arbres de recherche

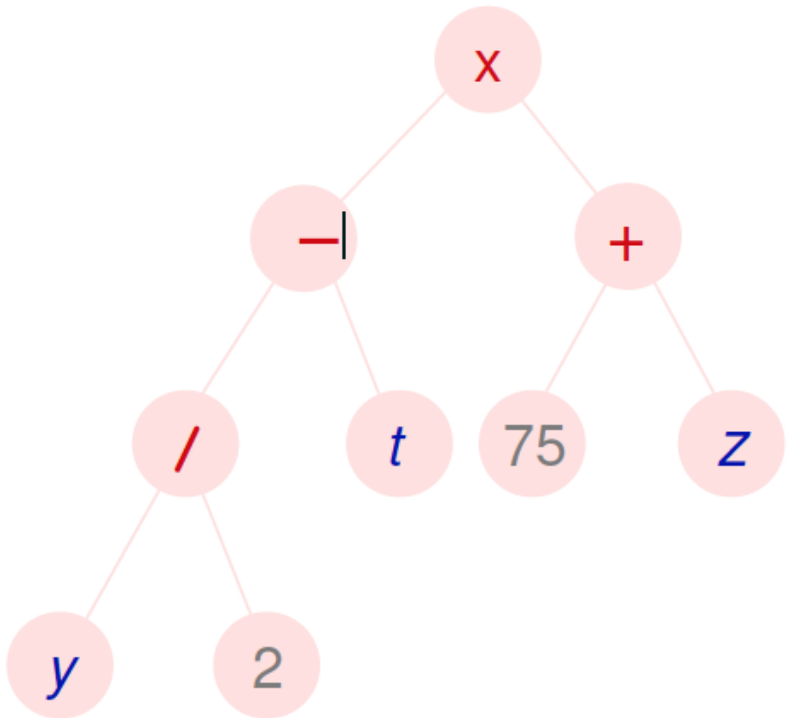
November 26, 2021

- **Les Arbres binaires de recherche, arbres lexicographiques Les arbres et les structures arborescentes** sont tres utilises en informatique.
- D'une part les informations sont souvent hierarchisees, et se présentent donc naturellement sous une forme arborescente, et d'autre part de nombreuses structures de données parmi les plus efficaces sont représentées par des arbres (les tas, les arbres binaires de recherches, les B-arbres, les forets, . . .).

Exemples

On représente souvent des expressions arithmétiques avec des arbres étiquettes par des opérateurs, des constantes et des variables. La structure de l'arbre élimine les ambiguïtés qui peuvent apparaître en fonction des priorités des opérateurs et qui sont supprimées en utilisant des parenthèses.

$$(y/2 - t)X(75 + z)$$



Arbres syntaxiques.

Un arbre syntaxique représente l'analyse d'une phrase à partir d'un ensemble de règles qui constitue la grammaire : une phrase est composée d'un groupe nominal suivi d'un groupe verbal, un groupe nominal peut-être constitué d'un article et d'un nom commun,. . .

Figure:

Arbres généalogiques.

Un arbre généalogique (descendant dans le cas présent) représente la descendance d'une personne ou d'un couple. Les nœuds de l'arbre sont étiquetés par les membres de la famille et leurs conjoints. L'arborescence est construite à partir des liens de parenté (les enfants du couple).

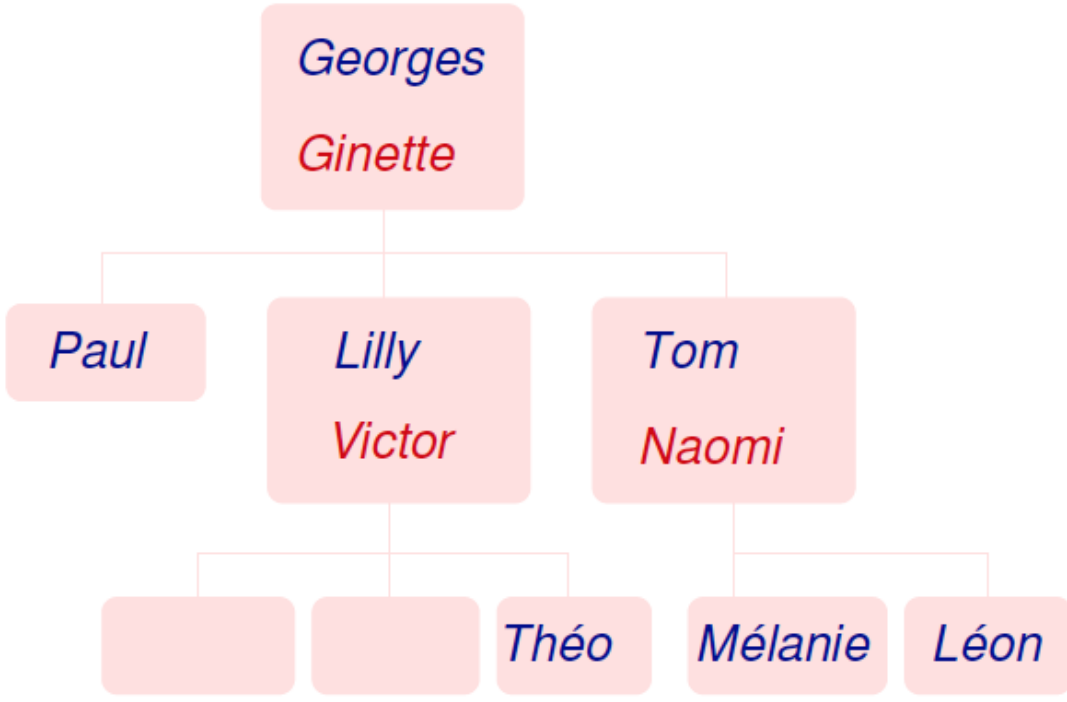
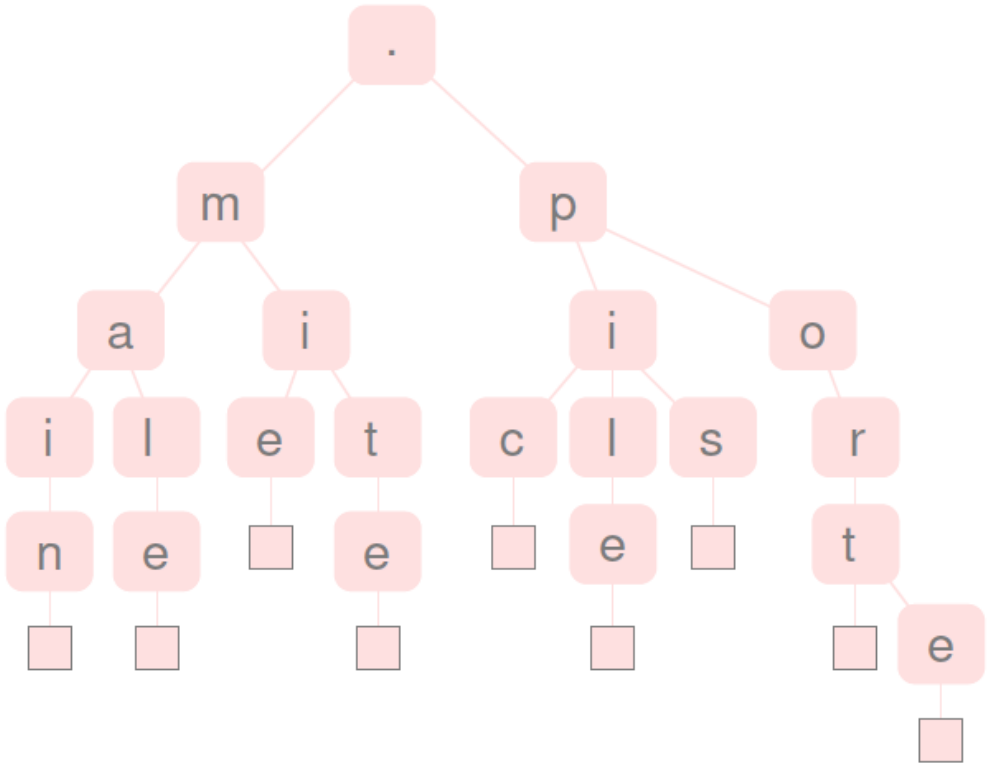

Georges
Ginette
Paul
Lilly
Victor
Tom
Naomi
Théo
Mélanie
Léon

Figure:

Arbre lexicographique.

- Un arbre lexicographique, ou arbre en parties communes, ou dictionnaire, représente un ensemble de mots.
- Les préfixes communs à plusieurs mots apparaissent une seule fois dans l'arbre, ce qui se traduit par un gain d'espace mémoire.
- De plus la recherche d'un mot est assez efficace, puisqu'il suffit de parcourir une branche de l'arbre en partant de la racine, en cherchant à chaque niveau parmi les fils du nœud courant la lettre du mot de rang correspondant.

