Chapitre 3 – arbres

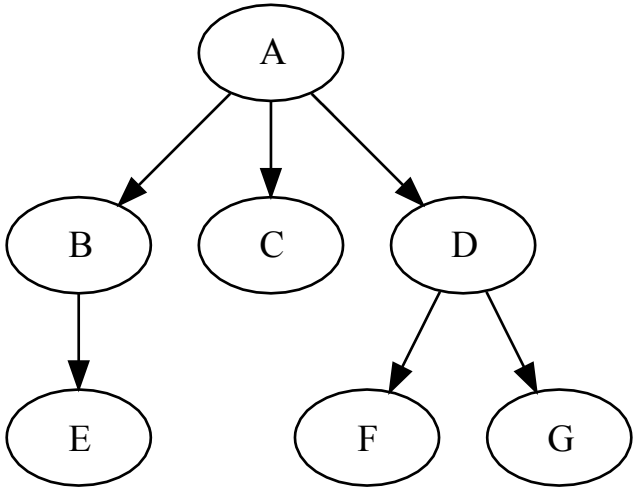
Plan du chapitre

- 1 – Définitions
- 2 – Les parcours d’arbre
- 3 – Les arbres binaires de recherche (ABR)
- 4 – Les B-arbres

3.1 – Définitions

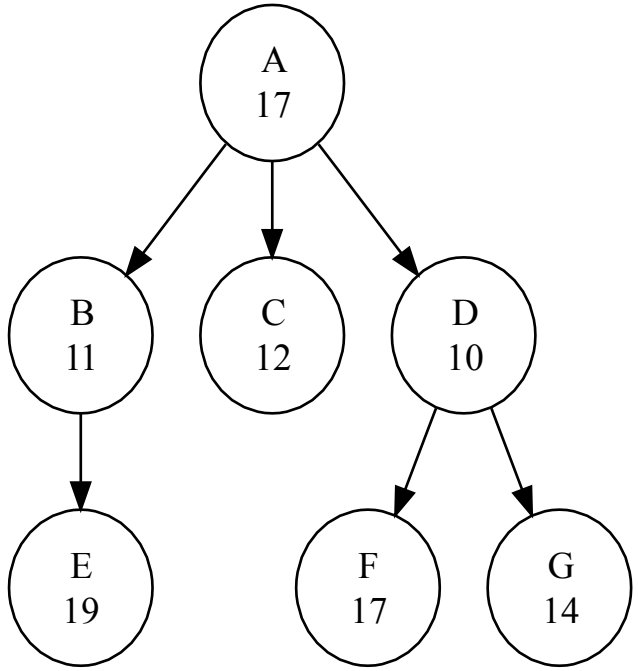
Définitions de base

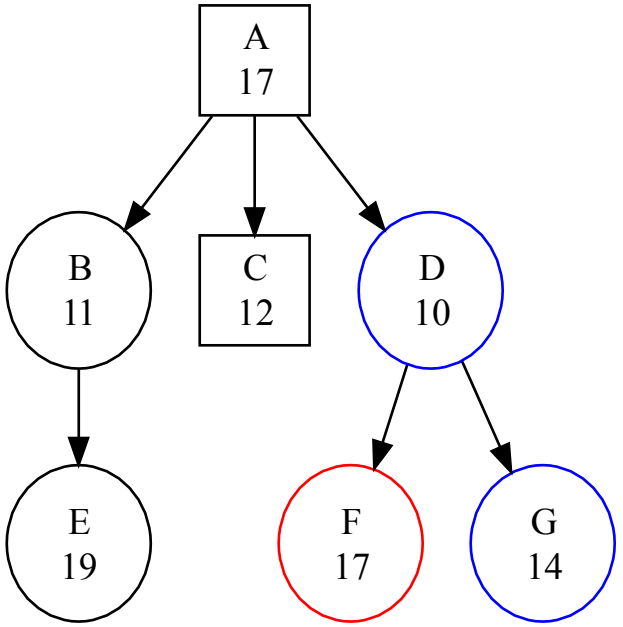
- Un arbre est un ensemble de nœuds dans lequel :
  - Il existe un nœud particulier appelé « racine »
  - Tous les nœuds, sauf la racine, ont un unique père
- Les fils d'un nœud n sont tous les nœuds dont n est le père
- Les feuilles d'un arbre sont les nœuds qui n'ont pas de fils
- Définition de la théorie des graphes :
  graphe connexe sans cycle (orienté ou non)

