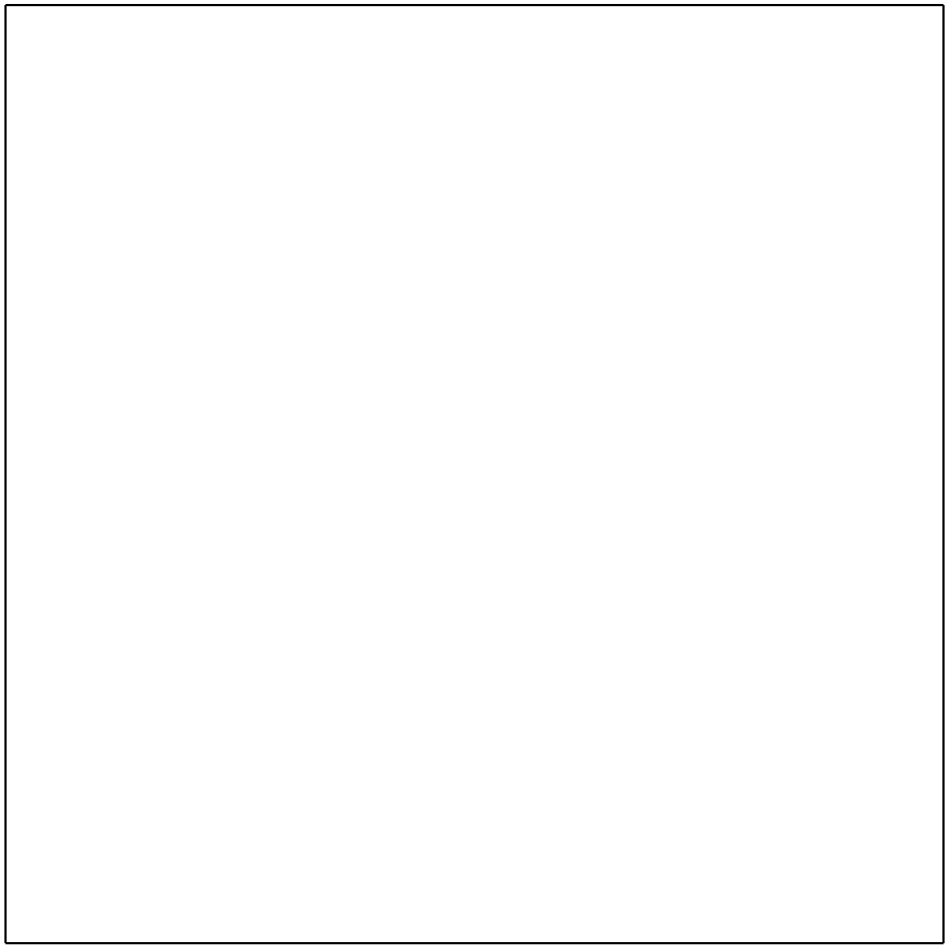
Un exemple de structure de données : les arbres

Céline Caldini Queiros

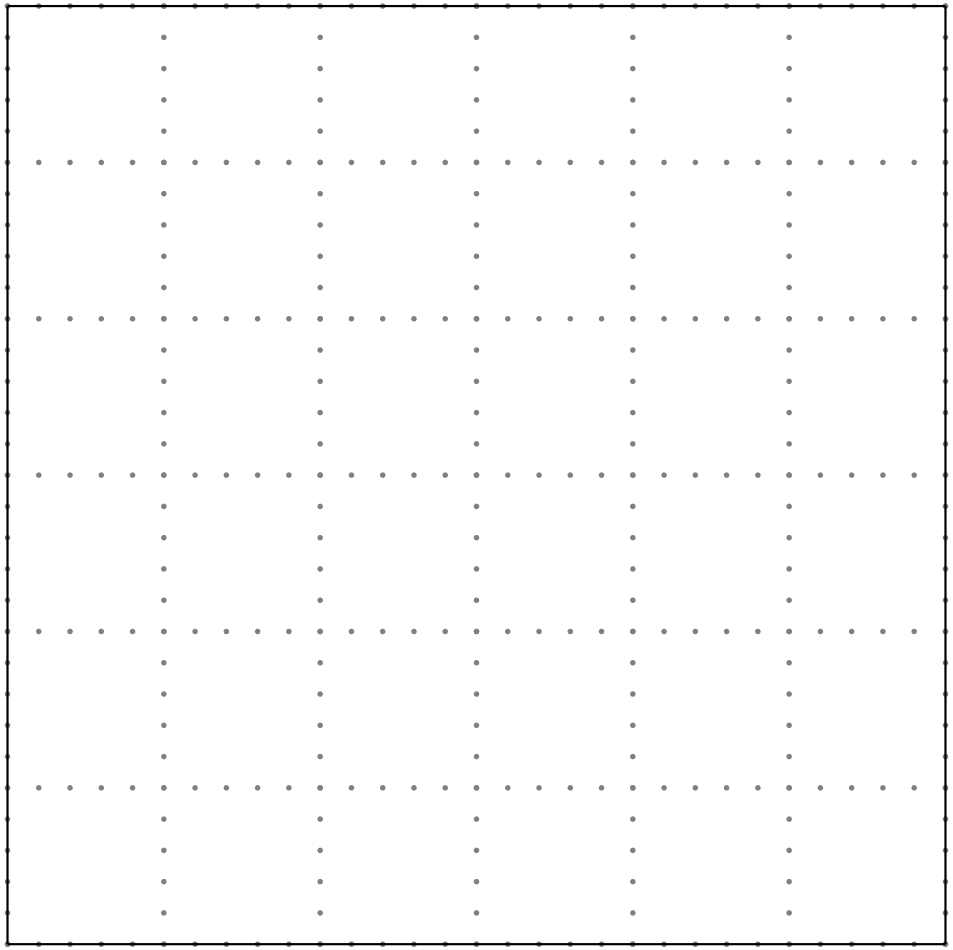
Laboratoire de Mathématiques de Besançon

$9\ mars\ 2013$

Présentation



Présentation



Aspects de la programmation

- traitement
- données

Aspects de la programmation

- traitement
- données

Algorithmique $\neq$ Structure de données

- Algorithme :
- Structure de données :

Aspects de la programmation

- traitement
- données

Algorithmique $\neq$ Structure de données

- Algorithme : suite d'opération permettant de résoudre un problème
- Structure de données :

Aspects de la programmation

- traitement
- données

Algorithmique $\neq$ Structure de données

- Algorithme : suite d'opération permettant de résoudre un problème
- Structure de données : mode d'organisation des données pour faciliter leurs traitment

Paradigmes de la programmation

- Calcul scientifique (données simples, traitements composés d'opérations arithmétiques et de boucles)