Arbre lexicographique.



main
male
mie
mite
pic
pile
pis
port
porte

Figure:

Définitions et terminologie

- **Définition**. Un arbre est un ensemble organisé de nœuds dans lequel chaque nœud *a* un père et un seul, sauf un nœud que l'on appelle la racine.
- Si le nœud p est le père du nœud f,nous dirons que f est un fils de p,et si le nœud p n'a pas de fils nous dirons que c'est une feuille. Chaque nœud porte une étiquette ou valeur ou clé.
- On a l'habitude, lorsqu'on dessine un arbre, de le représenter avec la tête en bas, c'est-à-dire que la racine est tout en haut, et les nœuds fils sont représentés en-dessous du nœud père.

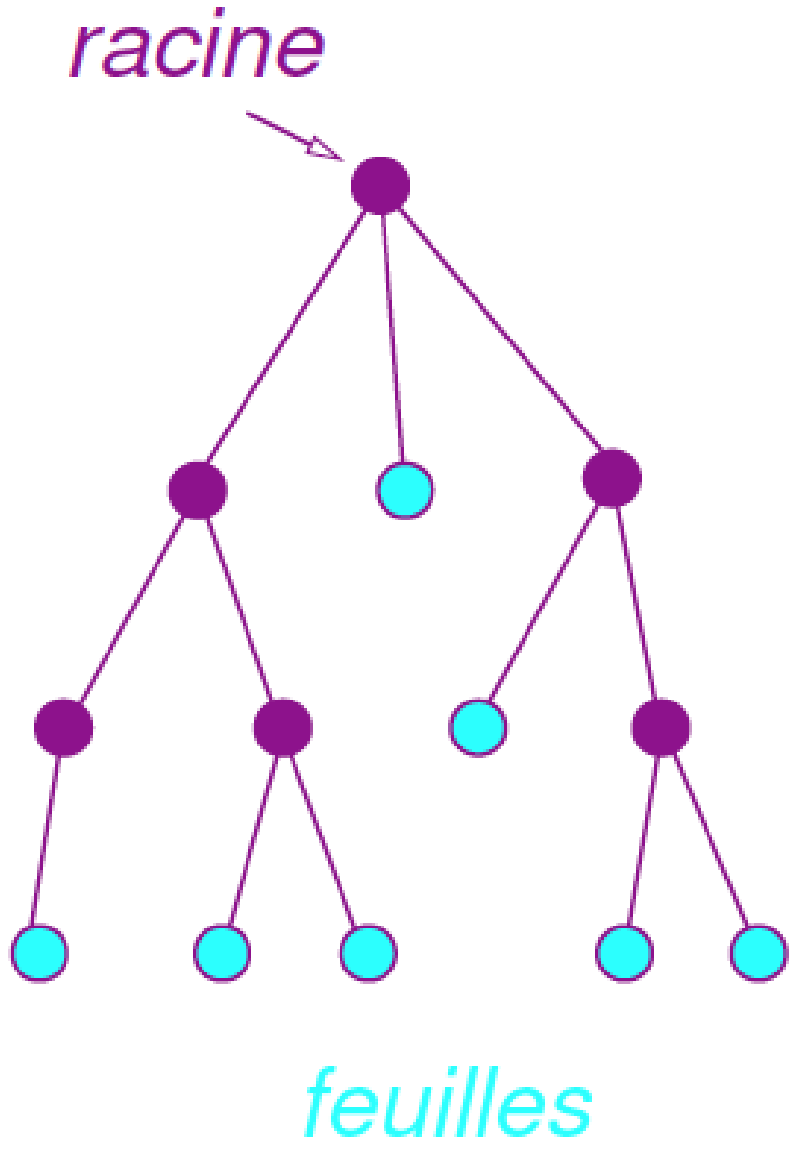


Figure:

Un nœud est défini par son étiquette et ses sous-arbres. On peut donc représenter un arbre par un $n-uplet$ de, $(e, a_1, ..., a_k)$ dans lesquels e est l étiquette portée par le nœud, et $a_1, ..., a_k$ sont ses sous-arbres. Par exemple l'arbre correspondant à l'expression arithmétique $(y/2 - t)x(75 + z)$ sera représenté par

$$(x, (-, (/, (y), (2)), (t), (+, (75), (z))))$$

- On distingue les arbres binaires des arbres généraux. Leur particularité est que les fils sont singularisés : chaque nœud a un fils gauche et un fils droit. L'un comme l'autre peut être un arbre vide, que l'on notera .
- L'écriture d'un arbre s'en trouve modifiée, puisqu'un nœud a toujours deux fils. En reprenant l'expression arithmétique précédente, l'arbre binaire qui la représente s'écrit

$$\langle \times, \langle -, \langle /, \langle y, \langle\rangle, \langle\rangle\rangle, \langle 2, \langle\rangle, \langle\rangle\rangle\rangle, \langle t, \langle\rangle, \langle\rangle\rangle\rangle, \langle +, \langle 75, \langle\rangle, \langle\rangle\rangle, \langle z, \langle\rangle, \langle\rangle\rangle\rangle\rangle$$

Figure:

On utilise pour les arbres une terminologie inspirée des liens de parenté :

- les descendants d'un nœud p sont les nœuds qui apparaissent dans ses sous-arbres,
- un ancêtre d'un nœud p est soit son père, soit un ancêtre de son père,
- le chemin qui relie un nœud à la racine est constitué de tous ses ancêtres (c'est-a'-dire de son père et des nœuds du chemin qui relie son père à la racine),
- un frère d'un nœud p est un fils du père de p, et qui n'est pas p.

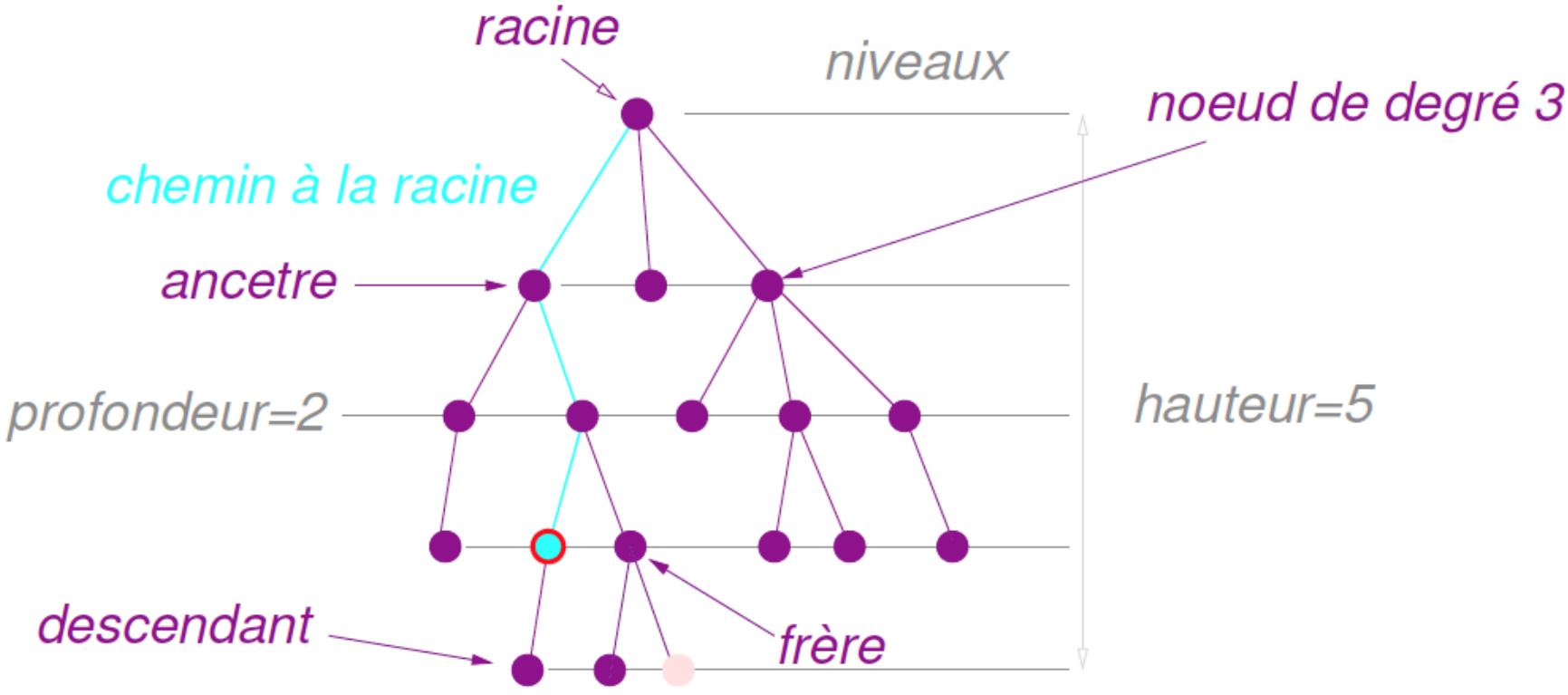

racine
niveaux
noeud de degré 3
chemin à la racine
ancetre
profondeur=2
hauteur=5
descendant
frère

Figure:

Les nœuds d'un arbre se répartissent par niveaux :

- le premier niveau (par convention ce sera le niveau 0) contient la racine seulement,
- le deuxième niveau contient les deux fils de la racine,. . . ,
- les nœuds du niveau k sont les fils des nœuds du niveau k -1,. . . .
- La hauteur d'un arbre est le nombre de niveaux de ses nœuds. C'est donc aussi le nombre de nœuds qui jalonnent la branche la plus longue. Attention, la définition de la hauteur varie en fonction des auteurs. Pour certains la hauteur d'un arbre contenant un seul nœud est 0.

