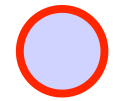
Parcours de graphes

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:

Sommets non visités

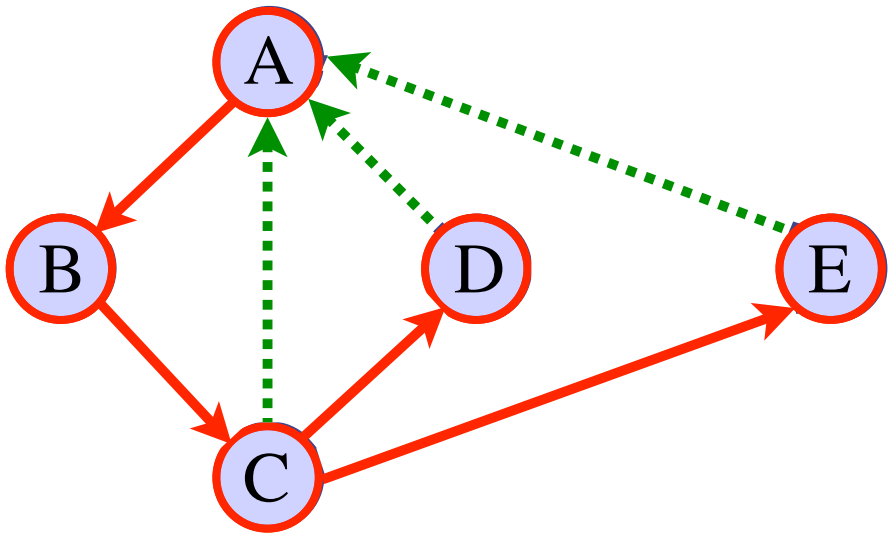


Sommets visités

Arêtes non visitées

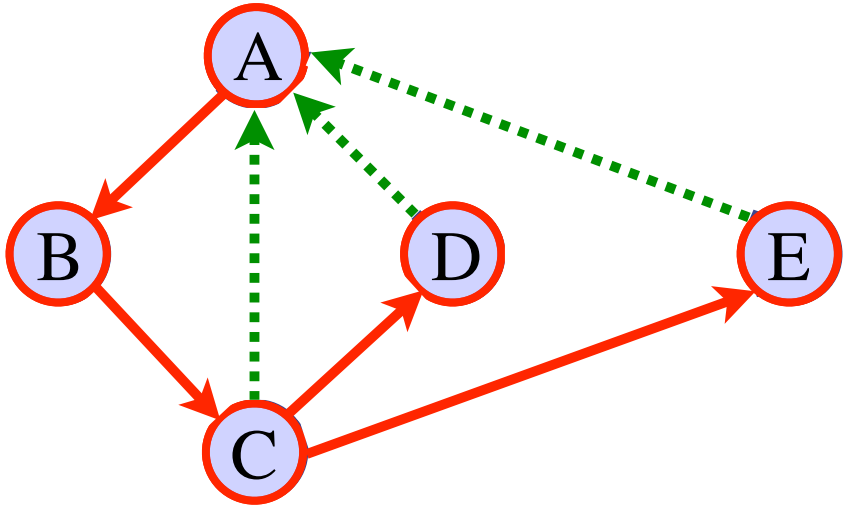
Arêtes sélectionnées

Arêtes de retour



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
  - une fois "non visité"
  - une fois "visité"

  } $O(n)$
- L'opération Adjacents(u) est appelée une fois pour chaque sommet u
- Si notre graphe est représenté par une liste d'adjacences, comme on a que