Paradigmes de la programmation

- Calcul scientifique (données simples, traitements composés d'opérations arithmétiques et de boucles)
- Programmation fonctionnelle (données : termes symboliques, transformation syntaxiques de termes)

Paradigmes de la programmation

- Calcul scientifique (données simples, traitements composés d'opérations arithmétiques et de boucles)
- Programmation fonctionnelle (données : termes symboliques, transformation syntaxiques de termes)
- Programmation logique (termes symboliques et formules logiques, traitements toujours les mêmes)

Paradigmes de la programmation

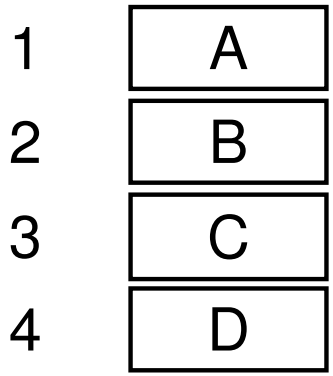
- Calcul scientifique (données simples, traitements composés d'opérations arithmétiques et de boucles)
- Programmation fonctionnelle (données : termes symboliques, transformation syntaxiques de termes)
- Programmation logique (termes symboliques et formules logiques, traitements toujours les mêmes)
- Programmation procédurale (visée générale : description explicite des données et des traitements)

Paradigmes de la programmation

- Calcul scientifique (données simples, traitements composés d'opérations arithmétiques et de boucles)
- Programmation fonctionnelle (données : termes symboliques, transformation syntaxiques de termes)
- Programmation logique (termes symboliques et formules logiques, traitements toujours les mêmes)
- Programmation procédurale (visée générale : description explicite des données et des traitements)
- Programmation orientée objet (évolution de la programmation procédurale qui organise un programme en unités relativement indépendantes)

Un peu de récursivité

On prend la liste

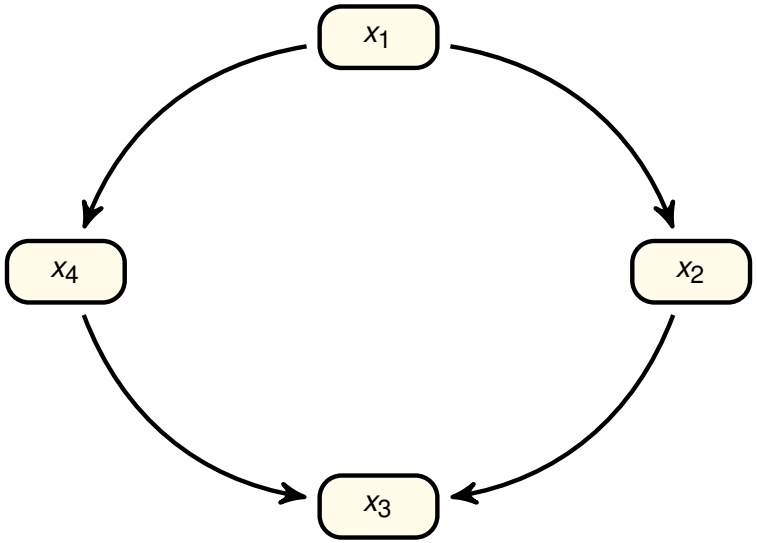


Un peu de récursivité

Pour afficher la liste de manière récursive on fait :

L : Liste	`AffichageRécursif(L)`
[]	
[x,L']	`affichage(x)` `AffichageRécursif(L')`

Qu'est ce qu'un graphe



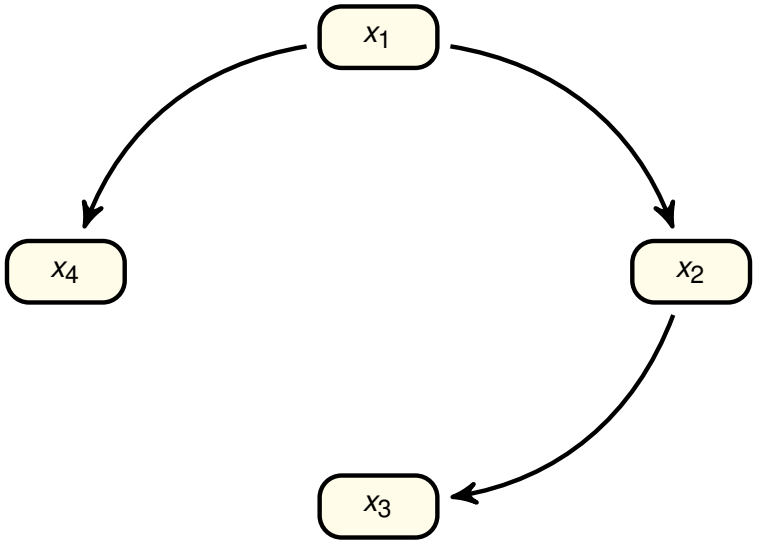
Arborescence

Définition

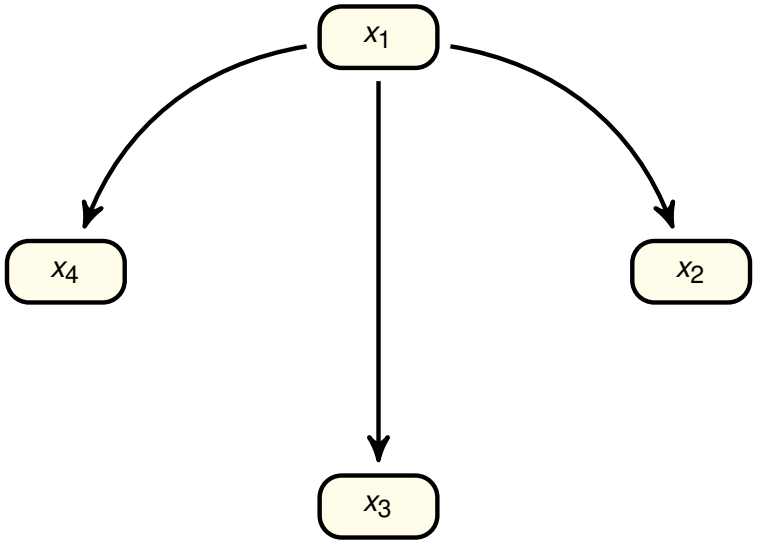
Une arborescence de racine r est un graphe orienté tel que

- Tout sommet $a \neq r$ a un seul prédecesseur, r n'en a aucun
- Tout sommet s est descendant de r