Arbres binaires : hauteur, nombre de nœuds et nombre de feuilles.

- Un arbre binaire est complet si toutes ses branches ont la même longueur et tous ses nœuds qui ne sont pas des feuilles ont deux fils.
- Soit A un arbre binaire complet. Le nombre de nœuds de A au niveau 0 est 1, le nombre de nœuds au niveau 1 est 2,. . . , et le nombre de nœuds au niveau p est donc 2^p.
- En particulier le nombre de feuilles est donc 2^{h-1} si h est le nombre de niveaux.

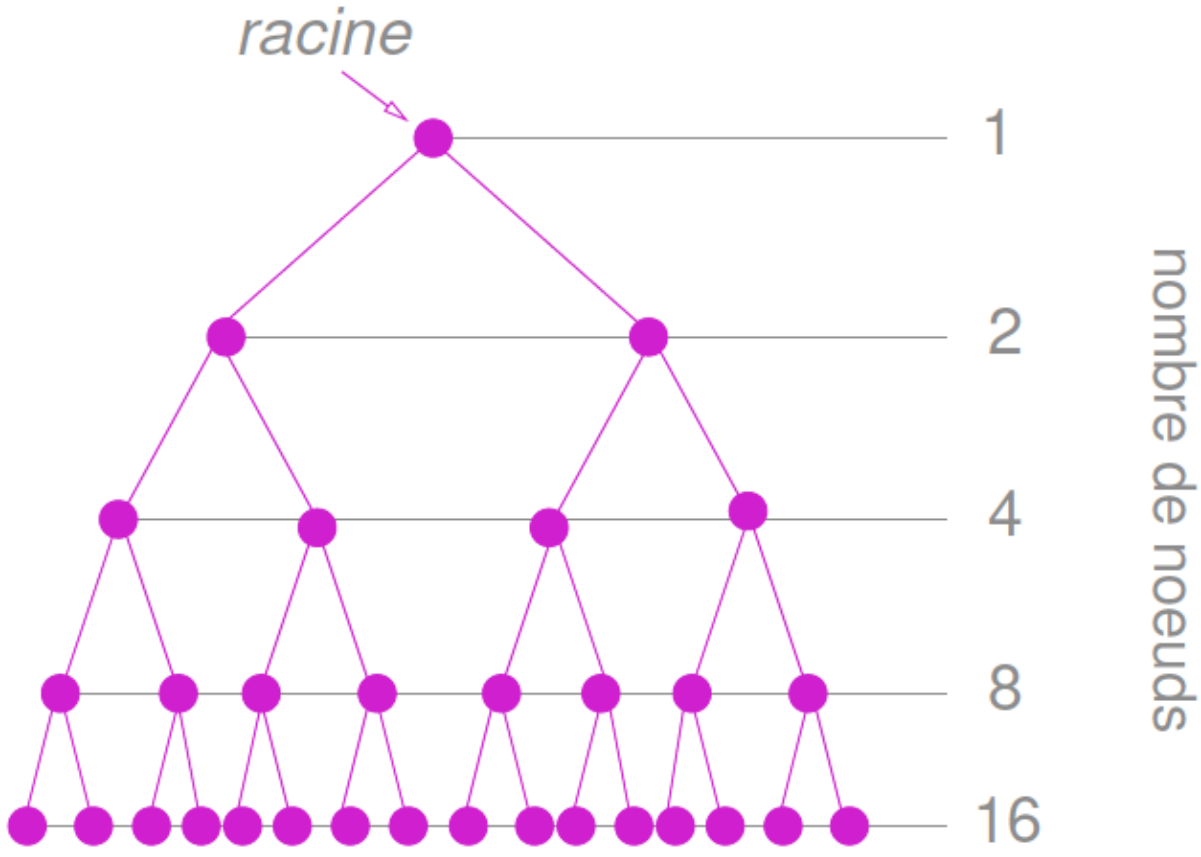


Figure:

Le nombre total de nœuds d'un arbre binaire complet de h niveaux est

$$\sum_{i=0}^{h-1} 2^i = 2^h - 1$$

Figure:

On en déduit que la hauteur ht(A) (le nombre de niveaux) d'un arbre binaire A contenant n nœuds est au moins égale à

$$\lfloor \log_2 n \rfloor + 1$$

Figure:

Preuve

- Si A est arbre de hauteur h et comportant n nœuds, pour tout entier m,on a:$h < m - 1 => n < 2^{m-1}$.
- Par contraposée, on a: $n >= 2^{m-1} => h >= m$. Le plus grand entier m vérifiant $n >= 2m - 1$ est $log_2(n) + 1$. On a donc $h >= log_2(n) + 1$.
- La hauteur minimale $log_2(n) + 1$ est atteinte lorsque l'arbre est complet. Pour être efficaces, les algorithmes qui utilisent des arbres binaires font en sorte que ceux ci soient équilibrés (voir les tas ou les AVL-arbres par exemple).

Les arbres en théorie des graphes.

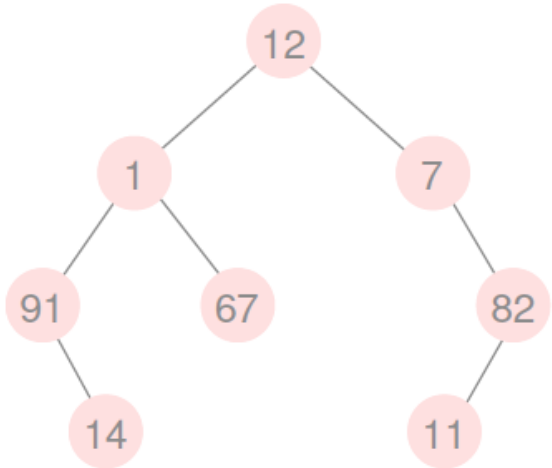
- En théorie des graphes un arbre est un graphe connexe et sans cycles, c'est-à-dire qu'entre deux sommets quelconques du graphe il existe un chemin et un seul.
- On parle dans ce cas d'arbre non planté, puisqu'il n'y a pas de sommet particulier qui joue le rôle de racine.
- Les sommets sont voisins les uns des autres, il n'y a pas de hiérarchie qui permet de dire qu'un sommet est le père (ou le fils) d'un autre sommet.
- On démontre qu'un graphe de n sommets qui a cette propriété contient exactement n-1 arêtes, et que l'ajout d'une nouvelle arête crée un cycle, tandis que le retrait d'une arête le rend non connexe.

Arbres binaires En C

- on utilise en général des pointeurs pour représenter les liens entre un nœud et ses fils dans un arbre binaire.
- Il existe plusieurs façons de définir le type ARBRE. Nous proposons la définition suivante, qui permet d'imbriquer différents types les uns dans les autres (par exemple si on veut des arbres dont les valeurs sont des listes chaînées dont les valeurs sont elles-mêmes des arbres,

```
typedef struct nœud * ARBRE;
struct nœud
 {
   TYPE_VALEUR val;
   ARBRE fg, fd;
};
```

Un arbre en C est donc désigné par l'adresse de sa racine. Il est facile d'accéder à un nœud quelconque de l'arbre à partir de la racine, en suivant les liens explicites vers le fils droit ou le fils gauche.



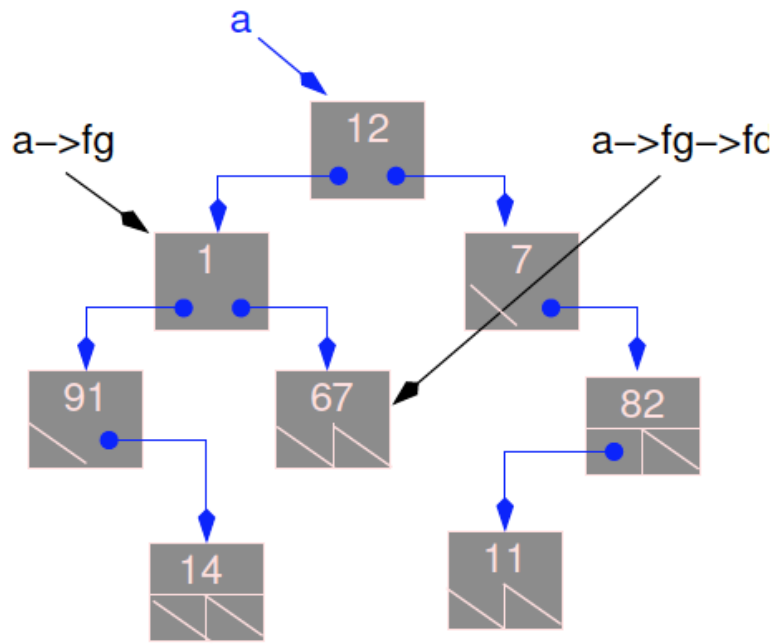


Figure:

La construction d'un arbre consiste,à partir d'un arbre vide,à insérer des nouveaux nœuds. La fonction suivante fait l'allocation d'un nœud et l'initialisation des champs de la structure. Il est recommandé d'utiliser cette fonction chaque fois qu'un nouveau nœud doit être crée

```
Creernœud(TYPE_VALEUR v, ARBRE fg, ARBRE fd) {
          ARBRE p;
          p = malloc(sizeof(struct nœud));
          assert(p != NULL);
          p->val = v;
          p->fg = fg;
          p->fd = fd;
          return p;
}
```

Arbres binaires : parcours.

