodrich et Tamassia 2004

- Un arbre couvrant pour un graphe G n'est pas unique sauf si G est une arbre



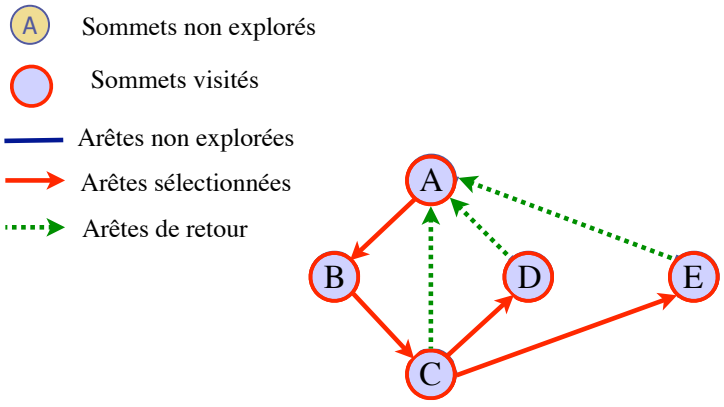
- Une forêt couvrante pour un graphe G est un sous-graphe couvrant qui est une forêt



Parcours en profondeur (Depth-First Search)

- Un parcours en profondeur (DFS) 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
- L'algorithme de parcours en profondeur (DFS) d'un graphe G prend un temps $O(n+m)$
- L'algorithme de parcours en profondeur peut être étendu pour résoudre d'autres problèmes sur les graphes:
  - Trouver un chemin entre 2 sommets
  - Trouver un cycle dans un graphe

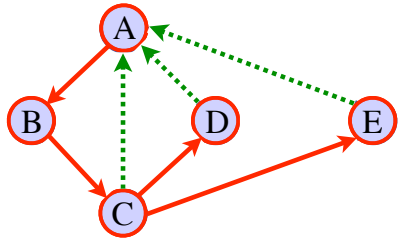
Exemple:



© adapté de Goodrich et Tamassia 2004

Propriétés du parcours en profondeur:

- Propriété 1: DFS(G,s) visite tous les sommets et les arêtes de la composante connexe de s



- Propriété 2: Les arêtes sélectionnées lors du parcours DFS(G,s) forme un arbre couvrant pour la composant connexe de s

Complexité en temps du parcours en profondeur:

- Étiquetter ou "lire" l'étiquette d'un sommet ou d'une arête $O(1)$
- Chaque sommet est étiquetté deux fois } $O(n)$
  - une fois "non exploré"
  - une fois "visité"
- Chaque arête est étiquettée deux fois } $O(m)$
  - une fois "non explorée"
  - une fois "sélectionnée" ou "de retour"
- L'opération Incidents(u) est appelée une fois pour chaque sommet u
- Si notre graphe est représenté par une liste d'adjacences, la complexité en temps de l'algorithme DFS est $O(m+n)$

Algorithme de recherche de chemins

- On peut étendre l'algorithme DFS en un algorithme pour trouver un chemin entre 2 sommets donnés u et z
- L'idée est d'appeler DFS(G,u), sur u le premier sommet
- On utilise une pile P qui garde en mémoire un chemin entre le sommet de départ et le sommet courant
- Quand le sommet final z est atteint on retourne le contenu de la pile qui contient le chemin cherché

```
Algorithme cheminDFS(G, v, z)
  setÉtiquette(v, VISITÉ)
  P.empiler(v)
  si v = z
    retourner P.éléments()
  Pour tout e ∈ G.incidents(v)
   si étiquette(e) = NON EXPLORÉE
      w ← opposé(v,e)
      si étiquette(w) = NON EXPLORÉ
         setÉtiquette(e, SÉLECTIONNÉE)
         P.empiler(e)
         cheminDFS(G, w, z)
         P.dépiler()
      sinon
         setÉtiquette(e, DE RETOUR)
  P.dépiler()
```