Arborescence

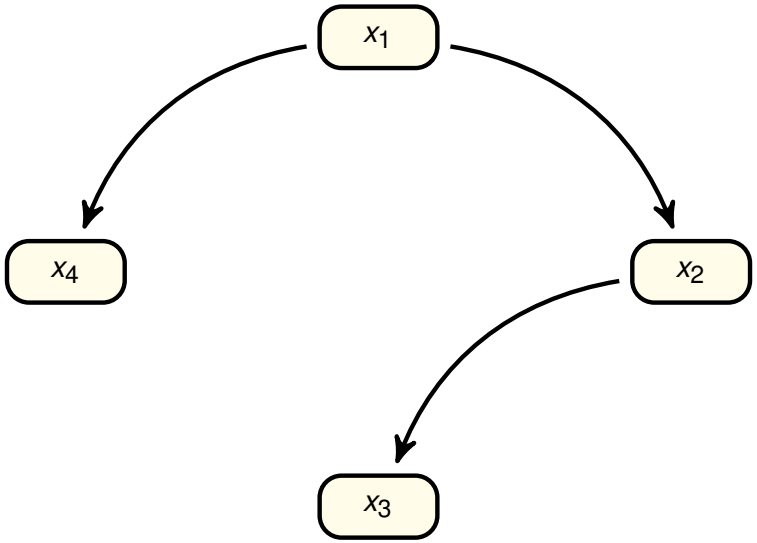


Arborescence

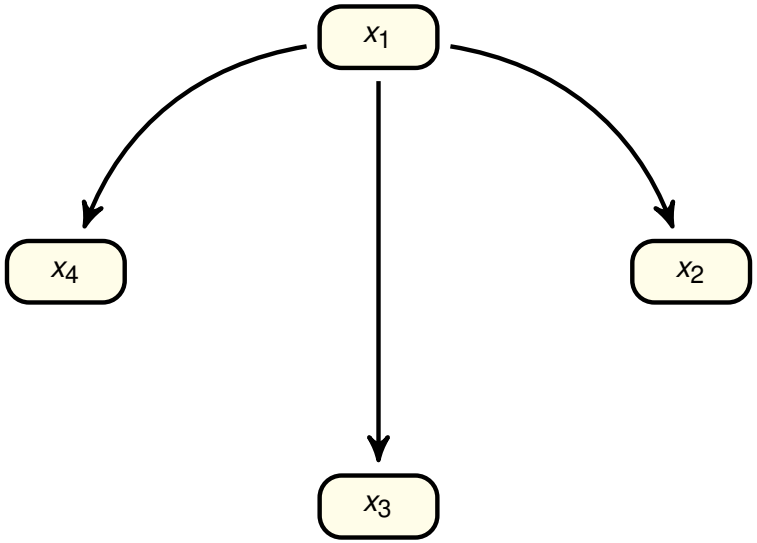


Arbre binaire

Chaque noeud a *au plus* 2 fils.



Arbre binaire



Terminologie

- **hauteur**(n) : max des longueurs allant de n vers une feuille

Arbre

Terminologie

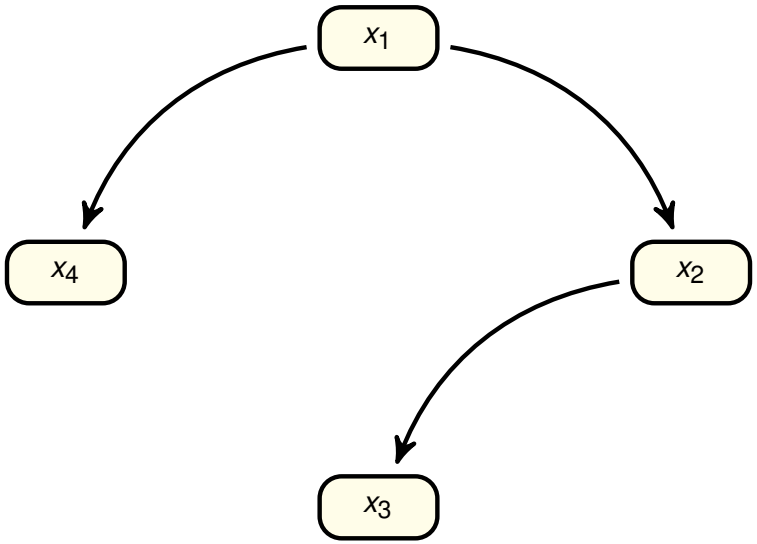
- **hauteur**(n) : max des longueurs allant de n vers une feuille
- **profondeur**(n) : longueur du chemin allant de la racine à n

Arbre

Terminologie

- **hauteur**(n) : max des longueurs allant de n vers une feuille
- **profondeur**(n) : longueur du chemin allant de la racine à n
- **profondeur de l'arborescence** : max des profondeurs de ses fils
  = hauteur de l'arbre