Le principe est simple : pour parcourir l'arbre a, on parcourt récursivement son sous-arbre gauche, puis son sous- arbre droit. Ainsi le sous-arbre gauche sera exploré dans sa totalité avant de commencer l'exploration du sous-arbre droit.

```
void Parcours(ARBRE a)
        {
           if (a != NULL)
           {
              /* traitement avant */
              Parcours (a->fg);
              /* traitement entre */
              Parcours (a->fd);
              /* traitement apres */
           }
}
```

- Dans ce schéma l'exploration descend d'abord visiter les nœuds les plus à gauche : on parle de parcours en profondeur d'abord.
- On distingue le parcours **préfixe**, le parcours **infixe** et le parcours **postfixe**.
- La différence entre ces parcours tient uniquement à l'ordre dans lequel sont traités les nœuds du sous-arbre gauche, le nœud courant, et les nœuds du sous-arbre droit.

Parcours préfixe :

```
void Parcours(ARBRE a)
        {
           if (a != NULL)
           {
              /* traitement avant */
              Parcours (a->fg);
              /* traitement entre */
              Parcours (a->fd);
              /* traitement apres */
           }
}
```

```
void ParcoursPrefixe(ARBRE a)
{
   if (a != NULL)
   {
      TraiterLaValeur (a->val);
      ParcoursPrefixe (a->fg);
      ParcoursPrefixe (a->fd);
   }
}
```

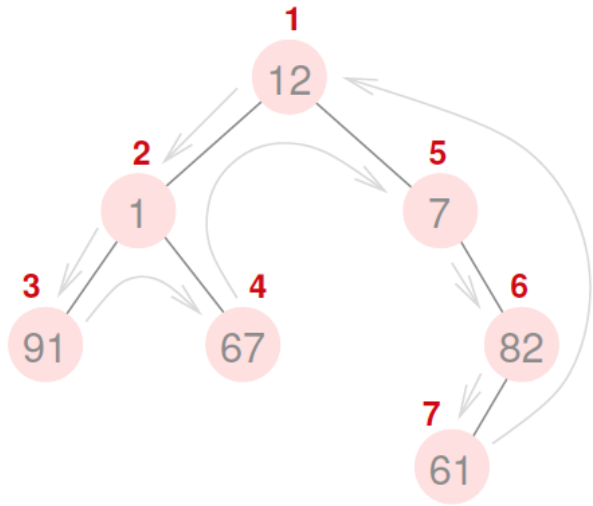


Figure:

la valeur du nœud courant est traitée avant les valeurs figurant dans ses sous-arbres. Si le traitement consiste simplement à afficher la valeur du nœud, le parcours préfixe du l'arbre ci-dessous produirait l'affichage **12 1 91 67 7 82 61.**

Parcours infixe :

la valeur du nœud courant est traitée après les valeurs figurant dans son sous-arbre gauche et avant les valeurs figurant dans son sous-arbre droit.

```
void ParcoursInfixe(ARBRE a)
{
   if (a != NULL)
   {
      ParcoursInfixe (a->fg);
      TraiterLaValeur (a->val);
      ParcoursInfixe (a->fd);
   }
}
```

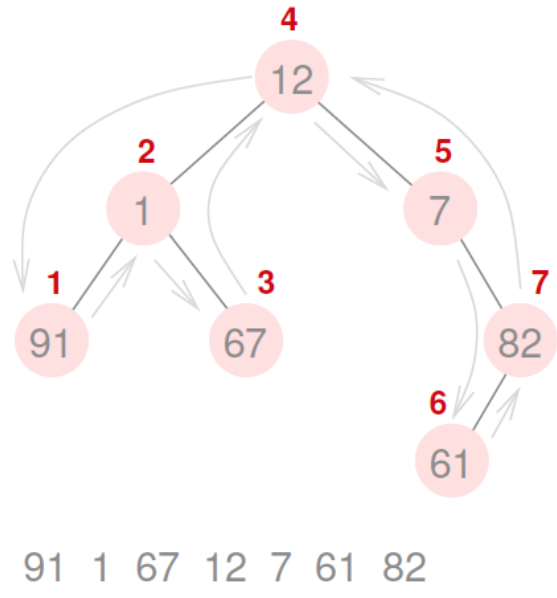


Figure:

Parcours postfixé :

la valeur du nœud courant est traitée après les valeurs de ses sous-arbres.

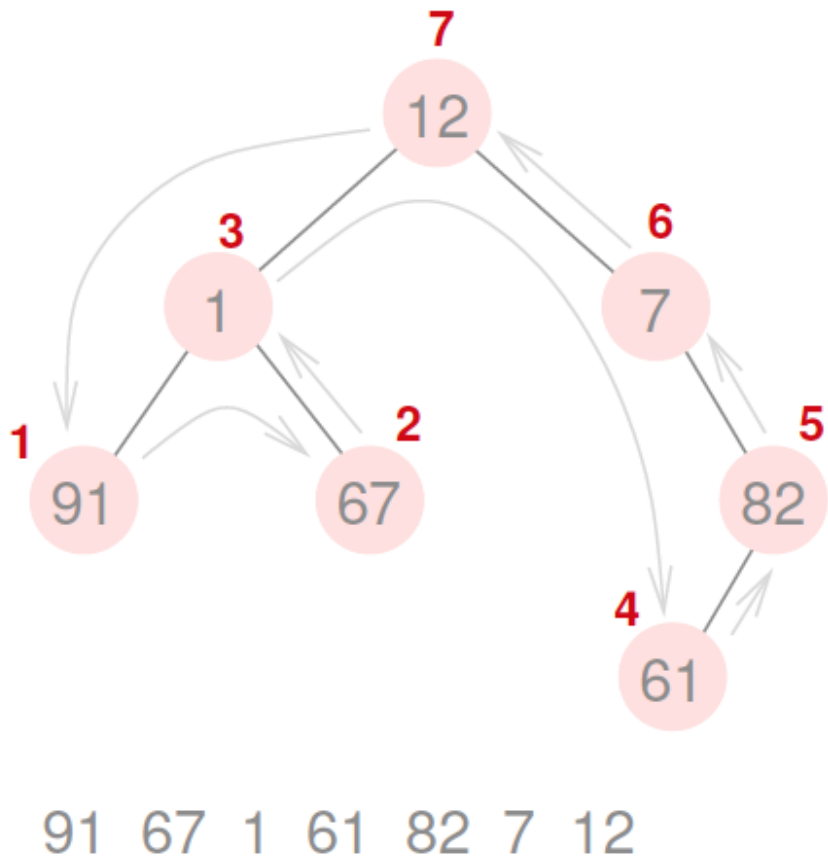


Figure:

Arbres binaires de recherche.

- Un arbre binaire de recherche (ou ABR) est une structure de donnée qui permet de représenter un ensemble de valeurs si l'on dispose d'une relation d'ordre sur ces valeurs.
- Les opérations caractéristiques sur les arbres binaires de recherche sont **l'insertion, la suppression, et la recherche d'une valeur.** Ces opérations sont peu coûteuses si l'arbre n'est pas trop déséquilibré.
- Soit E un ensemble muni d'une relation d'ordre, et a un arbre binaire portant des valeurs de E.
- a est un arbre binaire de recherche si pour tout nœud p de a, la valeur de p est plus grande que les valeurs figurant dans son sous-arbre gauche et plus petite que les valeurs figurant dans son sous-arbre droit.

Parcours postfixé :

la valeur du nœud courant est traitée après les valeurs de ses sous-arbres.

