Chapitre 4

Arbres binaires

4.1 Introduction

Un arbre est une structure composée de noeuds et de feuilles (noeuds terminaux) reliés par des branches. On le représente généralement en mettant la racine en haut et les feuilles en bas (contrairement à un arbre réel).

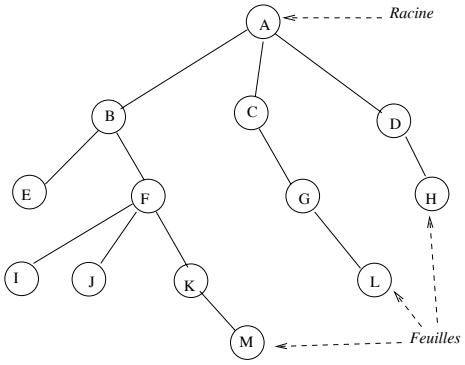

FIG. 4.1 – *Exemple d'arbre*

- Le noeud A est la racine de l'arbre.
- Les noeuds E, I, J, M, L et H sont des feuilles.
- Les noeuds B, C, D, F, G et K sont des noeuds intermédiaires.
- Si une branche relie un noeud n_i à un noeud n_j situé plus bas, on dit que n_i est un ancêtre de n_j.
- Dans un arbre, un noeud n'a qu'un seul père (ancêtre direct).
- Un noeud peut contenir une ou plusieurs valeurs.
- La hauteur (ou profondeur) d'un noeud est la longueur du chemin qui le lie à la racine.

4.2 Arbres binaires

Un arbre binaire est un arbre tel que les noeuds ont au plus deux fils (gauche et droit).

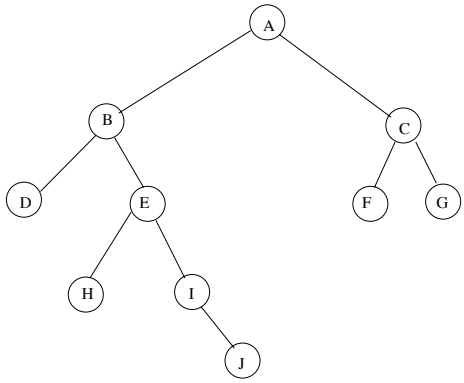

FIG. 4.2 – *Exemple d'arbre binaire*

4.3 Arbres binaires de recherche

Un arbre binaire de recherche est un arbre binaire qui possède la propriété fondamentale suivante:

- tous les noeuds du sous-arbre de gauche d'un noeud de l'arbre ont une valeur inférieure ou égale à la sienne.
- tous les noeuds du sous-arbre de droite d'un noeud de l'arbre ont une valeur supérieure ou égale à la sienne.

Exemple:

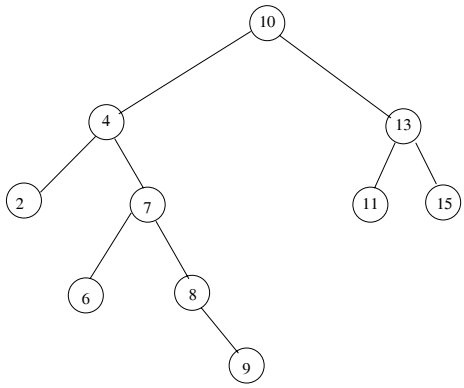

FIG. 4.3 – *Arbre binaire de recherche*

4.3.1 Recherche dans l'arbre

Un arbre binaire de recherche est fait pour faciliter la recherche d'informations.
La recherche d'un noeud particulier de l'arbre peut être définie simplement de manière récursive:

Soit un sous-arbre de racine n_i,

- si la valeur recherchée est celle de la racine n_i, alors la recherche est terminée. On à trouvé le noeud recherché.
- sinon, si n_i est une feuille (pas de fils) alors la recherche est infructueuse et l'algorithme se termine.
- si la valeur recherchée est plus grande que celle de la racine alors on explore le sous-arbre de droite c'est à dire que l'on remplace n_i par son noeud fils de droite et que l'on relance la procédure de recherche à partir de cette nouvelle racine.

– de la même manière, si la valeur recherchée est plus petite que la valeur de n_i, on remplace n_i par son noeud fils de gauche avant de relancer la procédure..

Si l'arbre est équilibré chaque itération divise par 2 le nombre de noeuds candidats. La complexité est donc en $O(\log_2 n)$ si n est le nombre de noeuds de l'arbre.

4.3.2 Ajout d'un élément

Pour conserver les propriétés d'un arbre binaire de recherche nécessite de, l'ajout d'un nouvel élément ne peut pas se faire n'importe comment.

L'algorithme récursif d'ajout d'un élément peut s'exprimer ainsi:

- soit x la valeur de l'élément à insérer.
- soit v la valeur du noeud racine n_i d'un sous-arbre.
  - si n_i n'existe pas, le créer avec la valeur x. fin.
  - sinon
    - si x est plus grand que v,
      - remplacer n_i par son fils droit.
      - recommencer l'algorithme à partir de la nouvelle racine.
    - sinon
      - remplacer n_i par son fils gauche.
      - recommencer l'algorithme à partir de la nouvelle racine.

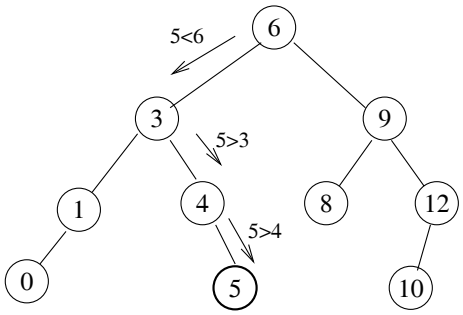

FIG. 4.4 – *Ajout de 5 dans l'arbre*

4.3.3 Implémentation

En langage C, un noeud d'un arbre binaire peut être représenté par une structure contenant un champ donnée et deux pointeurs vers les noeuds fils:

```
struct s_arbre
{
  int valeur;
  struct s_arbre * gauche;
  struct s_arbre * droit;
};
typedef struct s_arbre t_arbre;
```

La fonction d'insertion qui permet d'ajouter un élément dans l'arbre et donc de le créer de manière à ce qu'il respecte les propriétés d'un arbre binaire de recherche peut s'écrire ainsi:

```
void insertion(t_arbre ** noeud, int v)
```

```
{
  if (*noeud==NULL) /* si le noeud n'existe pas, on le crée */
    {
      *noeud=(t_arbre*) malloc(sizeof(t_arbre));
      (*noeud)->valeur=v;
      (*noeud)->gauche=NULL;
      (*noeud)->droit=NULL;
    }
  else
    {
      if (v>(*noeud)->valeur)
        insertion(&(*noeud)->droit,v);  /* aller a droite */
      else
        insertion(&(*noeud)->gauche,v); /* aller a gauche */
    }
}
```

On peut noter par ailleurs que l'arbre qui sera construit dépendra de l'ordre dans lequel seront insérées les différentes valeurs. En particulier, si les valeurs sont insérées dans un ordre croissnt on obtiendra un arbre complètement déséquilibré, chaque noeud n'ayant qu'un fils droit (gauche si les valeurs sont dans un ordre décroissant).