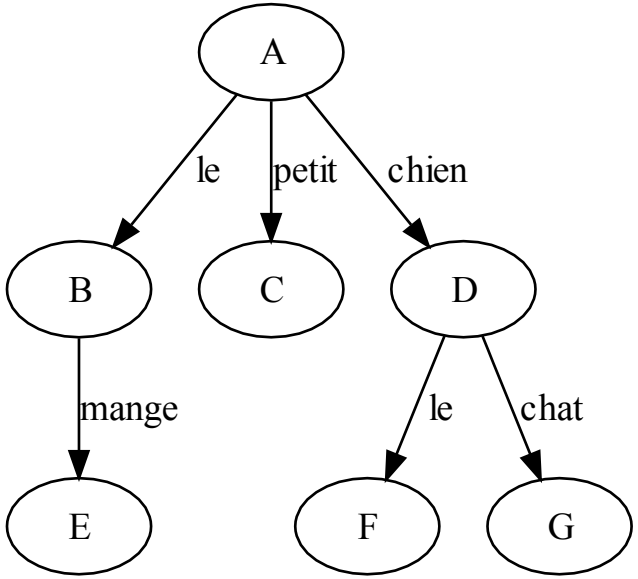


Stockage de valeurs

- Une valeur peut être stockée dans les nœuds ou dans les arcs père-fils (ou dans les nœuds et les arcs)





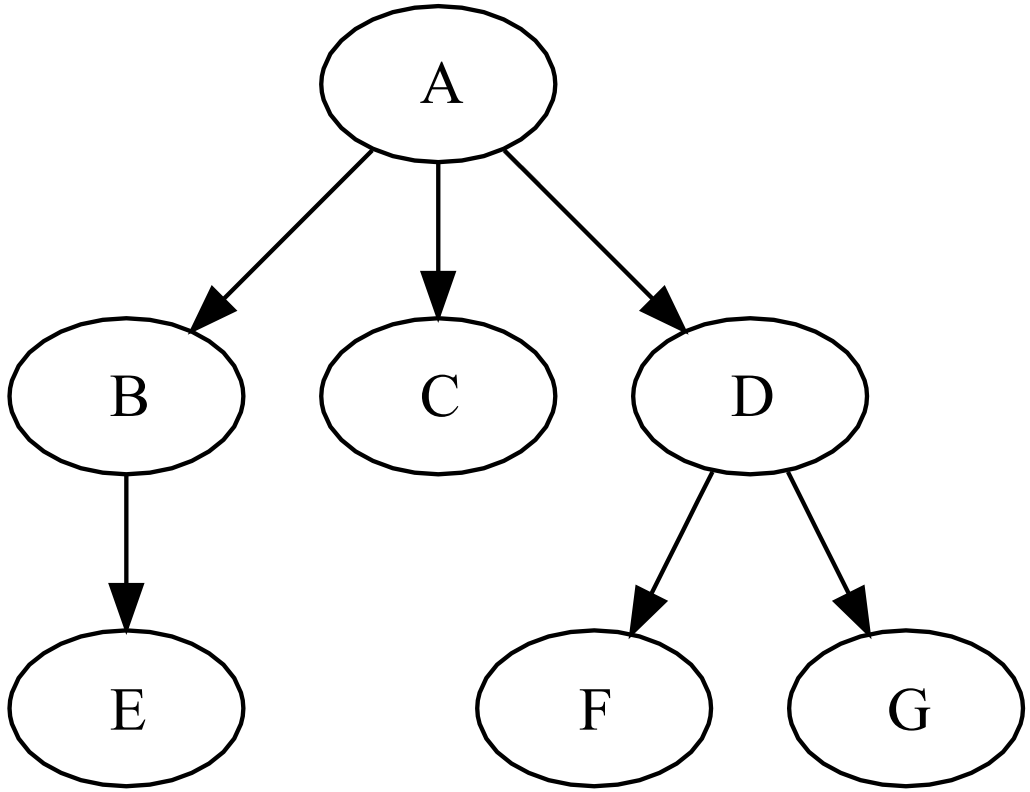


Définition récursive

- Définition récursive d'un arbre :
  - Soit un arbre est vide
  - Soit un arbre est constitué d'un nœud (la racine) et d'une suite de k arbres (appelés sous-arbres)

