Algorithmique et Structures de données

Mouhamadou GAYE

Enseignant-Chercheur
Département d'Informatique
Licence 2 en Ingénierie Informatique

2 octobre 2022

Chapitre 3 : Structures de données arborescentes

Introduction

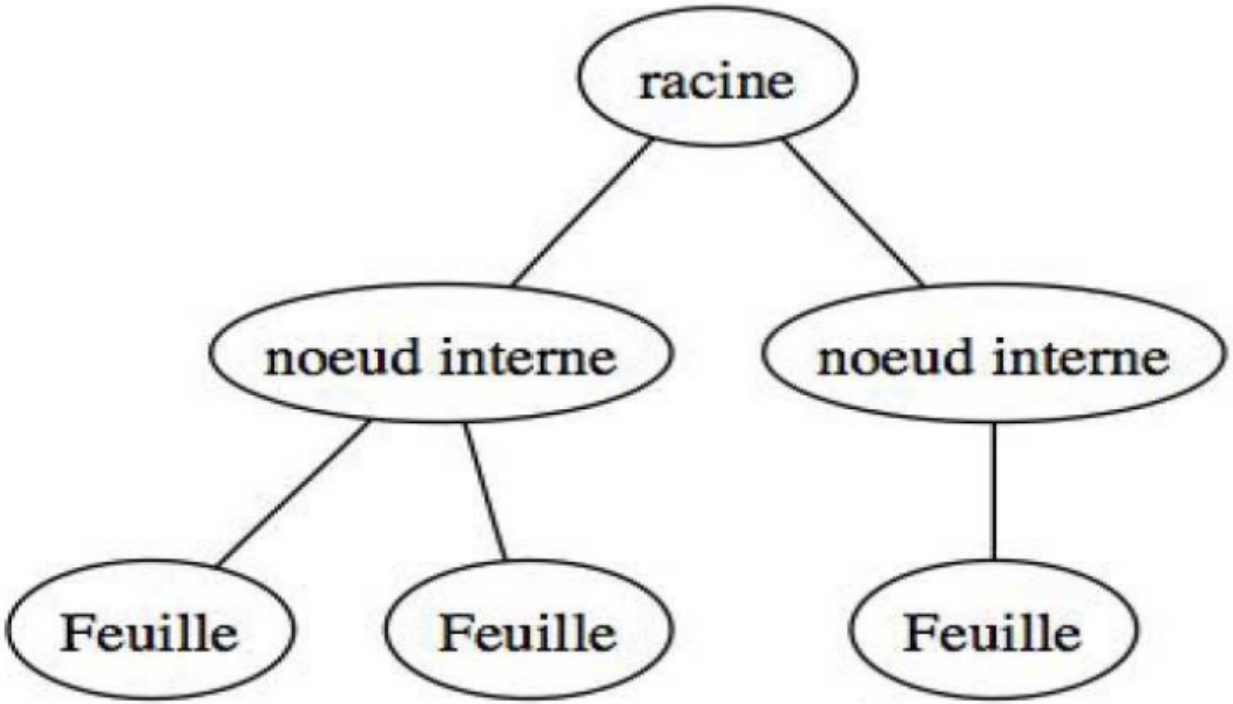
- Les structures arborescentes, tout comme les listes, permettent de représenter un nombre variable de données. Le principal avantage des arbres par rapport aux listes est qu'ils permettent de ranger les données de telle sorte que les recherches soient plus efficaces.

Arbres

Définition

Un arbre est une structure dynamique d'éléments de meme type organisés de manière hiérarchique. Ces éléments appelés souvent noeuds ou sommets sont reliés par arcs tel que chaque noeud (sauf le premier noeud) a exactement un arc pointé vers lui. Le premier noeud est appelé racine, les noeuds qui n'ont pas de successeurs (fils) sont appelés feuilles et le reste sont des noeuds internes.

