PC 5
Arbres

- Définitions
- Représentation
- Parcours d'arbres
- Files de priorité

Exemples

1. Toute expression possède un *arbre syntaxique*. Ainsi, l'instruction C

```
a = b*-c + b*-c;
```

se traduit par l'arbre

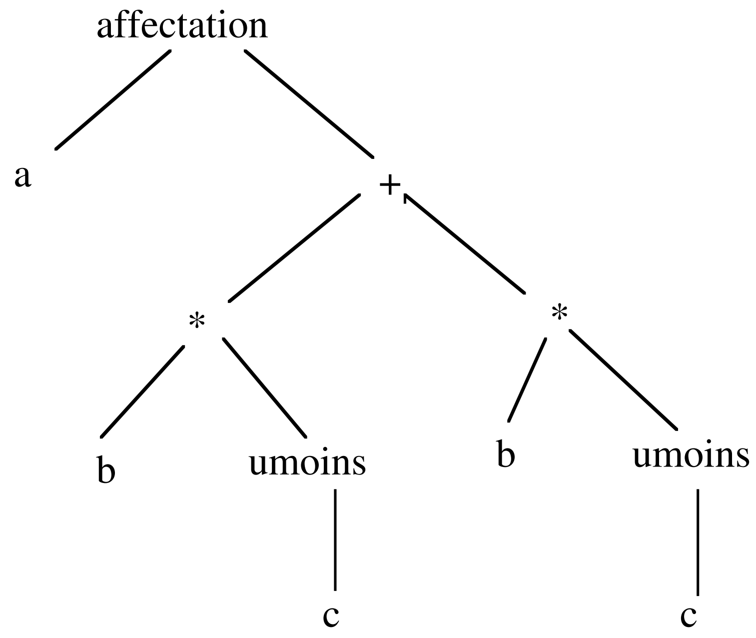

Figure 1: Un arbre syntaxique

2. Les fichiers enregistrés sur une machine sont en général organisés en arbre. Les noeuds internes sont les répertoires (ou dossiers) et les feuilles sont les fichiers.

Définitions

Un *arbre* est donné par

- Un ensemble N de *noeuds*.
- Un noeud distingué appelé la *racine*.
- Une fonction associant à tout noeud distinct de la racine un unique *père*.

Terminologie

Si n est le père de m, on dit que m est l' *enfant* de n.

Si m et n ont le même père, on dit qu'ils sont *frères*.

Si n est un ancêtre de m, on dit aussi que m est un *descendant* de n.

Une *feuille* est un noeud qui n'a pas de fils. Les autres sont des noeuds *internes*.

Définition récursive

Un arbre est soit réduit à sa racine, soit constitué d'un noeud (sa racine) et d'un ensemble d'arbres. Les racines de ces arbres sont les fils de la racine.

Symboliquement on peut écrire:

$$T = x + (x, T) + (x, T, T) + \dots$$

Cas particuliers

Un *arbre planaire* est un arbre sur lequel on a défini un ordre sur l'ensemble des fils de chaque noeud.

Un *arbre binaire* est un arbre dans lequel chaque noeud a au plus deux fils désignés comme fils gauche et droit.

Un arbre binaire complet est un arbre binaire dont tout noeud a 0 fils ou 2 fils.

Si un arbre binaire complet a n feuilles, il a $2n-1$ noeuds.

Représentation des arbres

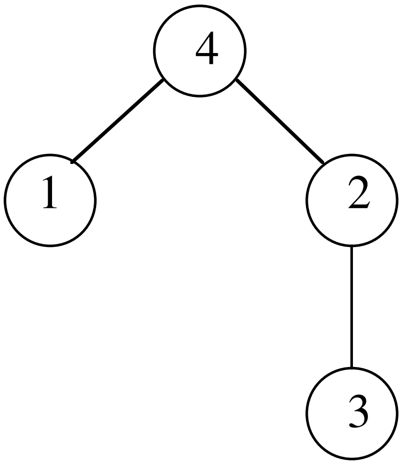

Figure 2:

Plusieurs représentations sont possibles:

- Tableau des pères:

1	2	3	4
4	4	2	0

- Liste de fils:

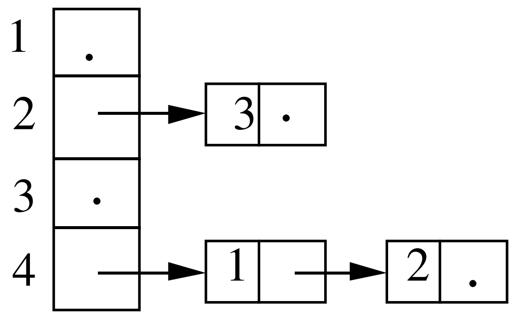

Figure 3:

Déclaration de types :

```
struct cellule{
  int num;
  struct cellule * suivant;
};
struct cellule * table[N];
```

- Représentation fils gauche-frère droit:

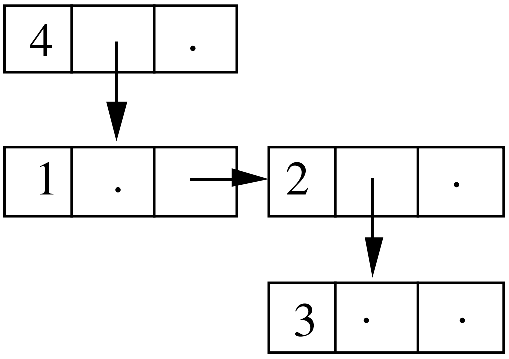

Figure 4:

Déclaration de type :

```
struct noeud {
  int num;
  struct noeud * fg;
  struct noeud * frd;
}
```

Parcours d'arbres

On considère pour simplifier un *arbre binaire* dans lequel chaque noeud a au plus deux fils: le fils gauche et le fils droit, éventuellement vides.