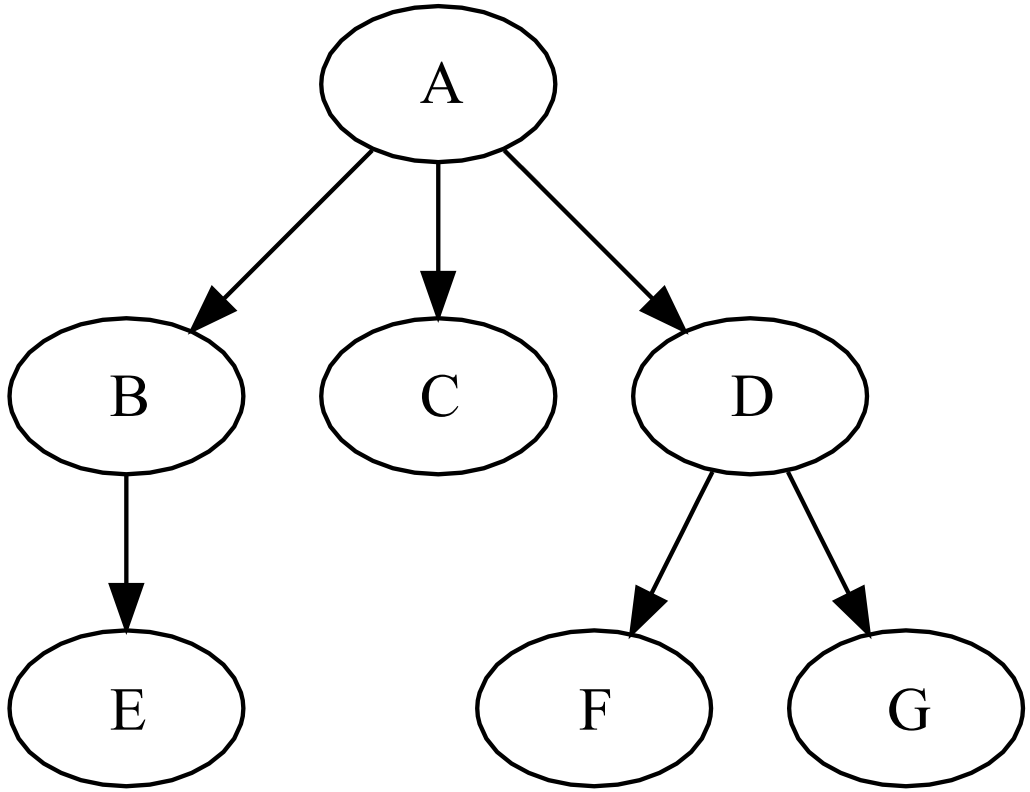
Définitions avancées (1/2)

- Branche : Une branche d'un arbre est un chemin $(n_1,n_2,\ldots,n_k)$, tel que :
  - n_1 est la racine de l'arbre
  - Pour tout $1 \leq i < k$, n_i est le père de n_{i+1}
  - Exemple : (A,B,E) est une branche
- Fils : Le nœud f est fils du nœud n si n est le père de f
  - Exemple : G est fils de D
- Frère : Le nœud f est frère du nœud n si n et f ont le même père
  - Exemple : B et D sont frères