Exemple

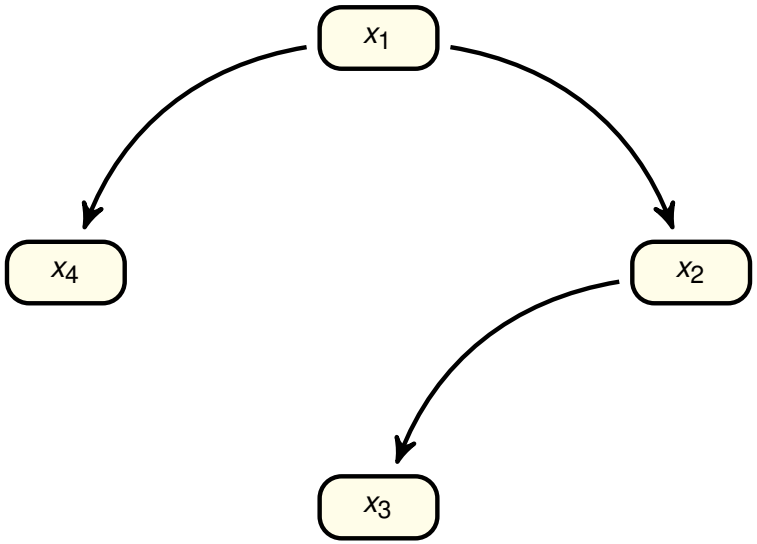


Exemple : arbre complet

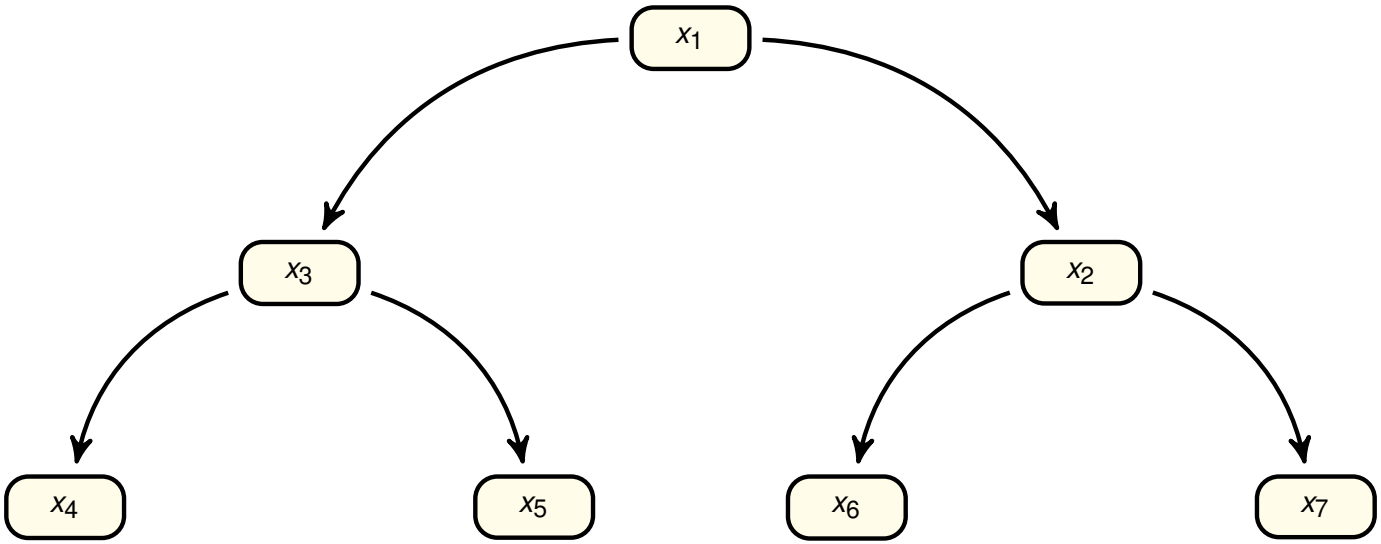
Définition

- Non vide.
- À la profondeur i, il y a 2^i nœuds

Arbre non complet



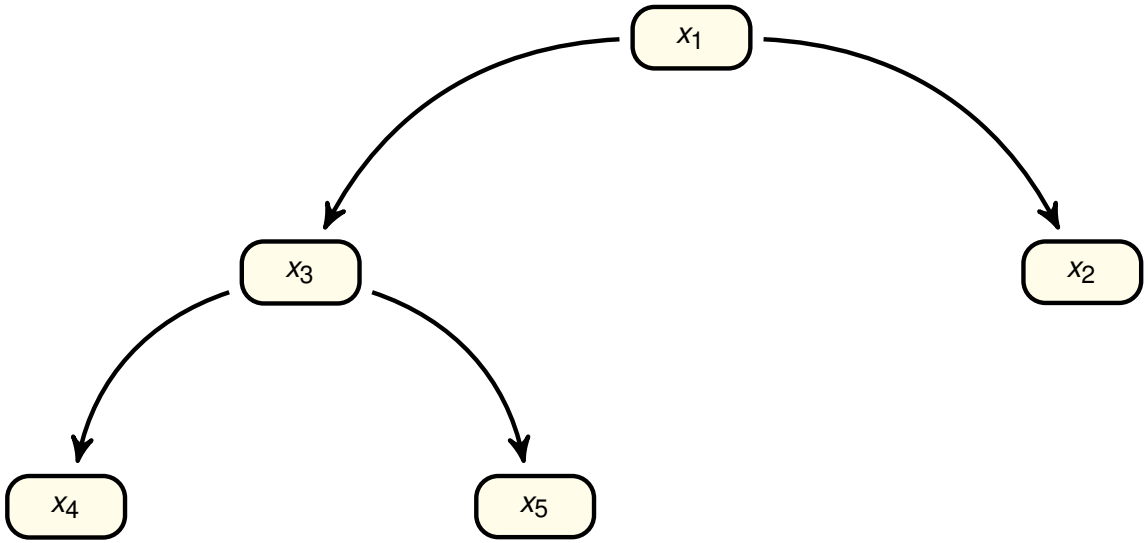
Arbre complet



Exemple : arbre localement complet

Définition

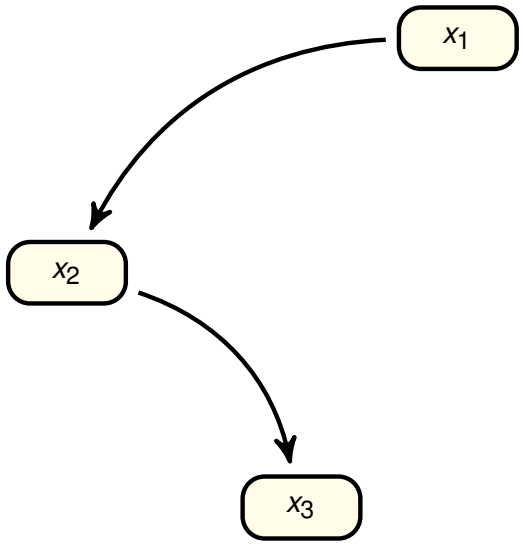
- Non vide.
- Sans nœud unaire



Exemple : arbre dégénéré

Définition

- Non vide.
- Sans nœud binaire

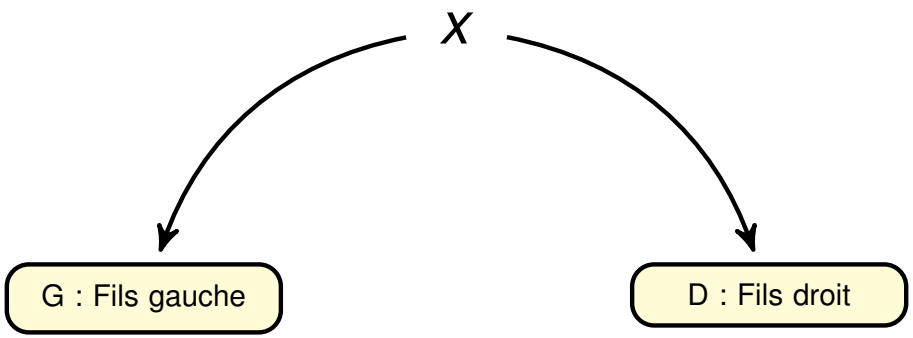


Parcours en profondeur

Totalement récursif, les parcours en profondeurs respectent la grammaire de la structure