Exemple


racine
noeud interne
noeud interne
Feuille
Feuille
Feuille

Arbres

- **Terminologie**
  - **Arité**
    L'arité d'un noeud désigne le nombre de fils que le noeud possède. L'arité d'un arbre est le maximum des arités de ses noeuds. Un arbre d'arité n est appelé arbre n-aire.
  - **Taille**
    Elle désigne le nombre de noeuds de l'arbre.
  - **Profondeur**
    La profondeur d'un noeud est le nombre de liaisons (arcs) qui séparent le noeud à la racine de l'arbre.
  - **Hauteur**
    La hauteur d'un arbre est le maximum des profondeurs de ses fils. C'est le plus long chemin qui mène de la racine à une feuille.

Arbres binaires

Définition

Un arbre binaire est un arbre d'arité 2. C'est donc un triplet A = (sag, r, sad) où :

- r est la racine
- sag est un arbre binaire appelé sous-arbre gauche
- sad est un arbre binaire appelé sous-arbre droite.

Arbres binaires

- **Déclaration**

```
typedef struct noeud{
    type racine ;
    struct noeud *sag ;
    struct noeud *sad ;
} NOEUD, *arbre ;
```

Arbres binaires

- **Fonctions usuelles**
  - **Arbre vide**

```
int arbreVide(arbre A){
    return A == NULL;
}
```

  - **Fils gauche**

```
arbre filsGauche(arbre A){
    if (A != NULL)
        return A ->sag;
     return NULL;
}
```

Arbres binaires

- **Fonctions usuelles**
  - **Feuille**

```
int estUneFeuille(arbre A){
    if (A != NULL)
        return A->sag == NULL && A->sad == NULL;
     return 0;
}
```

Arbres binaires

- **Fonctions usuelles**
  - **Construction**

```
arbre construireArbre(type r, arbre Ag, arbre Ad){
    arbre nouv = (arbre)malloc(sizeof(NOEUD)) ;
        A->racine = r ;
        A->sag = Ag ;
        A->sad = Ad ;
     return nouv ;
}
```

Arbres binaires

- **Parcours**
  Le parcours d'un arbre consiste à visiter chaque noeud de l'arbre. Il peut se faire en largeur c'est-à-dire niveau par niveau ou en profondeur consistant à explorer l'arbre jusqu'au bout une branche avant de passer à la suivante.
  - **Parcours en profondeur**
    - **Parcours préfixé :** traiter la racine, parcours du sous-arbre gauche puis parcours du sous-arbre droite (RGD)
    - **Parcours infixé :** GRD
    - **Parcours postfixé :** GDR