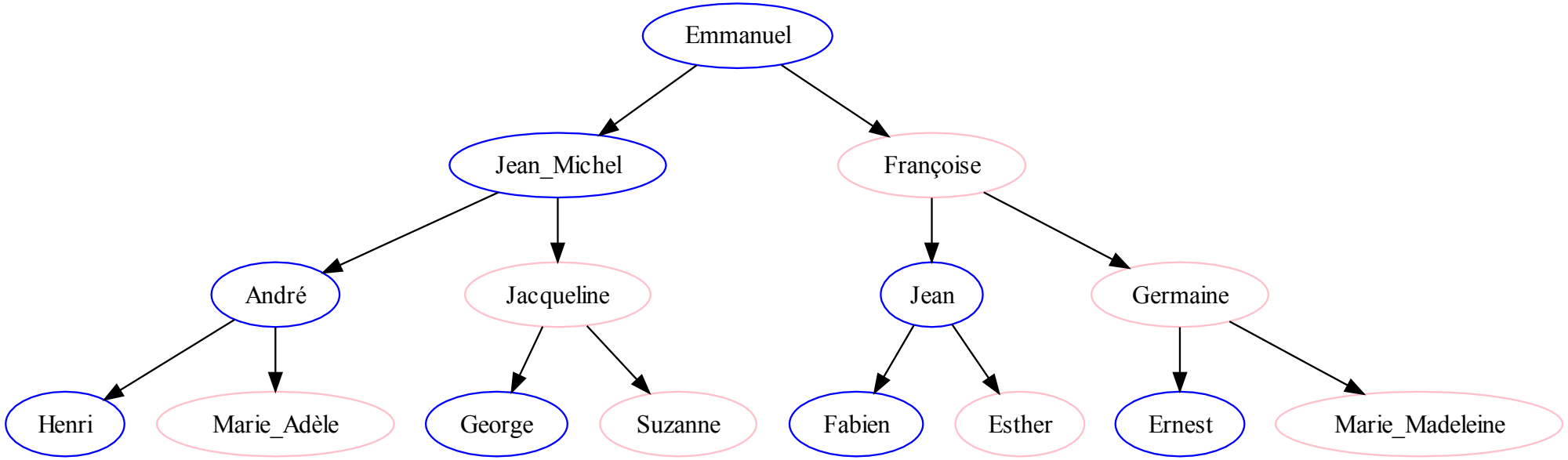
Définitions avancées (2/2)

- Descendant : Le nœud d est un descendant de n si d appartient à un sous-arbre de n
  - Exemple : F est descendant de A
- Ancêtre : Le nœud a est un ancêtre de n si n appartient à un sous-arbre de a
  - Exemple : A est ancêtre de E
- Hauteur : La hauteur d'un nœud n est la longueur de la branche qui mène à n (la racine ayant une profondeur 0)
  - Exemple : F est de hauteur 2



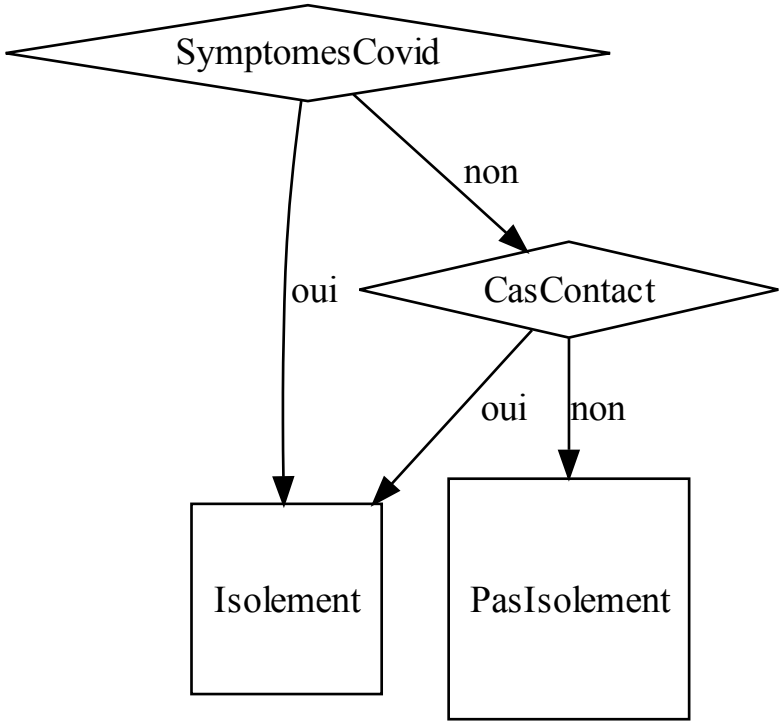
Exemple d’arbres (1/3)

- Arbre généalogique (d’Emmanuel Macron)