Algorithme de recherche de cycles

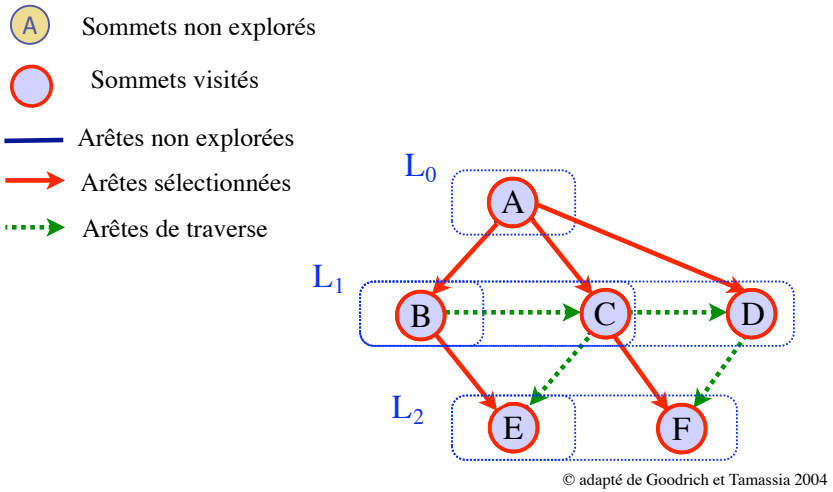
- On peut étendre l'algorithme DFS en un algorithme pour trouver un cycle dans un graphe (s'il en existe un)
- On utilise une pile P qui garde en mémoire un chemin entre le sommet de départ v et le sommet courant
- Si on trouve une arête de retour vers v, on retourne le cycle trouvé qui est contenu dans la pile

```
Algorithm cycleDFS(G, v, z)
  setÉtiquette(v, VISITÉ)
  P.empiler(v)
  Pour tout e ∈ G.incidents(v)
    si Étiquette(e) = NON EXPLORÉE
       w ← opposé(v,e)
       P.empiler(e)
       si Étiquette(w) = NON EXPLORÉ
          setÉtiquette(e, SÉLECTIONNÉE)
          cheminDFS(G, w, z)
          P.dépiler()
       sinon
          T ← nouvelle pile vide
          répéter
             o ← P.dépiler()
             T.empiler(o)
          tant que o = w
          retourner T.éléments()
  P.dépiler()
```


Parcours en largeur (Breadth-First Search)

- Un parcours en largeur (BFS) 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
- L'algorithme de parcours en largeur (BFS) d'un graphe G prend un temps $O(n+m)$
- L'algorithme de parcours en largeur peut être étendu pour résoudre d'autres problèmes sur les graphes:
  - Trouver le plus court chemin entre 2 sommets
  - Trouver un cycle simple dans un graphe

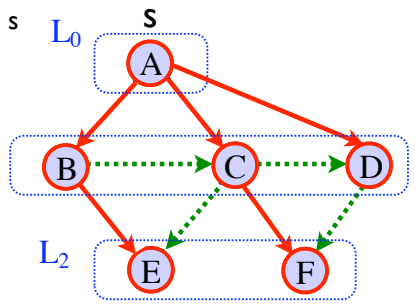
Exemple:



© adapté de Goodrich et Tamassia 2004

Propriétés du parcours en largeur:

- Propriété 1: BFS(G,s) visite tous les sommets et les arêtes de la composante connexe de s
- Propriété 2: Les arêtes sélectionnées lors du parcours DFS(G,s) forme un arbre couvrant pour la composant connexe de s
- Propriété 3: Pour tous sommets u dans L_i, le chemin de s à u suivant les arêtes de l'arbre couvrant contient exactement i arêtes et i+1 sommets. C'est un plus court chemin de s à u.

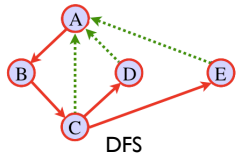


Complexité en temps du parcours en largeur:

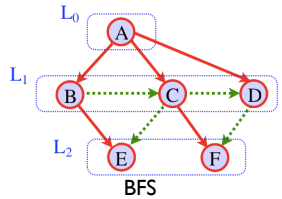
- Étiquetter ou "lire" l'étiquette d'un sommet ou d'une arête $O(1)$
- Chaque sommet est étiquetté deux fois } $O(n)$
  - une fois "non exploré"
  - une fois "visité"
- Chaque arête est étiquettée deux fois } $O(m)$
  - une fois "non explorée"
  - une fois "sélectionnée" ou "de traverse"
- Chaque arête est insérée au plus une fois dans un ensemble L_i
- L'opération Incidents(u) est appelée une fois pour chaque sommet u
- Si notre graphe est représenté par une liste d'adjacences, la complexité en temps de l'algorithme DFS est $O(m+n)$

DFS vs BFS

Applications	DFS	BFS
fôret couvrante, composantes connexes	ok	ok
chemins entre deux sommets, cycles	ok	ok
plus court chemin		ok



DFS



BFS

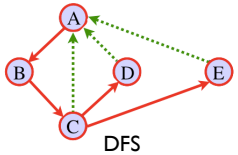
DFS vs BFS (suite)

arête de retour(v,w):

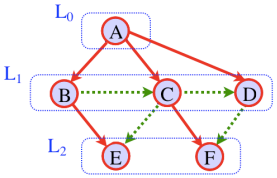
- Dans l'arbre couvrant représenté par les arêtes sélectionnées (en rouge) w est un ancêtre de v

arête de traverse(v,w):

- w est soit au même niveau que v soit au niveau inférieur dans l'arbre couvrant



DFS



BFS