Exemple de parcours en profondeur

Dans la suite, on écrit [G,x,D] notre arbre

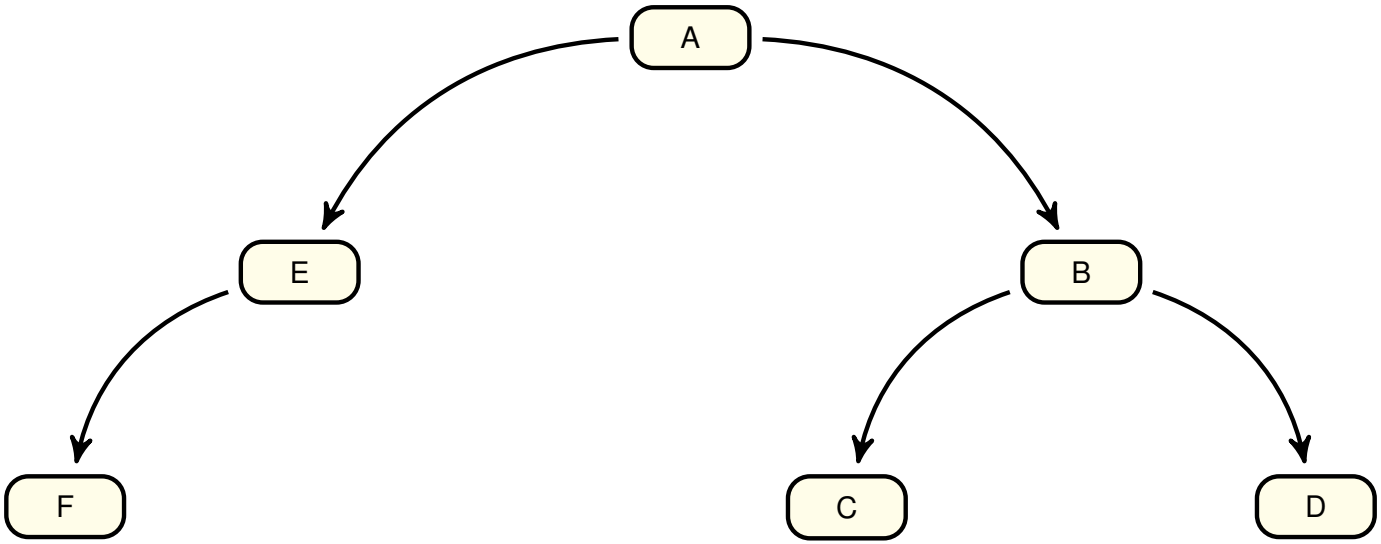


Exemple de parcours en profondeur

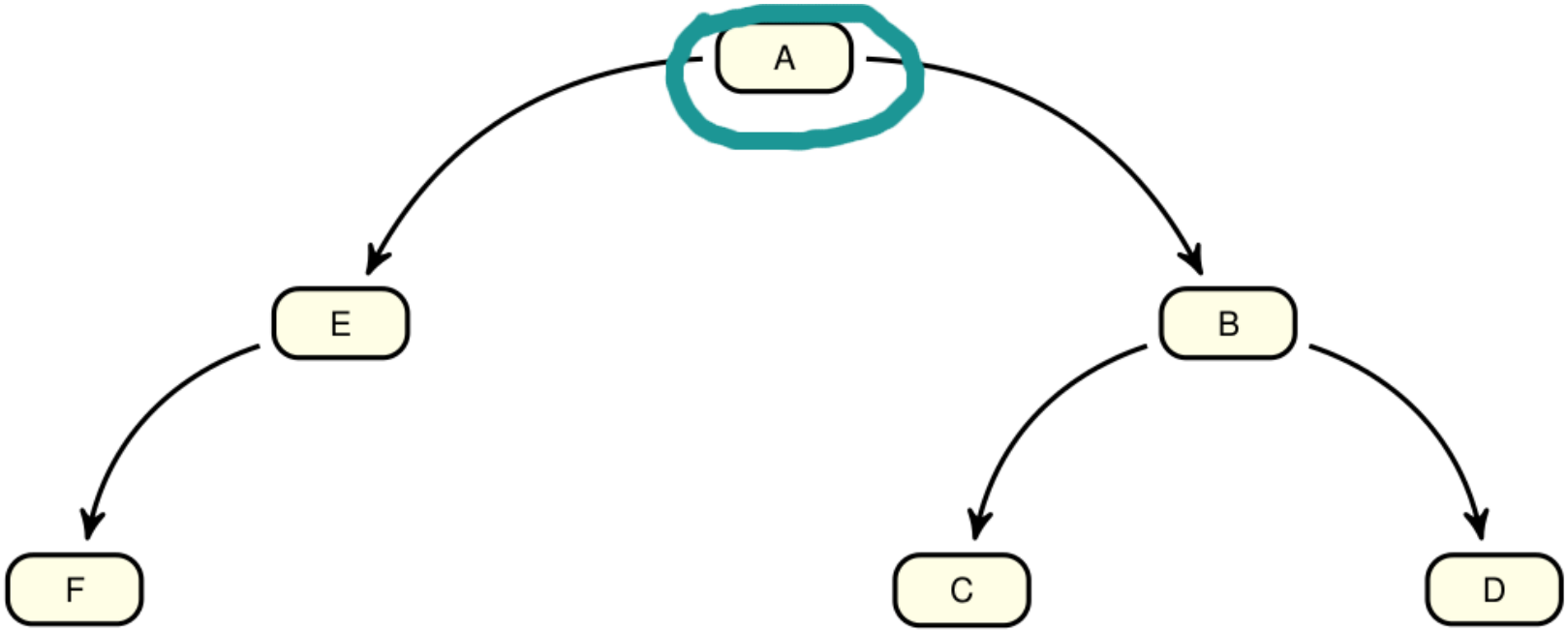
A=[G,x,D] : arbre	`Parcours(A)`
[]	