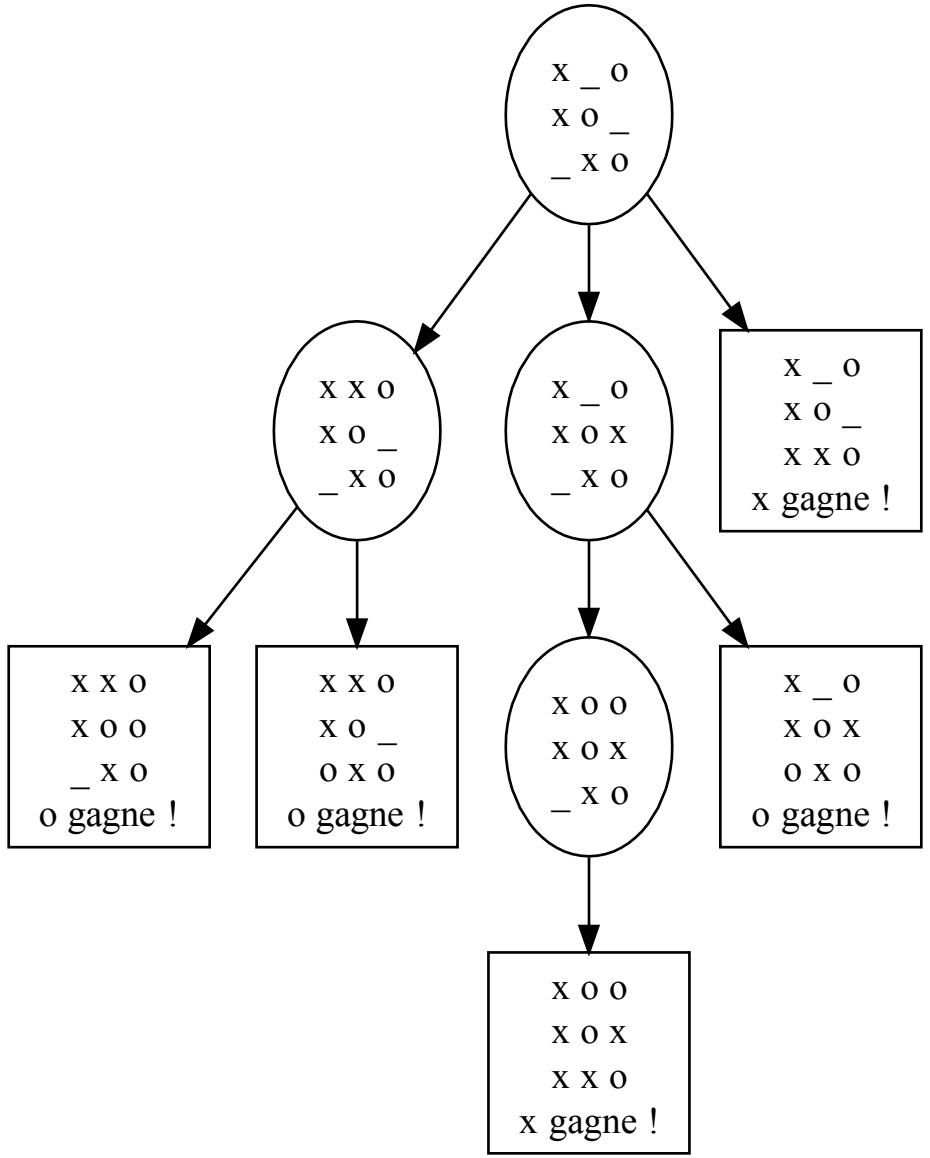
Exemple d’arbres (2/3)

- Arbre de décision (isolement lors de la crise sanitaire du Covid19)



Exemple d’arbres (3/3)

- Arbre des possibilités (depuis une position donnée du Morpion)

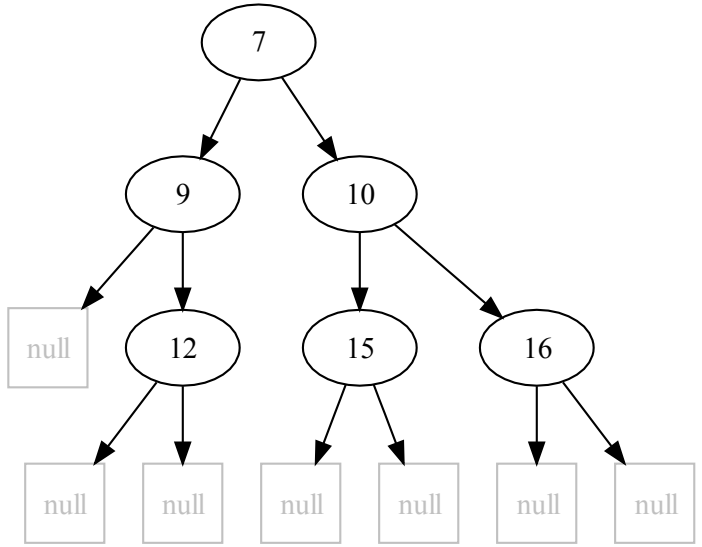


Propriétés

- Un arbre n’a pas de cycle
- Un arbre est connexe = il existe un chemin (= une branche) de la racine à n’importe quel nœud
- Le nombre d’arcs dans un arbre est égal au nombre de nœuds moins un