Arbres binaires de recherche

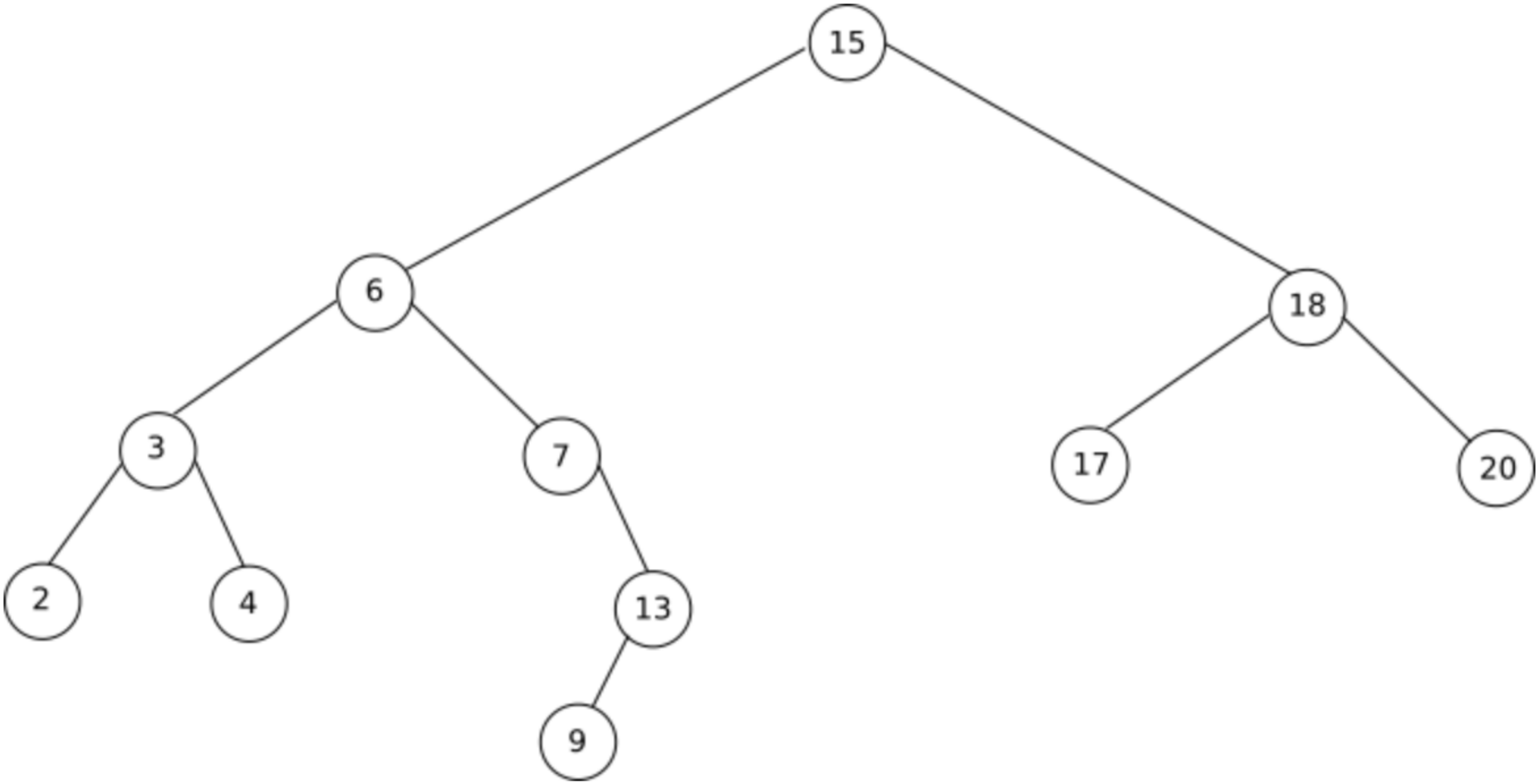
Définition

Un arbre binaire de recherche ABR est un arbre binaire ayant les propriétés suivantes :

- Tous les noeuds du sous-arbre gauche ont des valeurs inférieures ou égales à la racine ;
- Tous les noeuds du sous-arbre droite ont des valeurs strictement supérieures à la racine ;
- Le sous-arbre gauche et le sous-arbre droite sont des ABR.

Arbres binaires de recherche

Exemple



Arbres binaires de recherche

- **Fonctions usuelles**
  - **Recherche d'un élément**

```
int recherche(arbre A, type x){
    if (A == NULL)
        return 0 ;
    if (A ->racine == x)
        return 1 ;
    if (A ->racine > x)
        return recherche(A->sag, x) ;
    return recherche(A->sad, x) ;
}
```

Arbres binaires de recherche

- **Fonctions usuelles**
  - **Maximum / Minimum**

```
type maximum(arbre A){
    if (A->sad == NULL)
        return A->racine;
    return maximum(A->sad);
}
type minimum(arbre A){
    if (A->sag == NULL)
        return A->racine;
    return minimum(A->sag);
}
```

- **Fonctions usuelles**
  - **Suppression**
    Trois cas de figure :
    1. Le noeud concerné est une feuille : on supprime directement le feuille ;
    2. Le noeud concerné possède un fils : on replace le noeud par son fils et on supprime le fils feuille ;
    3. Le noeud concerné possède des descendants : on peut remplacer le noeud par la feuille du sous-arbre gauche contenant la valeur maximale puis on supprime cette feuille.

Arbre AVL

Définition

Un arbre AVL (Adelson-Velsky et Landis) est arbre binaire de recherche tel que pour chaque noeud la différence de hauteur de ses sous-arbres est au maximum un.