[G,x,D]	`visiter(x)`
	`Parcours(G)`
	`Parcours(D)`

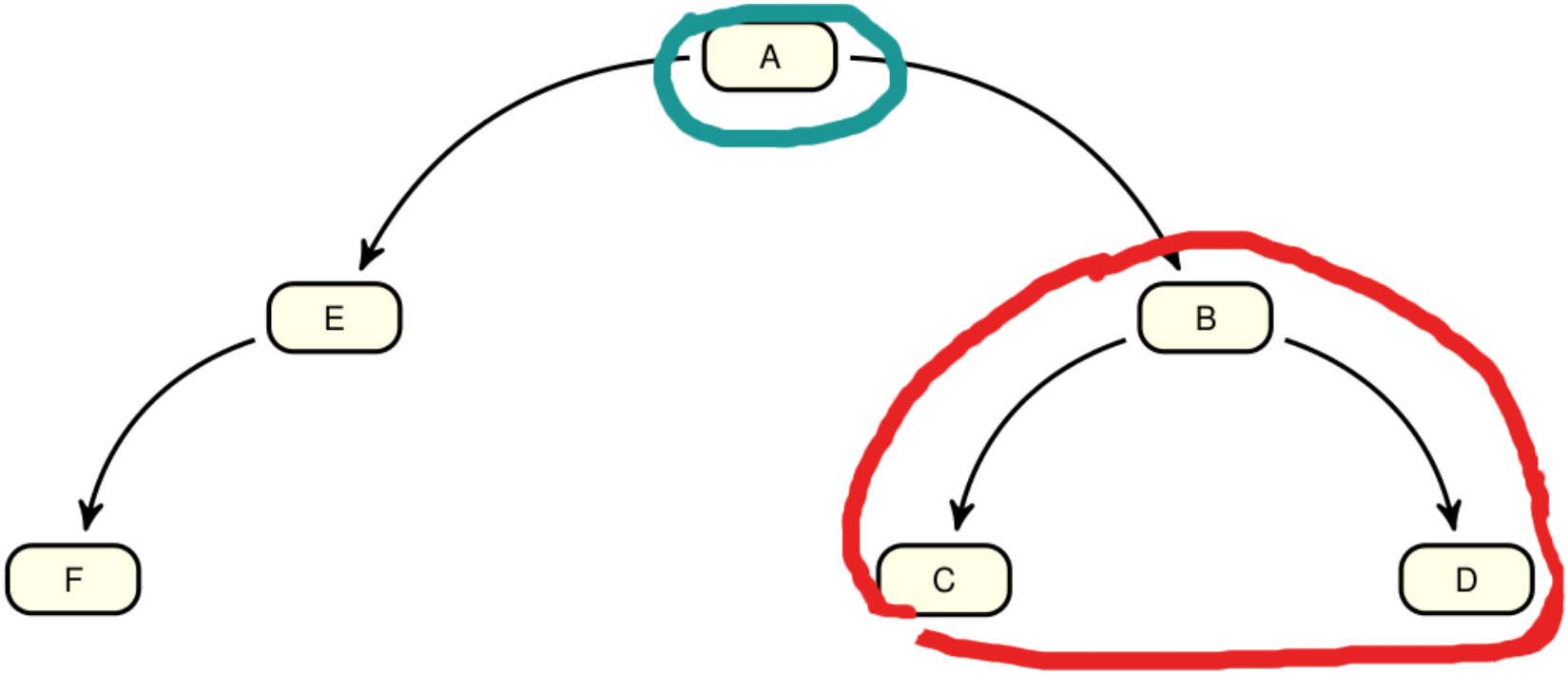
Exemple de parcours en profondeur



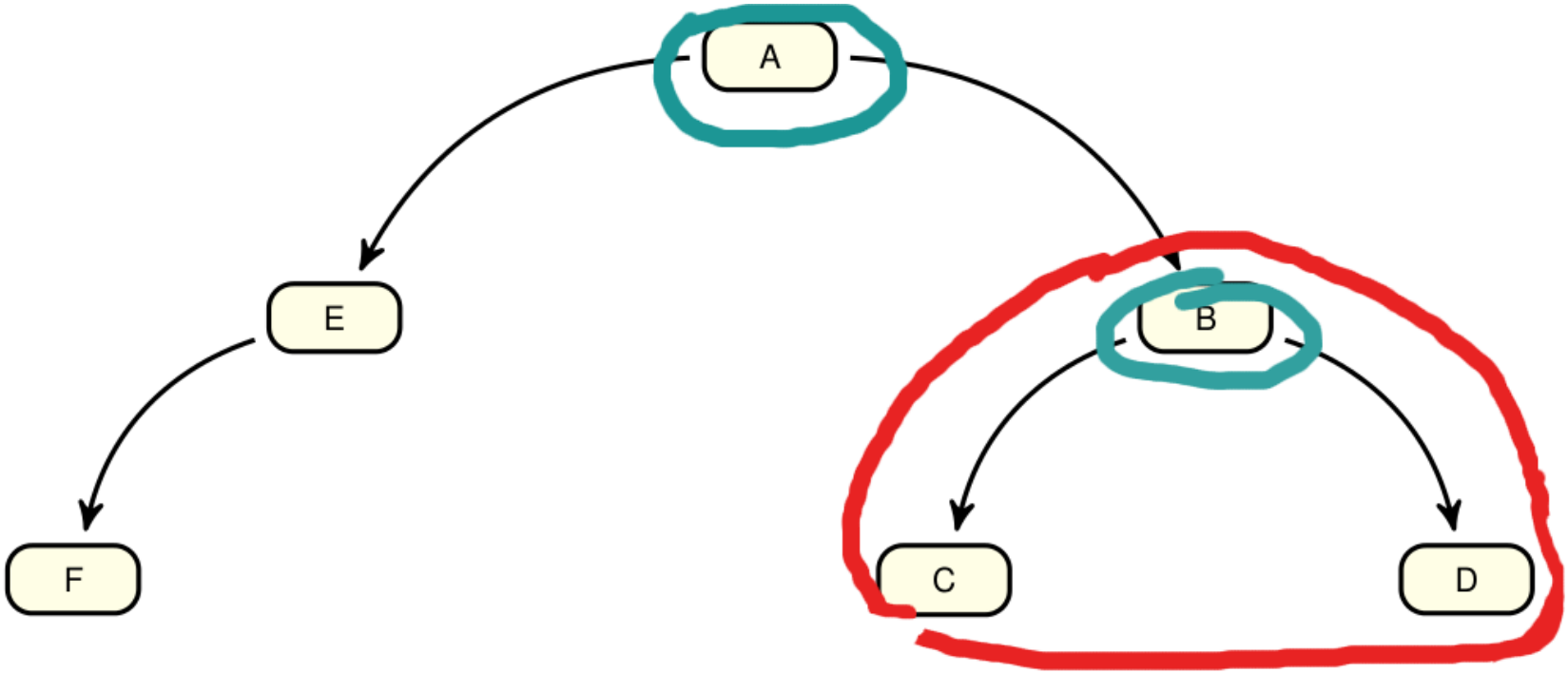
Exemple de parcours en profondeur



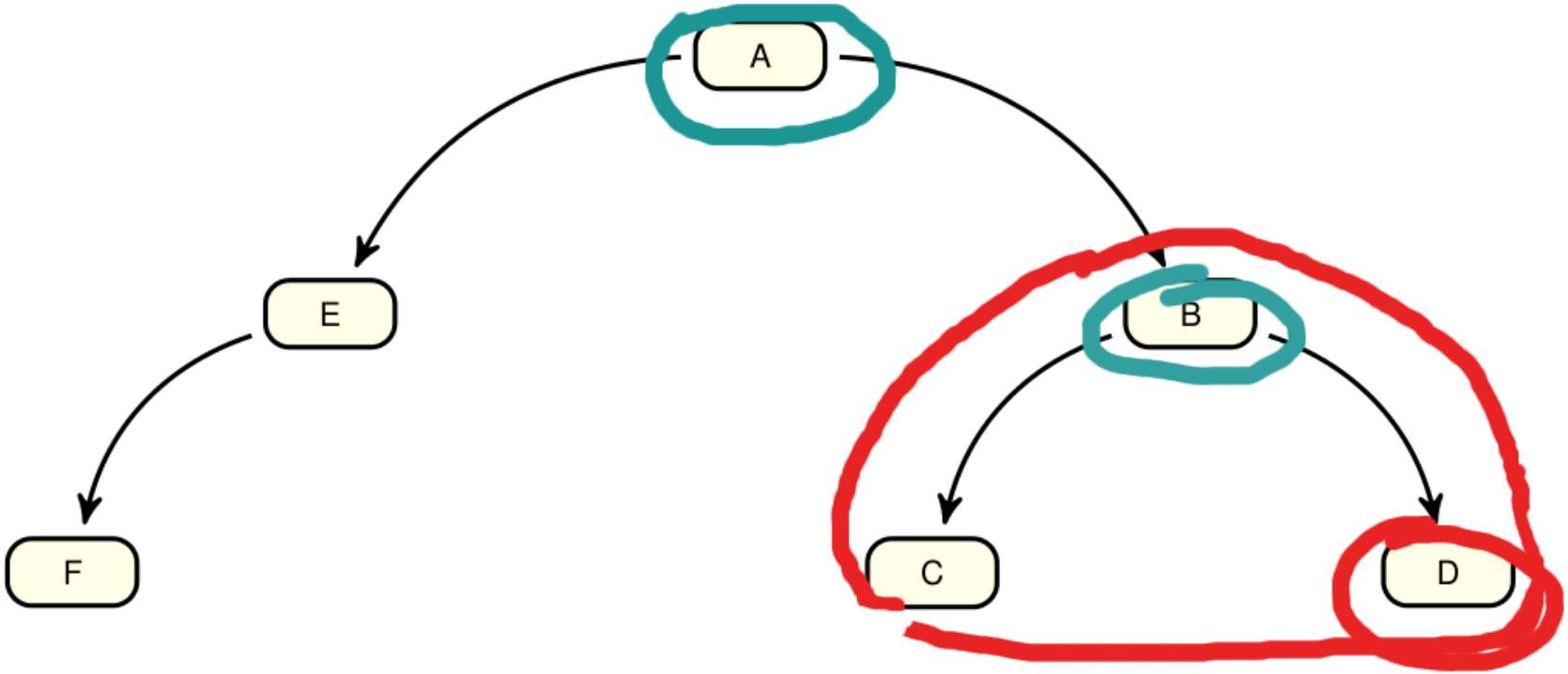
Exemple de parcours en profondeur



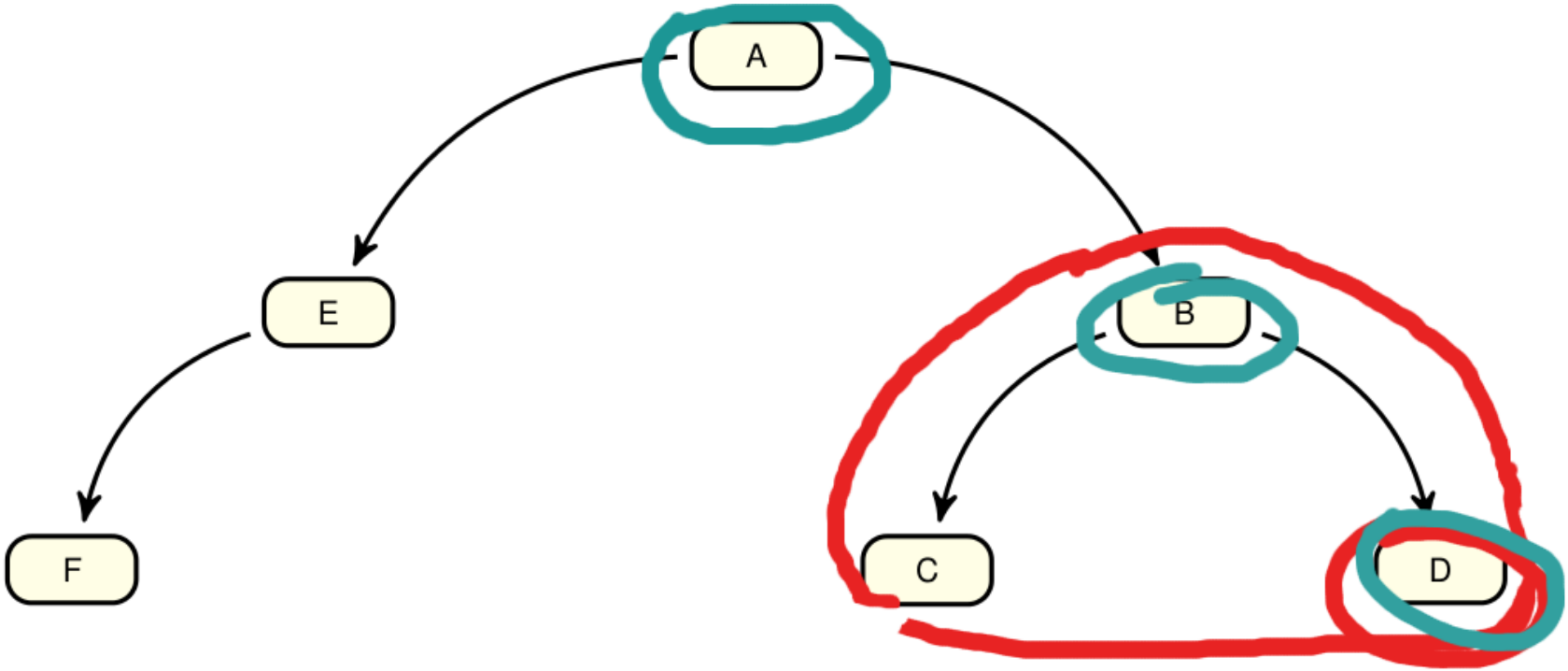
Exemple de parcours en profondeur



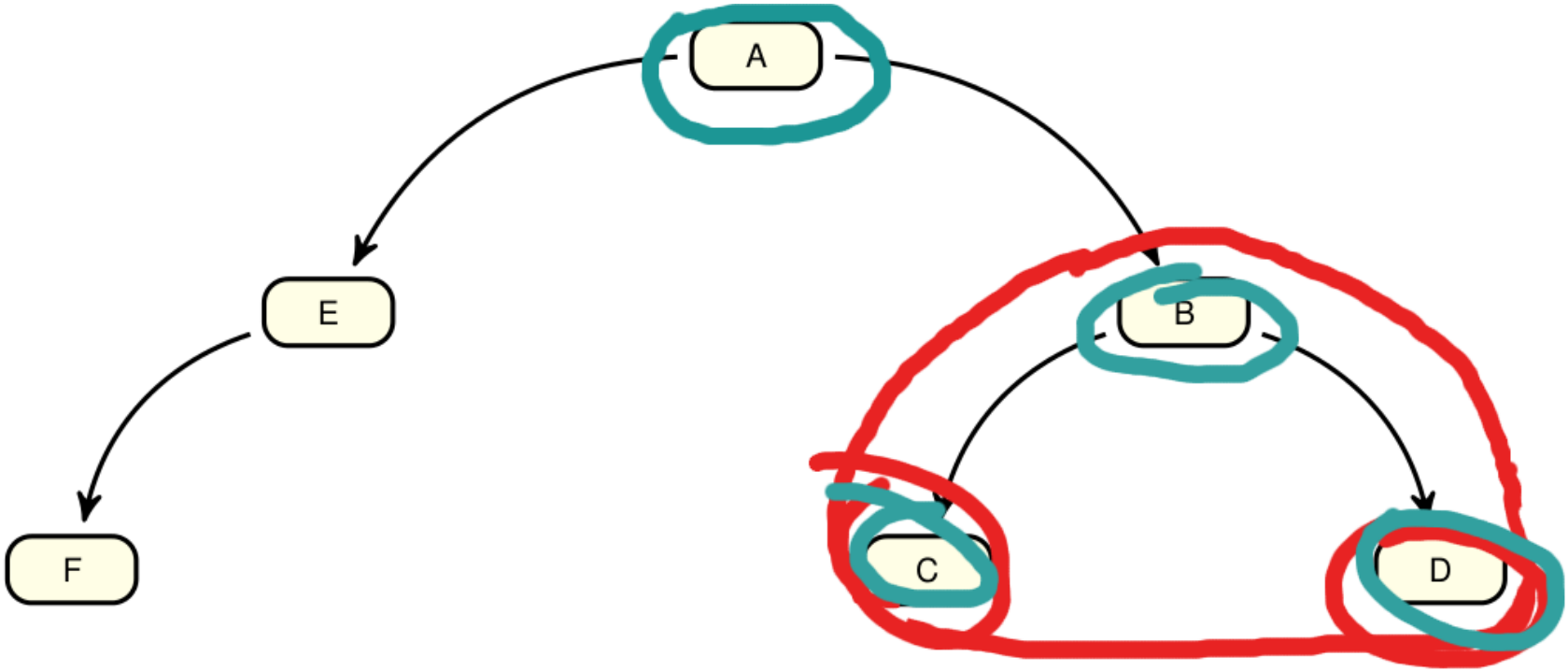
Exemple de parcours en profondeur



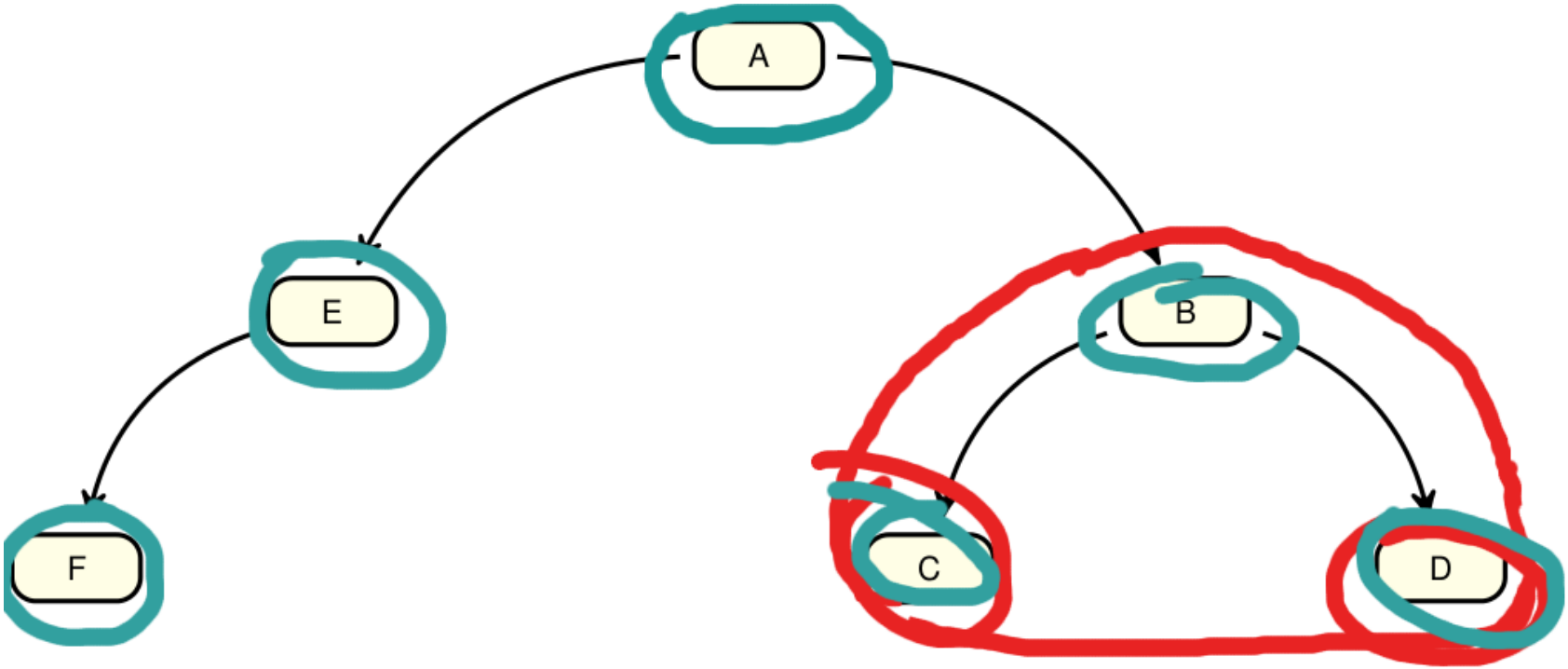