Quelques arbres classiques

- Arbre n-aire :
  - Chaque nœud a au plus n fils
- Arbre binaire : arbre n-aire avec n=2
  - Les deux fils d'un arbre non-vide sont appelés fils gauche et fils droit
- Arbre n-aire complet :
  - Chaque nœud contient 0 ou n fils
  - Toutes les feuilles ont la même hauteur
- Peigne
  - Chaque nœud a 0 ou 1 fils
  - Un peigne est une liste chaînée



Propriétés des arbres binaires et des arbres binaires complets

- Dans un arbre binaire complet :
  - Il y a 2^h nœuds à la hauteur h
  - Le nombre total de nœuds est $2^{h+1}-1$
- Dans un arbre binaire :
  - Le nombre total de nœuds est inférieur ou égal à $2^{h+1}-1$

Implémentation des arbres binaires en C (1/2)

- Structure :

```
struct tree {
        int value;
        struct tree * leftTree;
        struct tree * rightTree;
};
```

- Création d'un nouveau nœud :

```
struct tree * newNode(int v) {
        struct tree * t;
        t = (struct tree *)
        malloc(1*sizeof(struct tree));
        if (t==NULL) { return NULL; }
        t->value = v;
        t->leftTree = NULL;
        t->rightTree = NULL;
        return t;
}
```

Implémentation des arbres binaires en C (2/2)

- Suppression des nœuds d'un sous-arbre :

```
void removeNodes(struct tree * t) {
        if (t!=NULL) {
                if (t->leftTree!=NULL) {
                        removeNodes(t->leftTree);
                }
                if (t->rightTree!=NULL) {
                        removeNodes(t->rightTree);
                }
                free(t);
        }
}
```

A quoi servent les arbres en programmation ?

- Les arbres peuvent servir :
  - À modéliser des structures complexes (ex : modélisation de processus de décision simples par des arbres de décision, expression arithmétique, arbre de syntaxe abstraite pour la compilation, etc.)
  - À implémenter des opérations de recherche rapide

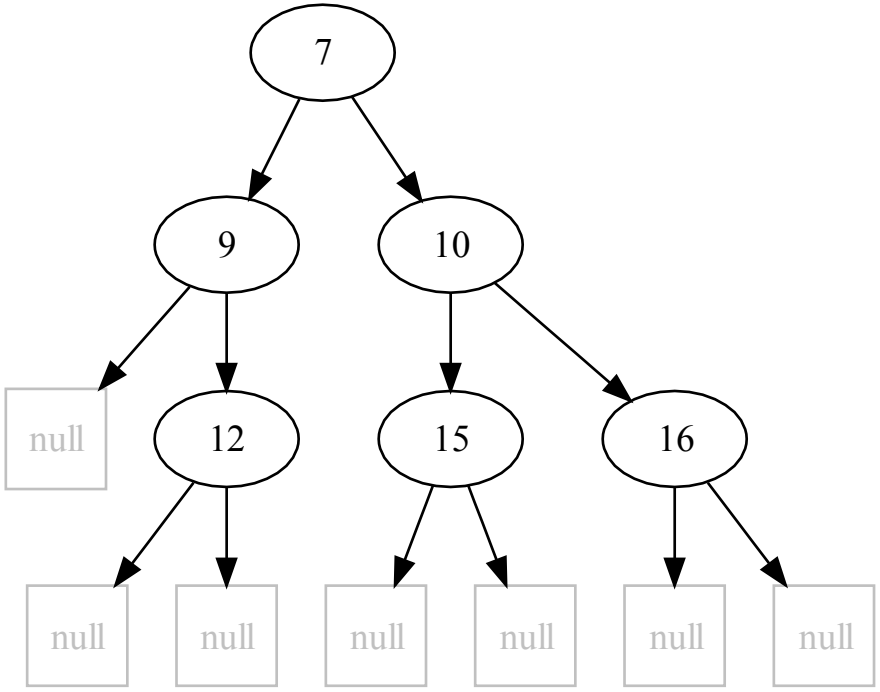
3.2 – Les parcours d’arbre

Définition

- Un parcours d'arbre est une énumération des nœuds de l'arbre, dans un certain ordre
- Le parcours de l'arbre est souvent :
  - Soit en largeur (donc, niveaux par niveaux)
  - Soit en profondeur (donc, en suivant une branche le plus loin possible, puis en revenant en arrière)