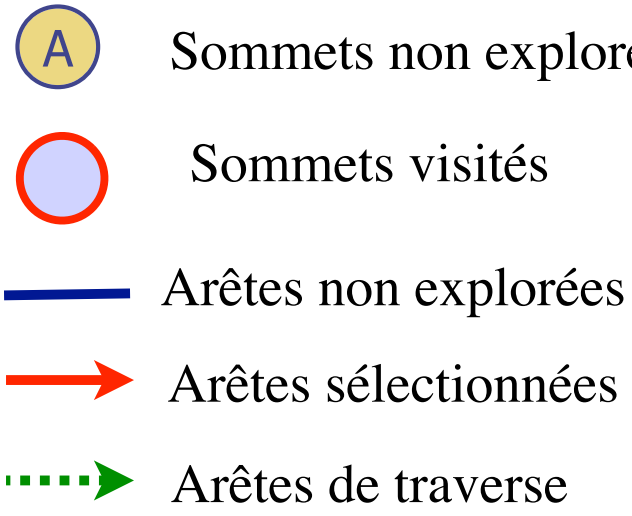
$$\sum_{v \in N} deg(v) = 2m$$

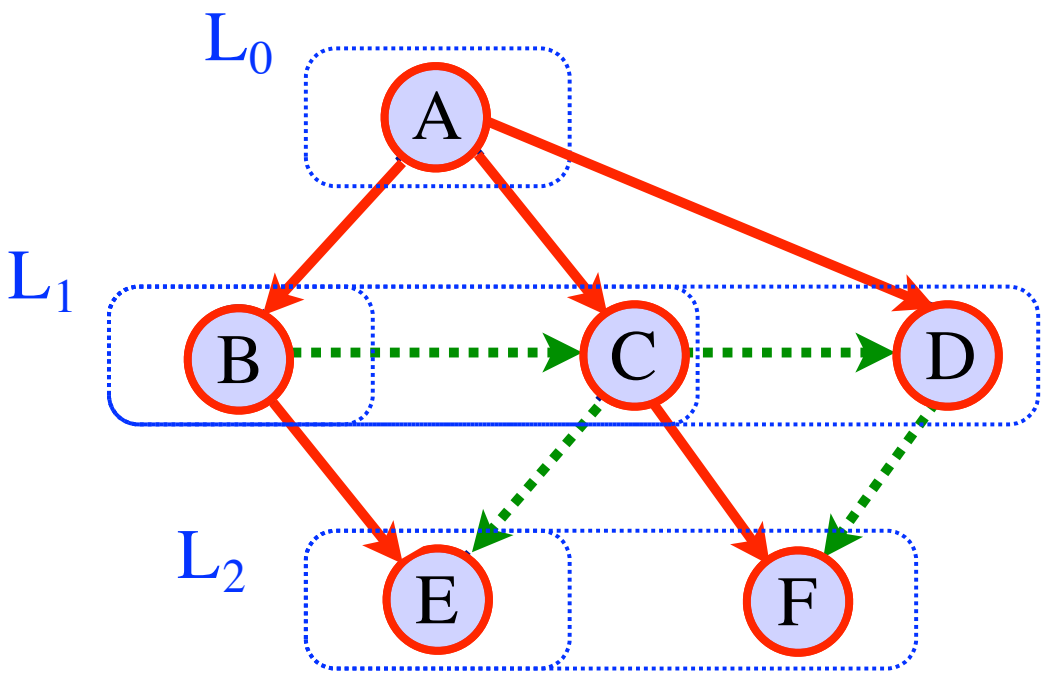
la complexité en temps de l'algorithme DFS est $O(m+n)$

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

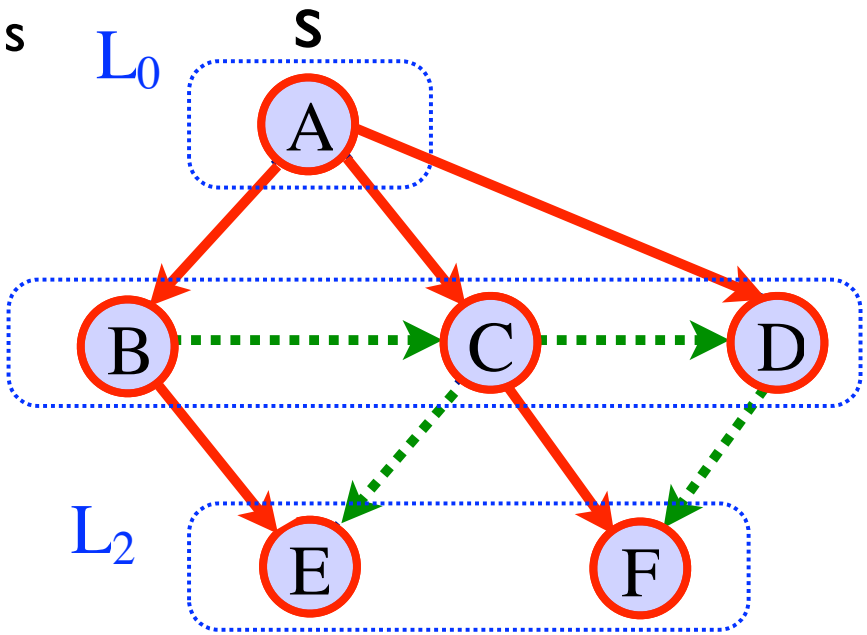
Exemple:





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.



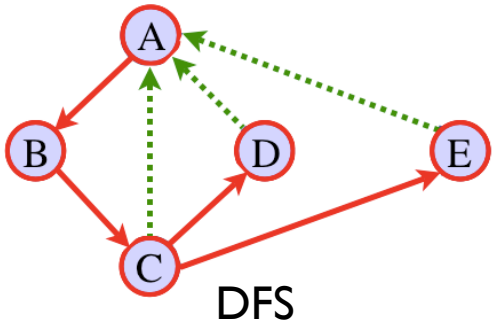
Complexité en temps du parcours en largeur:

- Étiquetter ou "lire" l'étiquette d'un sommet ou d'une arête $O(1)$
- Chaque sommet est étiquetté deux fois
  - une fois "non exploré"
  - une fois "visité"

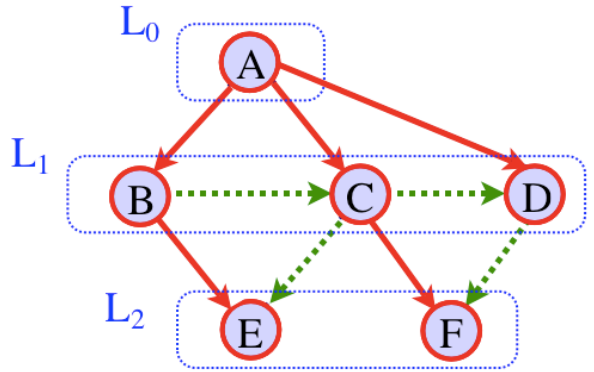
  } $O(n)$
- L'opération Adjacents(u) est appelée une fois pour chaque sommet u
- Si notre graphe est représenté par une liste d'adjacences, la complexité en temps de l'algorithme DFS est $O(m+n)$

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