On définit plusieurs ordres de parcours:

1. L'ordre préfixe:

$$\pi((r, S, T)) = (r, \pi(S), \pi(T))$$

2. L'ordre infixe:

$$i((r, S, T) = (i(S), r, i(T))$$

3. L'ordre suffixe:

$$\sigma((r, S, T)) = (\sigma(S), \sigma(T), r)$$

L'ordre des feuilles est le même dans tous les cas.

Parcours préfixe

On considère des arbres binaires représentés par deux tableaux `fg[ ]`, `fd[ ]` d'entiers.

```
void Parcours(int r) {
  int n;
  printf("%d",r);
  n= fg[r];
  if (n != VIDE)
    Parcours(n);
  n= fd[r];
  if (n != VIDE)
    Parcours(n);
}
```

Enumération des arbres

Soit b_n le nombre d'arbres binaires ayant n noeuds. On a $b_0 = 1$,$b_1 = 1$ puis:

$$b_n = \sum_{i=0}^{n-1} b_i b_{n-i-1}$$

de sorte que si on note

$$b(z) = \sum_{n \geq 0} b_n z^n$$

on obtient:

$$b = zb^2 + 1$$

On en déduit que:

$$b = \frac{1 - \sqrt{1-4z}}{2z}$$

d'où la formule:

$$b_n = \frac{1}{n+1}\binom{2n}{n}$$

déduite de la formule du binôme généralisée

$$\sqrt{1-4z} = \sum_n \binom{\frac{1}{2}}{n}(-4)^n z^n$$

En effet
$$\binom{\frac{1}{2}}{n} = \frac{1}{2n}\binom{-\frac{1}{2}}{n-1}$$
et
$$\binom{-\frac{1}{2}}{n} = (\frac{-1}{4})^n\binom{2n}{n}$$
d'où
$$\begin{aligned} b &= (1 - \sum_{n\geq 0}\binom{\frac{1}{2}}{n}(-4)^n z^n)/2z \\ &= \sum_{n\geq 0}\binom{\frac{1}{2}}{n+1}(-4)^{n+1}z^{n+1}/2z \\ &= \sum_{n\geq 0}\frac{1}{2(n+1)}(-\frac{1}{4})^{n+1}\binom{2n}{n}(-4)^n(2z^n) \\ &= \sum_{n\geq 0}\frac{1}{n+1}\binom{2n}{n}z^n \end{aligned}$$

Code de Prüfer

Code de Prüfer d'un arbre étiqueté:

Soit T un arbre étiqueté dans $\{1, 2, \ldots, n\}$. Le *code de Prüfer* de T est la suite (i, S) obtenue en:

1. choisissant la plus petite feuille j de T et en notant son père i.
2. supprimant la feuille j.
3. Calculant le code de l'arbre restant

Le code d'un arbre réduit à sa racine est vide.

Conséquence: Le nombre d'arbres étiquetés dans $\{1, 2, \ldots, n\}$ est n^{n-1}.

Files de priorité

Arbres binaires représentés dans un tableau $a[0], a[1], \ldots, a[n]$ avec la convention que les sommets sont rangés dans l'ordre militaire (de gauche à droite, niveau par niveau). L'arbre:

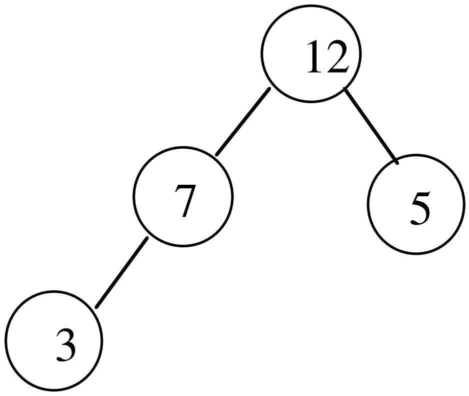

Figure 5:

est représenté par le tableau:

0	1	2	3
12	7	5	3

On suppose de plus que l'étiquette du père est supérieure à celles de ses fils.

Propriété essentielle: les fils du noeud i sont $2i+1$ et $2i+2$.

Opérations

Les opérations de base sur les files de priorité sont:

1. Ajouter un élément.
2. Supprimer le maximum

L'ajout d'un élément se fait par *promotion*:

```
void Ajouter (int v) {
 int i;

  n = n+1;
  i = n;
  while (i> 0 && a[(i-1) / 2] <= v ) {
    a[i] = a[(i-1) / 2];
    i = (i-1) / 2;
  }
  a[i] = v;
}
```

La suppression du maximum se fait par *tamisage* du dernier élément mis à la place de la racine:

```
void Supprimer() {
  int i, j, v;
  a[0] = a[n]; --n;
  i= 0; v = a[0];
  while (2*i+1 < n ) {
     j = 2*i+1;
     if (j<n && a[j+1] > a[j])
       ++j;
     if (v > a[j]) break;
     a[i] = a[j]; i = j;
  }
   a[i] = v;
}
```