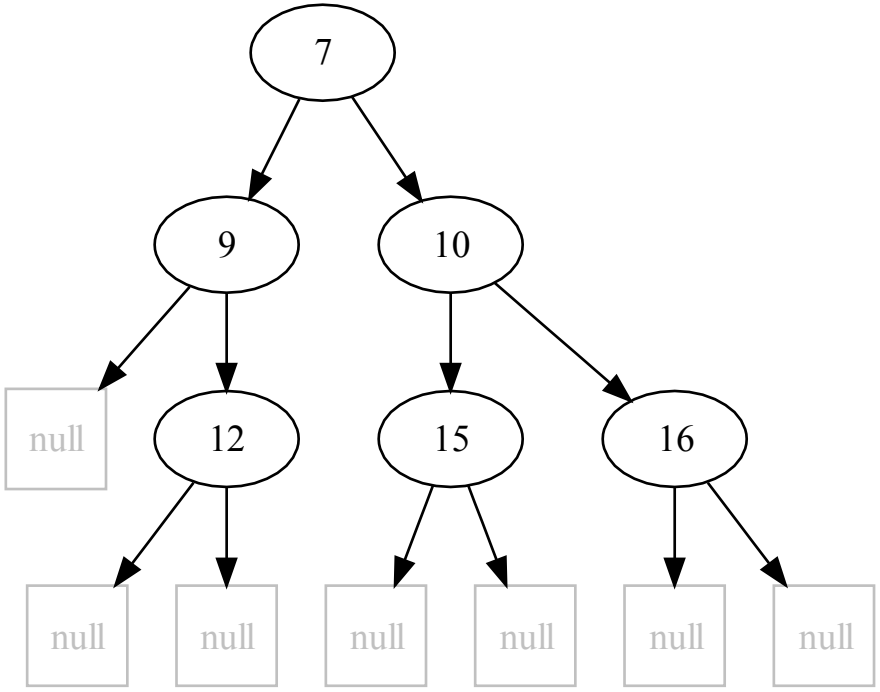
Parcours en largeur (1/2)

- Le parcours en largeur consiste à énumérer les nœuds d'un arbre par hauteur croissante (et généralement de gauche à droite)
  - Exemple : 7, 9, 10, 12, 15, 16



Parcours en largeur (2/2)

- Algorithme parcoursLargeur
  Entrée : arbre a
  f ← créer file vide
  enfiler a dans f
  tantque f non vide faire
  n ← défiler f
  si n != vide alors
  afficher n->valeur
  enfiler n->gauche dans f
  enfiler n->droit dans f
  finsi
  fintantque
- Exemple : 7, 9, 10, 12, 15, 16