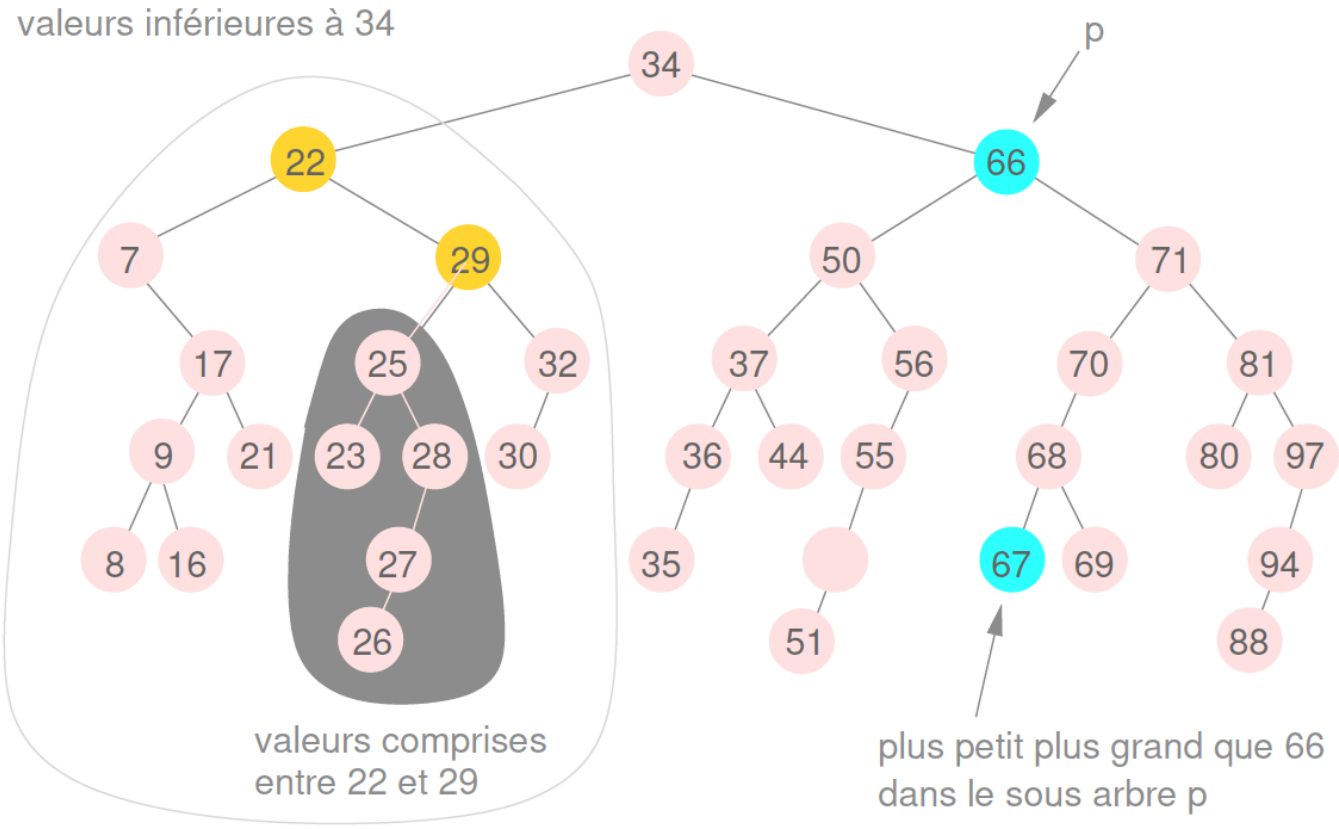
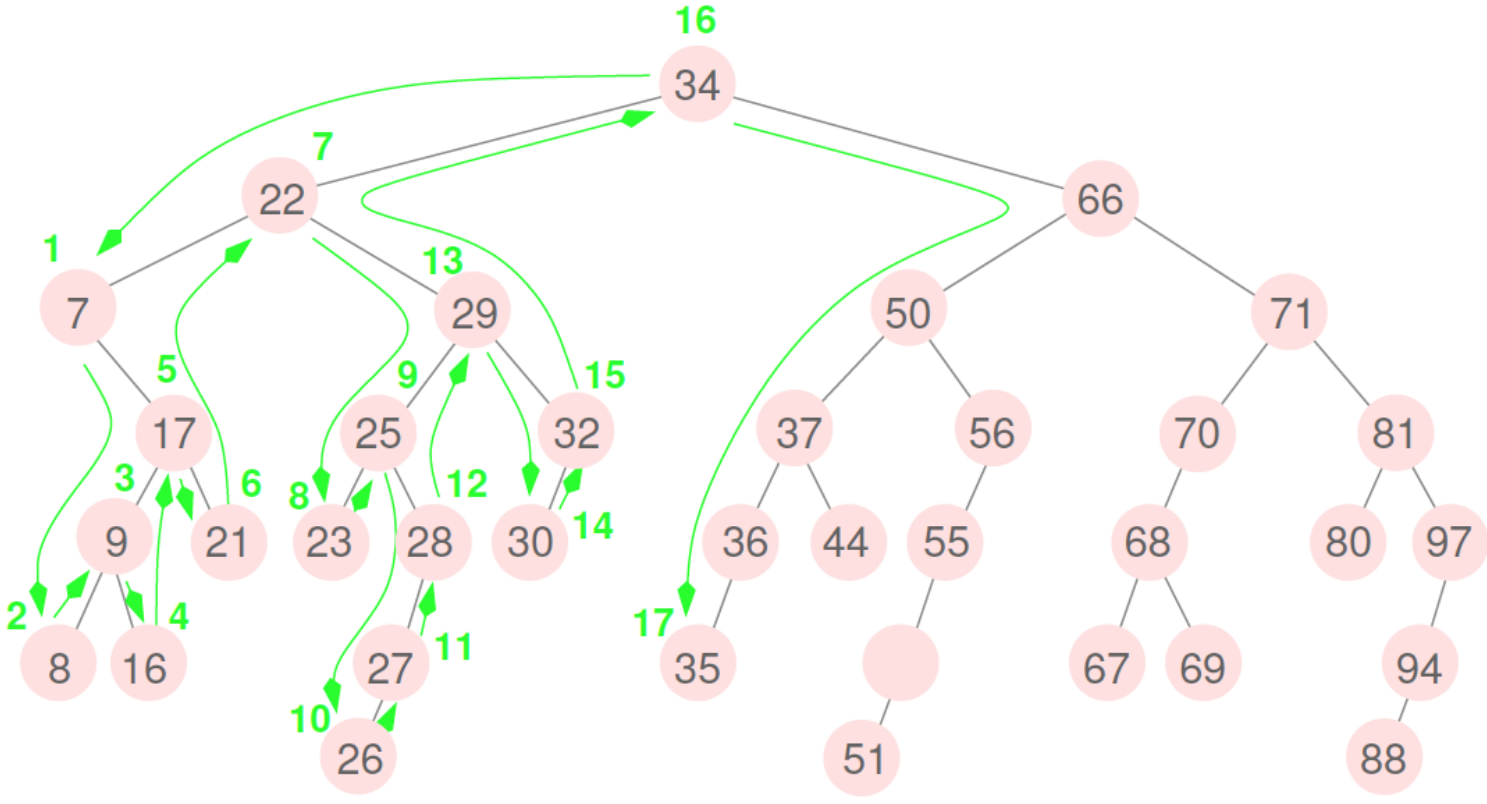


Figure:

Pour accéder à la clé la plus petite (resp. la plus grande) dans un ABR il suffit de descendre sur le fils gauche (resp. sur le fils droit) autant que possible. Le dernier nœud visité, qui n'a pas de fils gauche (resp. droit), porte la valeur la plus petite (resp. la plus grande) de l'arbre. Le parcours infixé de l'arbre produit la suite ordonnée des clés



7 8 9 16 17 21 22 23 25 26 27 28 29 30 32 34 35 36 37 44 50 51 52 55 56 66 ...

Figure:

Recherche d'une valeur.

- La recherche d'une valeur dans un ABR consiste à parcourir une branche en partant de la racine, en descendant chaque fois sur le fils gauche ou sur le fils droit suivant que la clé portée par le nœud est plus grande ou plus petite que la valeur cherchée.
- La recherche s'arête dès que la valeur est rencontrée ou que l'on a atteint l'extrémité d'une branche (le fils sur lequel il aurait fallu descendre n'existe pas).