Exemple de parcours en profondeur



Exemple de parcours en profondeur



Exemple de parcours en profondeur



Exemple de parcours en profondeur

A=[G,x,D] : arbre	`Parcours(A)`
[]	

[G,x,D]	`visiter(x)`
	`Parcours(D)`
	`Parcours(G)`

Exemple de parcours en profondeur

A=[G,x,D] : arbre	`ParcoursDG(A)`
[]	

[G,x,D]	`Parcours(D)`
	`visiter(x)`
	`Parcours(G)`

Exemple de parcours en profondeur

A=[G,x,D] : arbre	`ParcourGD(A)`
[]	

[G,x,D]	`Parcours(G)`
	`visiter(x)`
	`Parcours(D)`

Exemple de parcours en profondeur

A=[G,x,D] : arbre	`Parcours(A)`
[]	

[G,x,D]	`Parcours(G)`
	`Parcours(D)`
	`visiter(x)`

Exemple de parcours en profondeur

A=[G,x,D] : arbre	`Parcours(A)`
[]	

[G,x,D]	`Parcours(D)`
	`Parcours(G)`
	`visiter(x)`

Un autre parcours : en largeur

- Non récursif.
- Sur le graphe d'arborescence
- on visite les noeuds par profondeur croissante et à profondeur donnée on les viside de gauche à droite ou de droite à gauche.