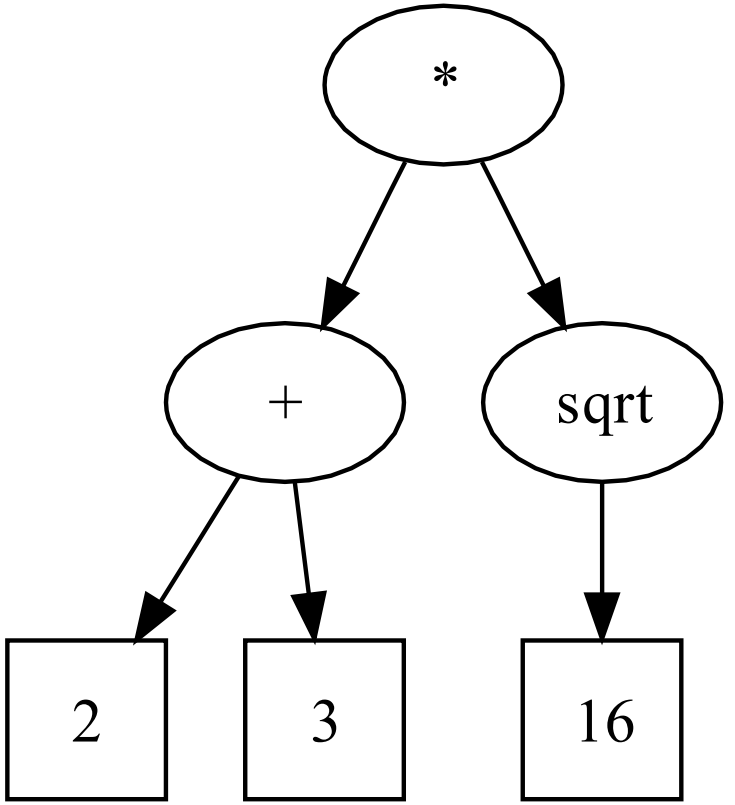


Parcours en profondeur

- Le parcours en profondeur consiste à énumérer les nœuds d'un arbre en partant d'une branche que l'on suit le plus loin possible
- Il existe trois types de parcours en profondeur :
  - Le parcours préfixe (= valeur en premier)
  - Le parcours postfixe (= valeur en dernier)
  - Le parcours infixe (= valeur au milieu, pour les arbres binaires)

Le parcours préfixe

- Algorithme parcoursPréfixe
  Entrée : un arbre a
  si a != vide alors
  afficher a->valeur
  parcoursPréfixe(a->gauche)
  parcoursPréfixe(a->droit)
  finsi
- Exemple : *, +, 2, 3, sqrt, 16

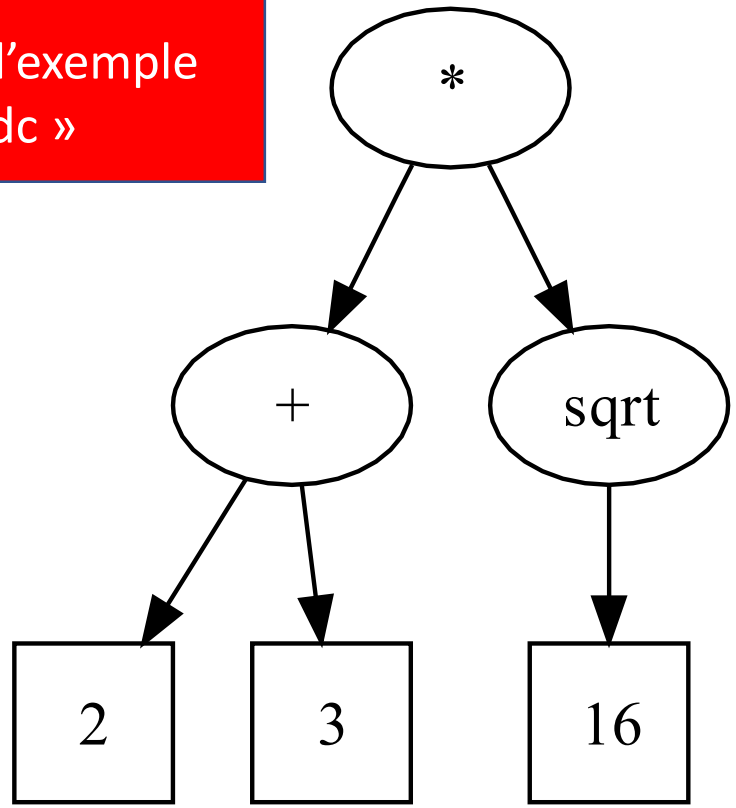


Le parcours postfixe

- Algorithme parcoursPostfixe
  Entrée : un arbre a
  si a != vide alors
  afficher a->valeur
  parcoursPostfixe(a->gauche)
  parcoursPostfixe(a->droit)
  finsi
- Exemple : 2, 3, +, 16, sqrt, *
- Remarque : la calculatrice “dc” sous Linux (qui supporte une précision arbitraire) utilise la notation postfixe (dite polonaise inversée)
  - Exemple : dc, 2, 3, +, 16, v, *, p, q

TODO

vérifier l’exemple « dc »



Le parcours infixe

- Algorithme parcoursInfixe
  Entrée : un arbre a
  si a != vide alors
  parcoursInfixe(a->gauche)
  afficher a->valeur
  parcoursInfixe(a->droit)
  finsi
- Exemple : 1, 2, 3, 4, 10, 13