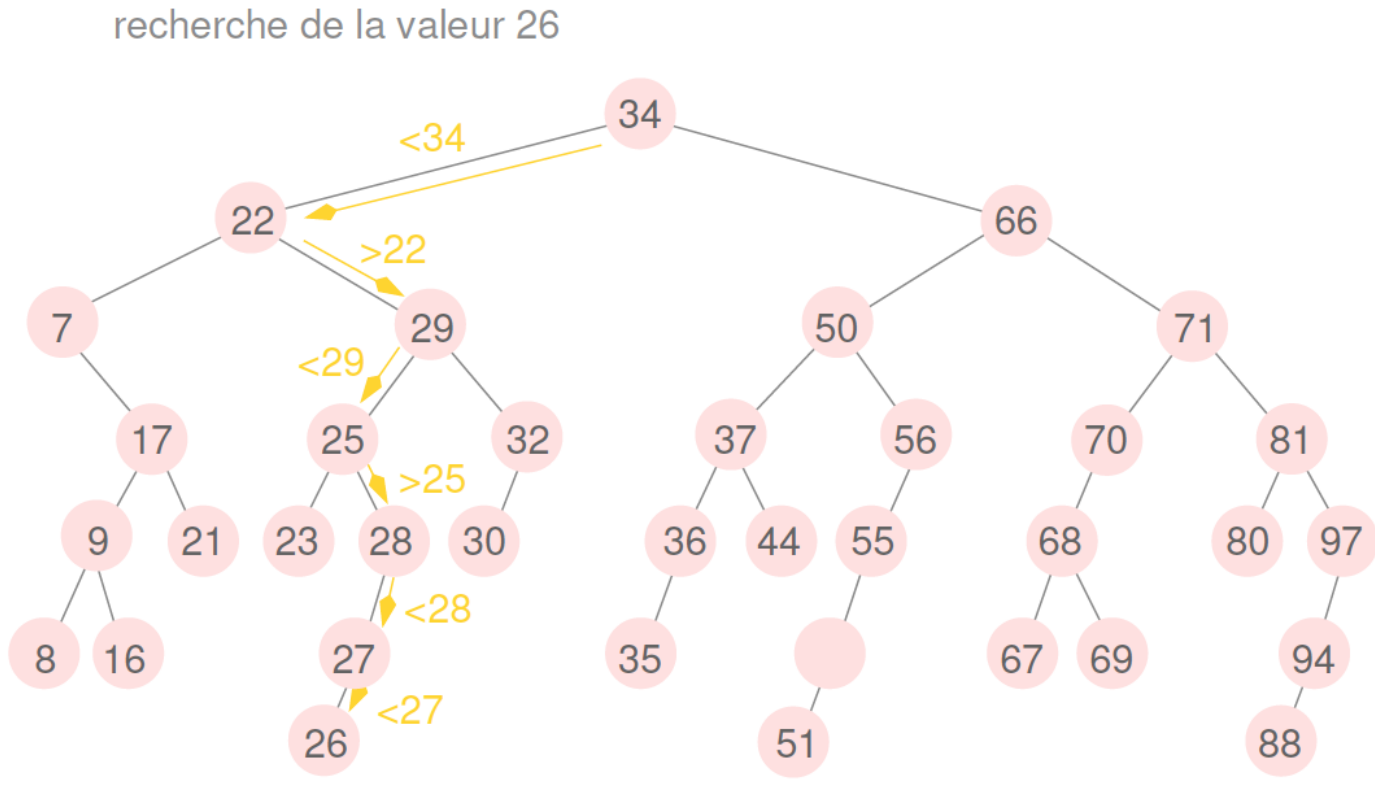

recherche de la valeur 26
34
<34
22
66
>22
7
29
50
71
<29
17
25
32
37
56
70
81
>25
9
21
23
28
30
36
44
55
68
80
97
<28
8
16
27
35
67
69
94
<27
26
51
88

Figure:

Arbres binaires de recherche,

```
ARBRE recherche (TYPE_VALEUR x, ARBRE a)
        {
           while ((a != NULL) && (x != a->val))
              if (x < a->val)
                 a = a->fg;
              else
                 a = a->fd;
           return a;
}
```

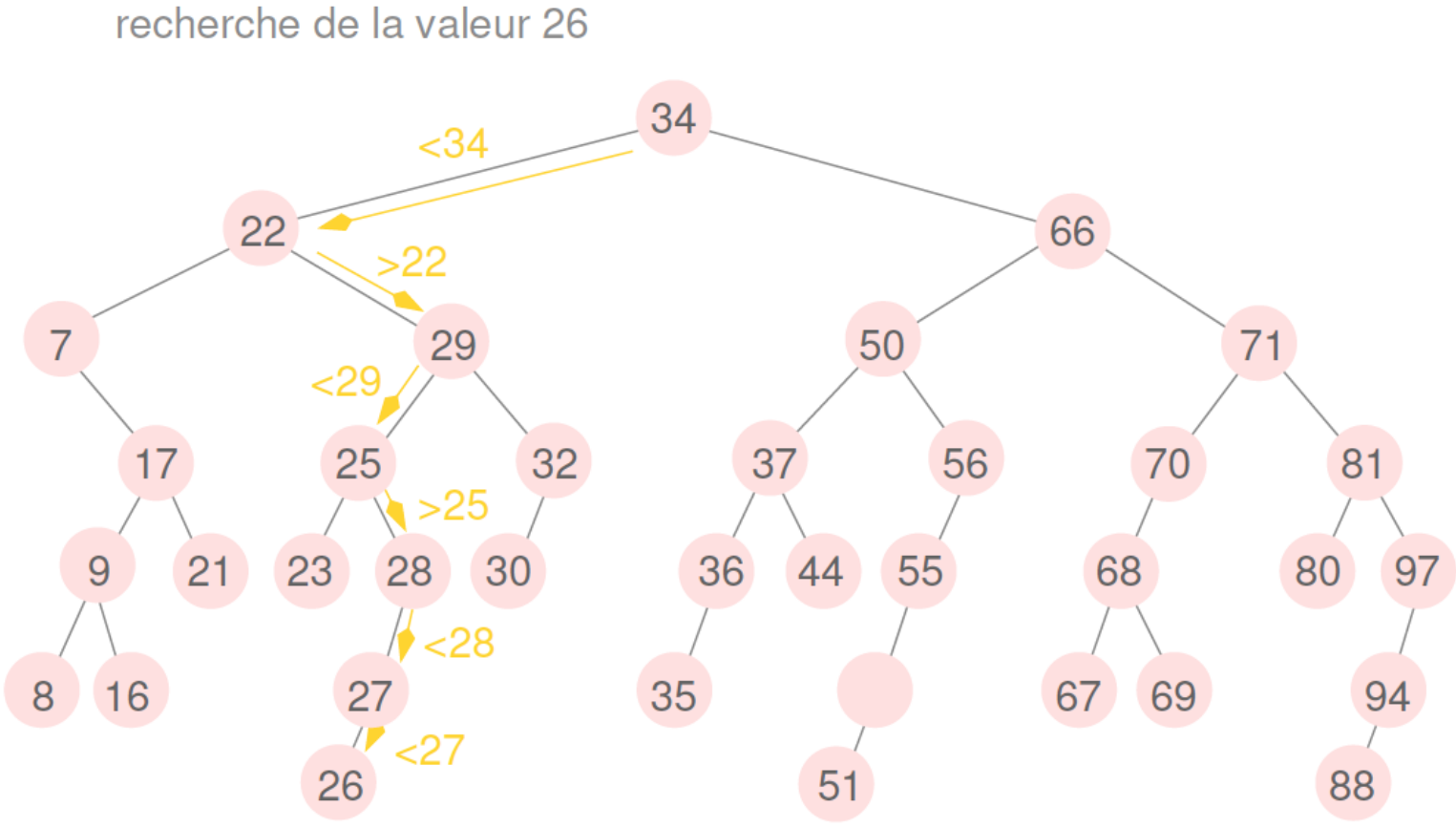
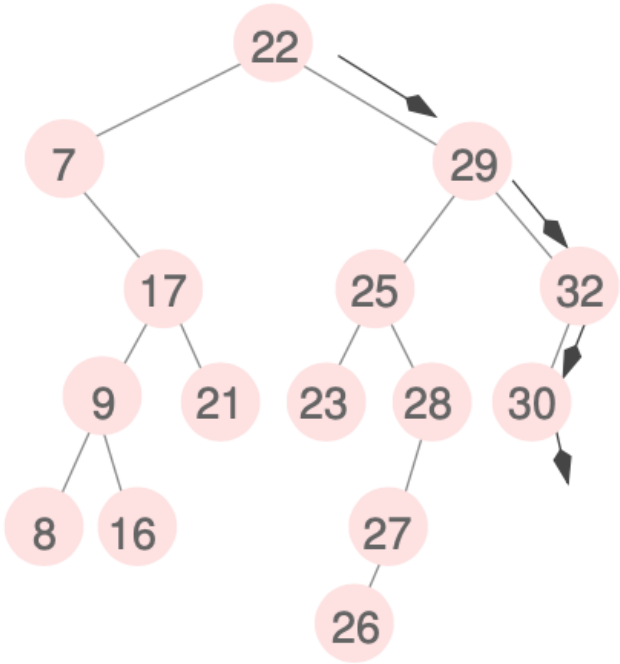

recherche de la valeur 26
34
<34
22
66
>22
7
29
50
71
<29
17
25
32
37
56
70
81
>25
9
21
23
28
30
36
44
55
68
80
97
<28
8
16
27
35
67
69
94
<27
26
51
88

Figure:

```
ARBRE recherche (TYPE_VALEUR x, ARBRE a)
        {
            while ((a != NULL) && (x != a->val))
                if (x < a->val)
                    a = a->fg;
                else
                    a = a->fd;
            return a;
}
```

Insertion d'une nouvelle valeur.

Le principe est le même que pour la recherche. Un nouveau nœud est créé avec la nouvelle valeur et inséré à l'endroit ou‘ la recherche s'est arêtée.


22
7
29
17
25
32
9
21
23
28
30
8
16
27
26

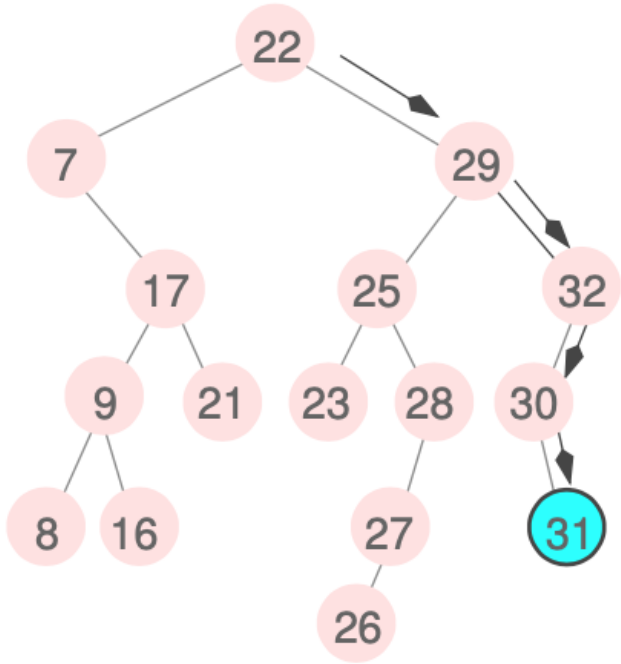

22
7
29
17
25
32
9
21
23
28
30
8
16
27
31
26

Figure:

Recherche du successeur d'un nœud.

- Etant donné un nœud p d'un arbre A, le successeur de p si il existe, est le nœud de A qui porte comme valeur la plus petite des valeurs qui figurent dans A et qui sont plus grandes que la valeur de p.
- Si p possède un fils droit, son successeur est le nœud le plus à gauche dans son sous-arbre droit (on y accède en descendant sur le fils gauche autant que possible)
- Si p n'a pas de fils droit alors sont successeur est le premier de ses ascendants tel que p apparaît dans son sous-arbre gauche.
- Si cet ascendant n'existe pas c'est que p portait la valeur la plus grande dans l'arbre.
- Remarquez qu'il est nécessaire d'avoir pour chaque nœud un lien vers son père pour mener à bien cette opération.

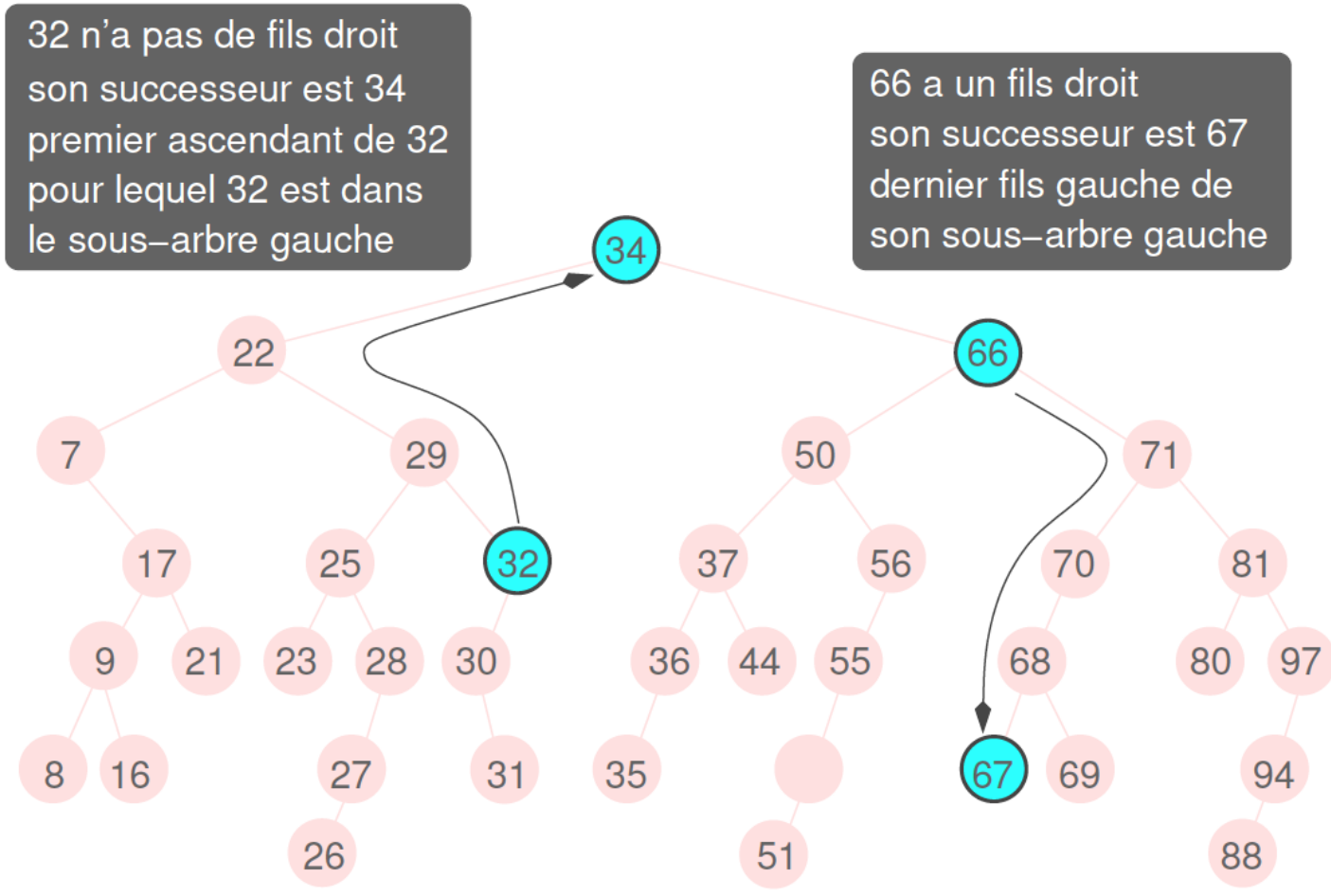
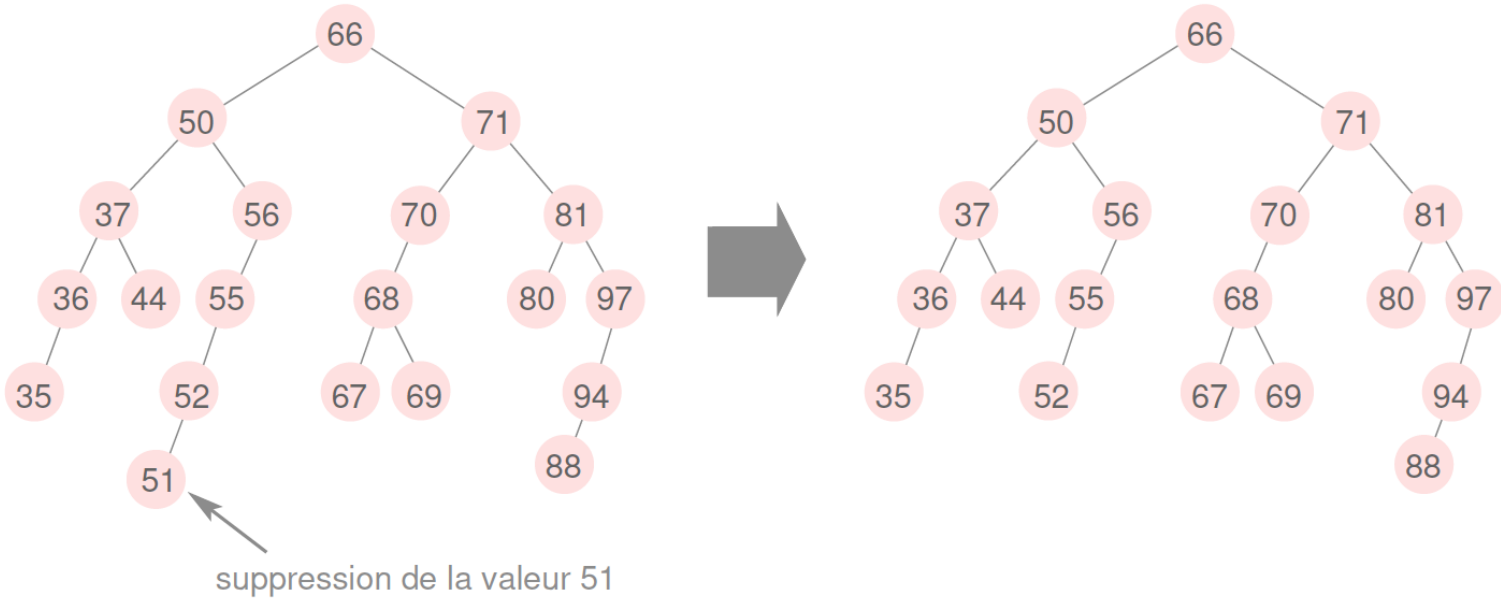

32 n'a pas de fils droit
son successeur est 34
premier ascendant de 32
pour lequel 32 est dans
le sous–arbre gauche
66 a un fils droit
son successeur est 67
dernier fils gauche de
son sous–arbre gauche
34
22
66
7
29
50
71
17
25
32
37
56
70
81
9
21
23
28
30
36
44
55
68
80
97
8
16
27
31
35
67
69
94
26
51
88

Figure:

Suppression d'un nœud

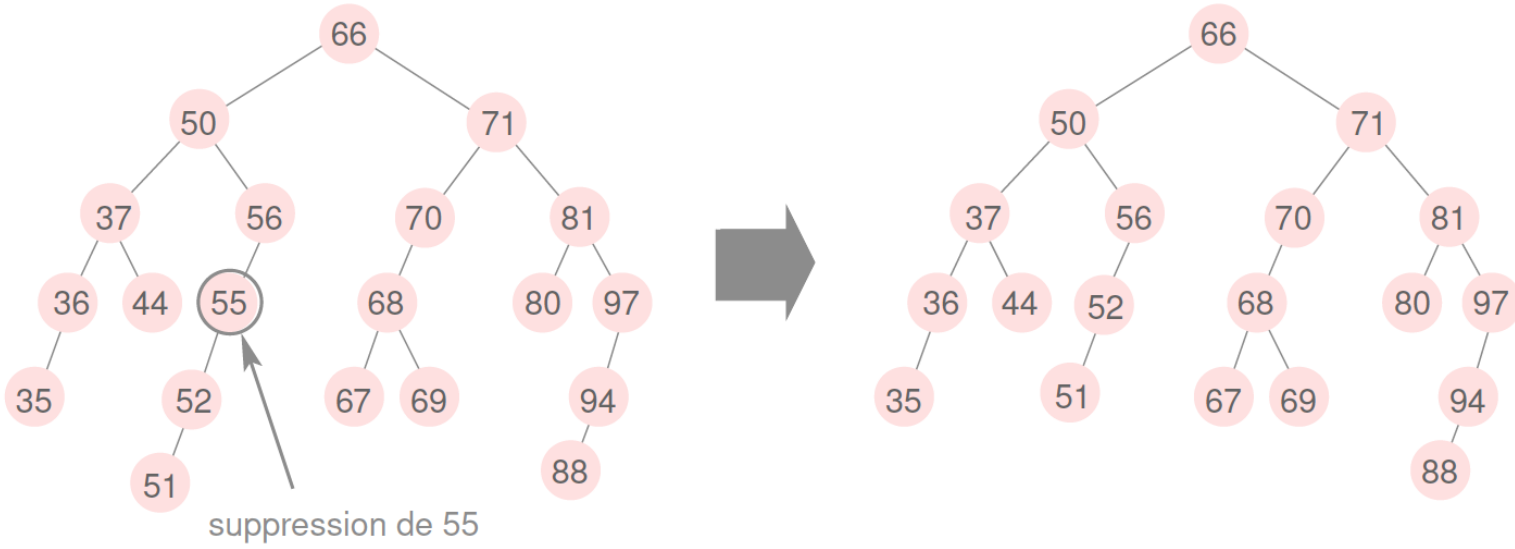
Suppression d'un noeud. L'opération dépend du nombre de fils du noeud à supprimer.



Cas 1 : le noeud à supprimer n'a pas de fils, c'est une feuille. Il suffit de *décrocher* le noeud de l'arbre, c'est-à-dire de l'enlever en modifiant le lien

Figure:

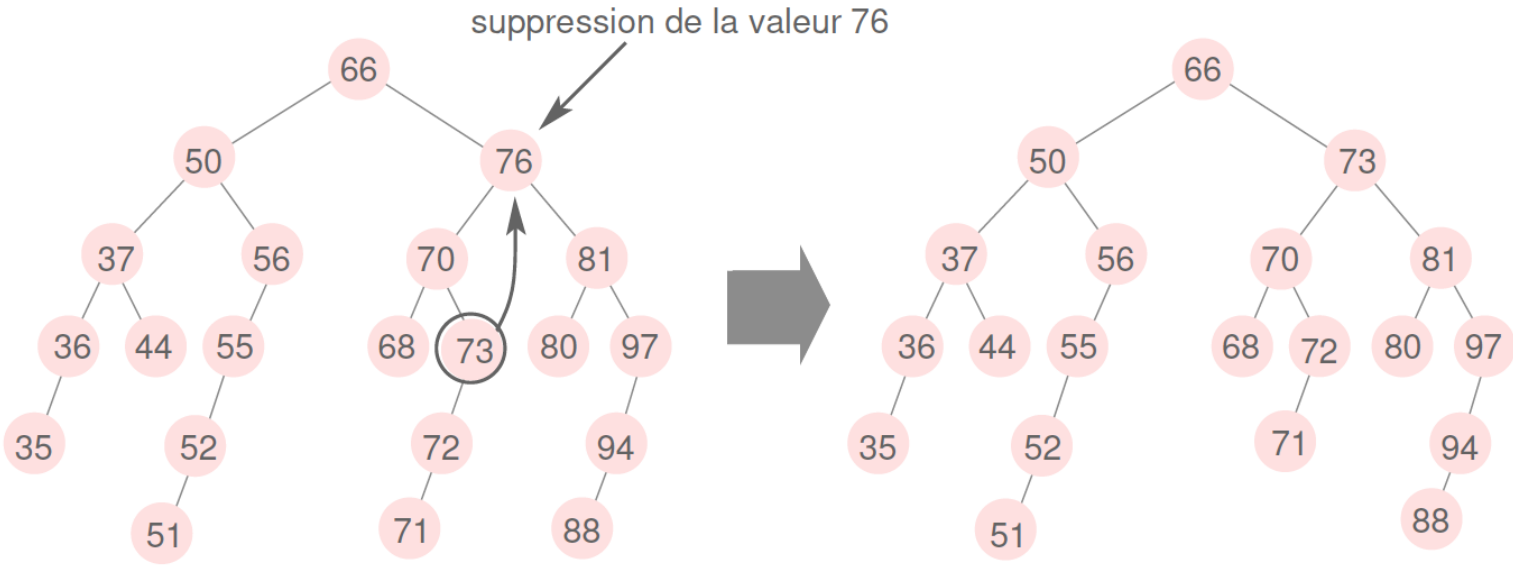
Cas 1 : le nœud à supprimer n'a pas de fils, c'est une feuille. Il suffit de décrocher le nœud de l'arbre, c'est-à-dire de l'enlever en modifiant le lien du père, si il existe, vers ce fils. Si le père n'existe pas l'arbre devient l'arbre vide.



Cas 2 : le noeud à supprimer a un fils et un seul. Le noeud est *décroché* de l'arbre en remplaçant ce fils par son fils unique dans le noeud père, si il existe. Si le père n'existe pas l'arbre est réduit au fils unique du noeud supprimé.

Figure:

Cas 2 : le nœud à supprimer a un fils et un seul. Le nœud est décroché de l'arbre en remplaçant ce fils par son fils unique dans le nœud père, si il existe. Si le père n'existe pas l'arbre est réduit au fils unique du nœud supprimé.



Cas 3 : le noeud à supprimer p a deux fils. On va chercher le noeud q qui porte la plus grande valeur qui figure dans son fils gauche (ou indifféremment le noeud de plus petite valeur qui figure dans son fils droit). Il suffit ensuite

Figure:

Cas 3 : le nœud à supprimer p a deux fils.

- On va chercher le nœud q qui porte la plus grande valeur qui figure dans son fils gauche (ou indifféremment le le nœud de plus petite valeur qui figure dans son fils droit).
- Il suffit ensuite de substituer la valeur de q à la valeur de p et de décrocher le nœud q. Puisque le nœud q a la valeur la plus grande dans le fils gauche, il n'a donc pas de fils droit, et peut être décroché comme on l'a fait dans les cas précédents.

Arbres généraux ou n-aires

- Un arbre n-aire est un arbre dans lequel le nombre de fils d'un nœud n'est pas limité.
- On les représente avec des pointeurs, en liant les fils d'un même nœud entre eux, comme dans une liste chaînée. Ainsi, à partir d'un nœud quelconque, on accède à son fils ainé (la liste de ses fils) et à son frère.

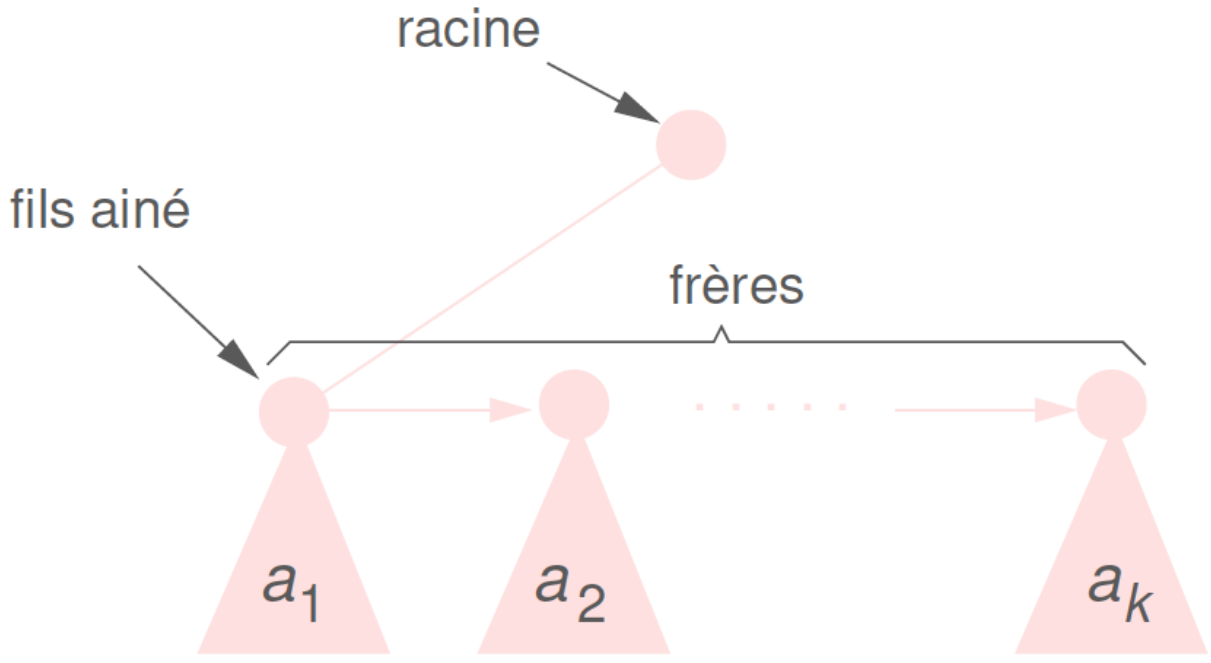


Figure: