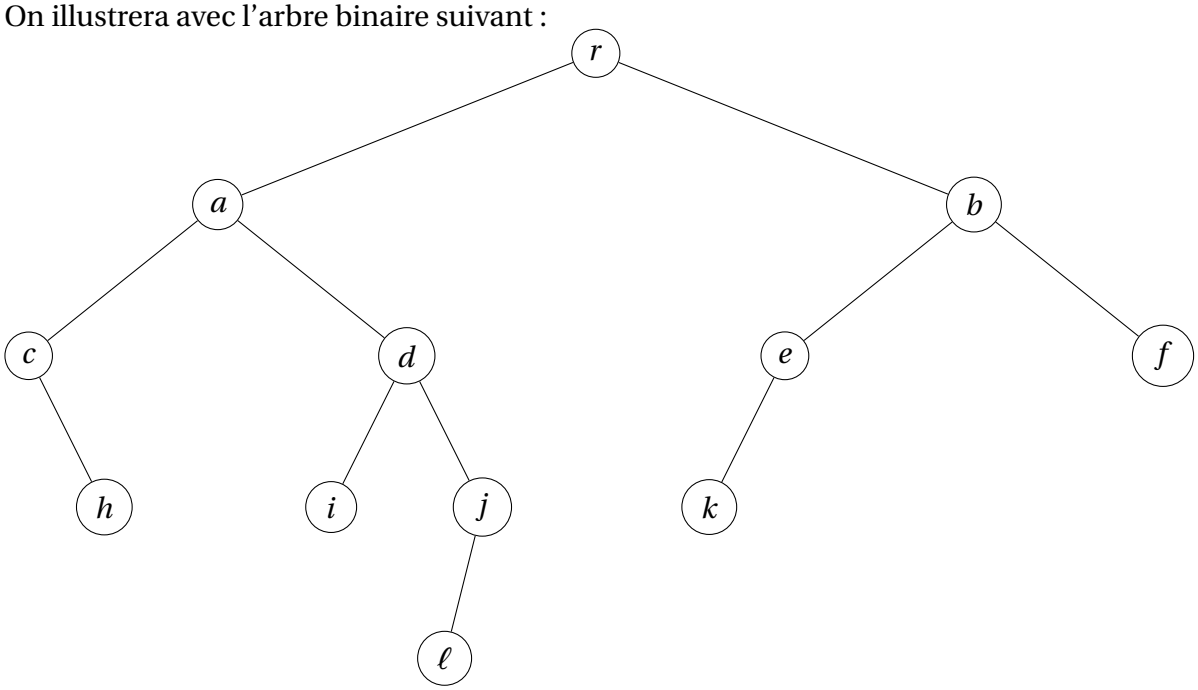
Parcours d'un arbre binaire

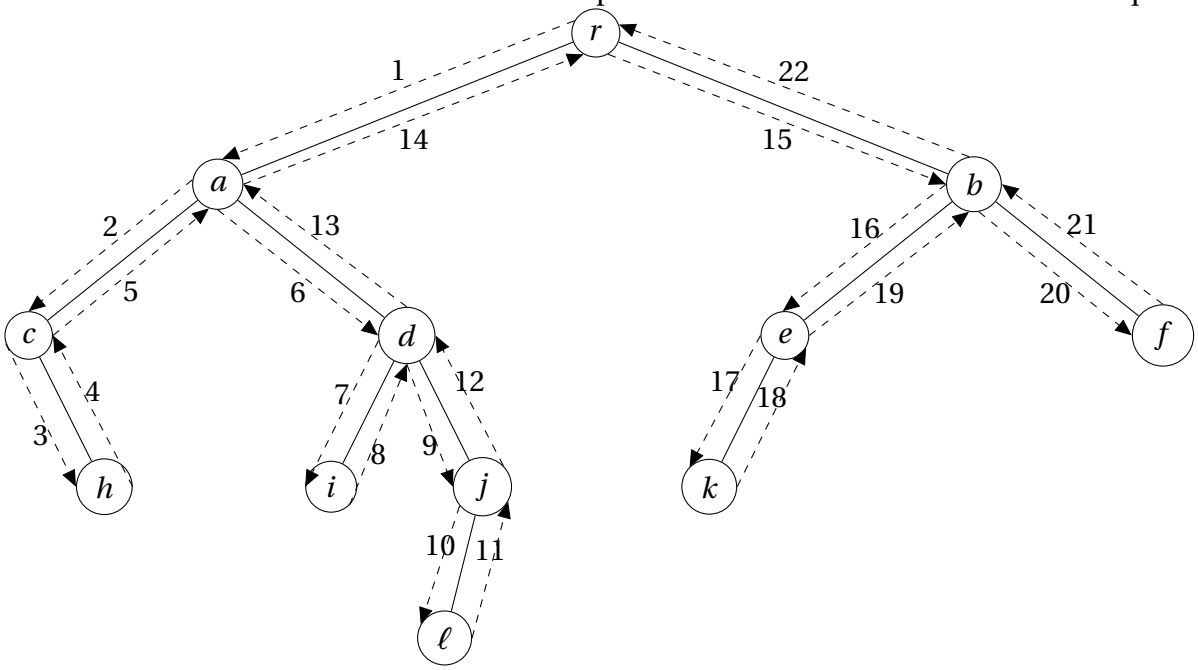
Un arbre binaire est un arbre avec racine dans lequel tout noeud a au plus deux fils : un éventuel fils gauche et un éventuel fils droit.

On illustrera avec l'arbre binaire suivant :



1 Balade autour de l'arbre

On se balade autour de l'arbre en suivant les pointillés dans l'ordre des numéros indiqués :



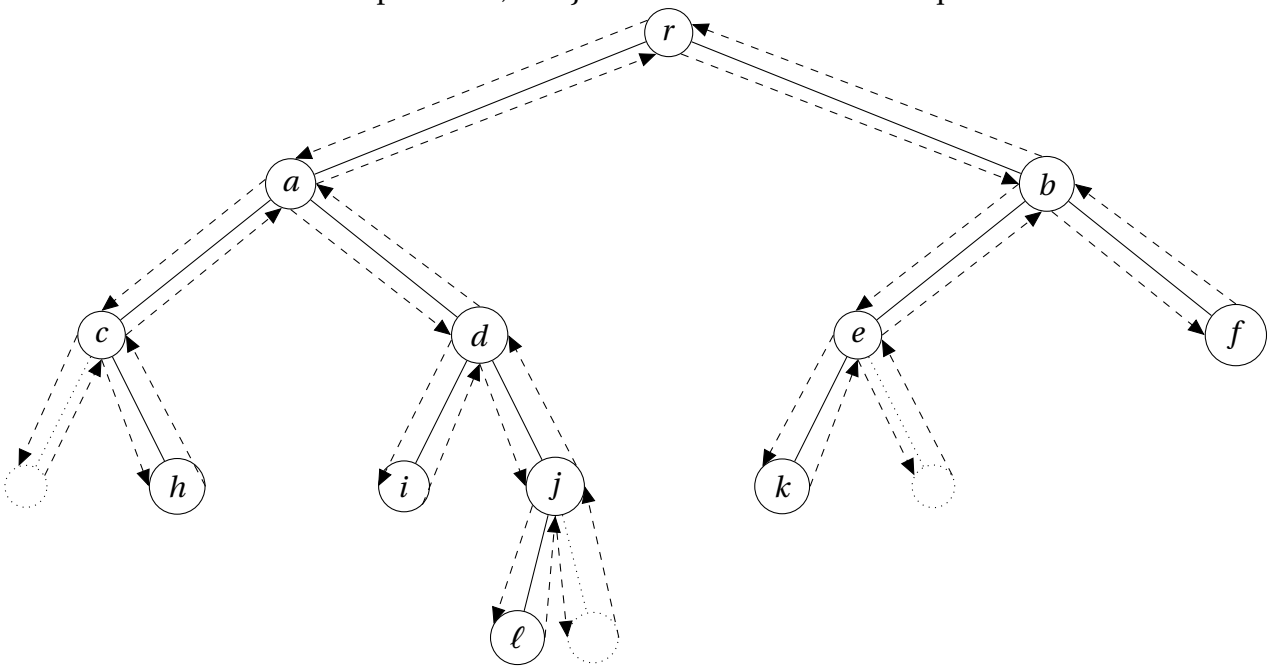
1.1 Première définition des trois parcours

A partir de ce contour, on définit trois parcours des sommets de l'arbre :

1. l'ordre préfixe : on liste chaque sommet la première fois qu'on le rencontre dans la balade. Ce qui donne ici : ...
2. l'ordre postfixe : on liste chaque sommet la dernière fois qu'on le rencontre. Ce qui donne ici : ...
3. l'ordre infixe : on liste chaque sommet ayant un fils gauche la seconde fois qu'on le voit et chaque sommet sans fils gauche la première fois qu'on le voit. Ce qui donne ici : ...

1.2 Seconde définition des trois parcours

Dans la balade schématisée plus haut, on ajoute les fils fantômes manquants :



On peut ainsi considérer qu'on passe une fois à gauche de chaque noeud (en descendant), une fois en-dessous de chaque noeud, une fois à droite de chaque noeud (en remontant).
Vérifier, sur l'exemple, que chacun des ordres préfixe, infixe, postfixe est obtenu en listant tous les mots :
– soit lorsqu'on passe à leur gauche,
– soit lorsqu'on passe à leur droite,
– soit lorsqu'on passe en-dessous.

2 Algorithmes récursifs

Pour chacun des parcours définis ci-dessus (postfixe, infixe, préfixe), définir récursivement le parcours.

3 Représentation en machine

Chaque nœud de l'arbre T est representé par un objet ayant un champ clef (des valeurs à trier par exemple), un champ père, un champ fils_gauche, un champ fils_droit (stockent des pointeurs).
Lorsque père[x]=NIL, x est la racine de l'arbre. Lorsque x n'a pas de fils gauche, fils_gauche[x]=NIL (idem pour fils_droit). La racine de l'arbre T est pointée par l'attribut racine[T]. Lorsque racine[T]=NIL, l'arbre est vide.

4 Complexité d'un parcours infixe

Vérifier qu'avec n noeuds, le parcours infixe

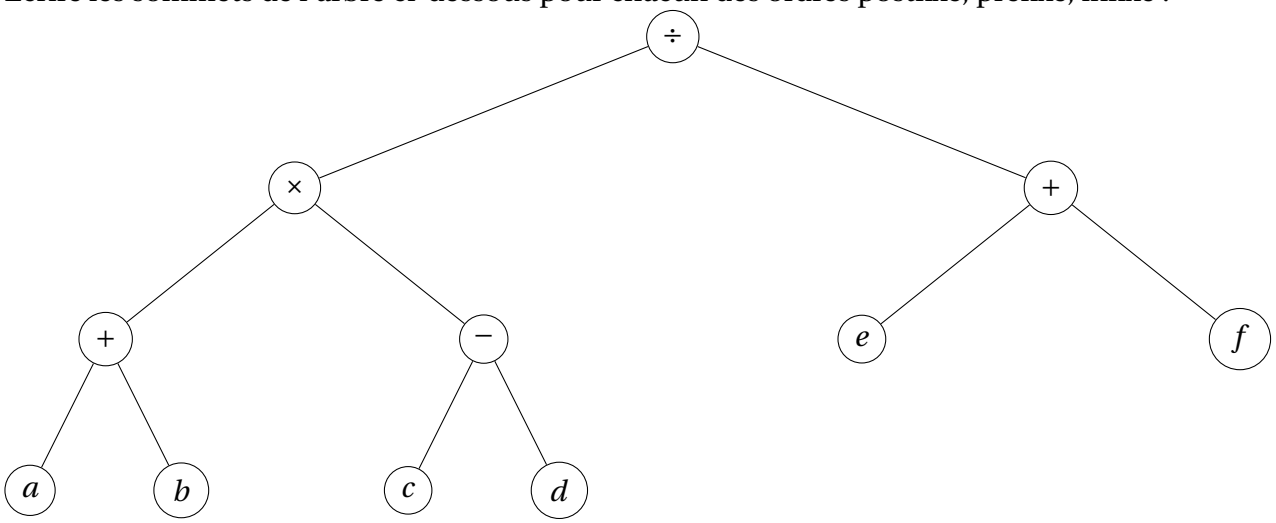
Pseudo-code

```
Parcours_Infixe(arbre binaire T de racine x)
        Si x distinct de NIL *****temps constant T(0)=c pour un sous-arbre vide
        alors
        Parcours_Infixe( arbre de racine fils_gauche[x]) *****temps T(k)
        Afficher clef[x]  *****temps constant d
        Parcours_Infixe( arbre de racine fils_droit[x]) *****temps T(n-k-1)
        FinSi
```

prend un temps en $\Theta(n)$ (établir avec les notations suggérées ci-dessus : $T(n) = (c+d)n+c$)

5 La notation polonaise inverse

Écrire les sommets de l'arbre ci-dessous pour chacun des ordres postfixe, préfixe, infixe :



Pour le parcours infixe, on ajoute la convention suivante : on ajoute une parenthèse ouvrante à chaque fois qu'on entre dans un sous-arbre et on ajoute une parenthèse fermante lorsqu'on quitte ce sous-arbre.

6 Le tri du bijoutier

On dispose d'une liste de nombres. Par exemple, la liste 7, 9, 3, 5, 4, 1, 8. On associe à chaque élément n de la liste un nœud v (initialisation : père[v]=NIL, fils_gauche[v]=NIL, fils_droit[v]=NIL, clef[v]=n).

Pseudo-code

```
Arbre_Insérer(Arbre T, noeud z)
        y:=NIL
        x:=racine[T]

        TantQue x distinct de NIL faire
                y:=x
                Si clef[z]<clef[x]
                alors x:=fils_gauche[x]
                sinon x:=fils_droit[x]
                FinSi
        FinTantQue

        père[z]:=y
        Si y=NIL
        alors racine[T]:=z
        sinon
                Si clef[z]<clef[y]
                alors fils_gauche[y]:=z
                sinon fils_droit[y]:=z
                FinSi
         FinSi
```

1. Dresser l'arbre obtenu en appliquant l'algorithme Arbre_Insérer aux éléments de la liste (dans l'ordre de la liste) en partant d'un arbre vide pour le premier élément, chaque appel à l'algorithme modifiant l'arbre.
2. L'un des parcours postfixe, infixe, préfixe de la liste trie la liste. Lequel ?
3. Dans la construction de l'arbre pour une liste de n nombres, quel est le nombre de comparaisons effectuées dans le pire des cas ?
4. Quel est le nombre de comparaisons effectuées si l'arbre final est un arbre binaire complet (arbre binaire dans lequel tout nœud autre qu'une feuille a deux fils et dans lequel les feuilles sont tous des nœuds de même profondeur).

7 Références

1. Introduction à l'algorithmique. Auteurs : Cormen, Leiserson, Rivest, Stein. Edition française : Dunod 2002.

   Plus de 1100 pages sur les algorithmes et les structures de référence. Le chapitre 12 concerne les arbres binaires de recherche.

2. Un article de Jean-Claude Oriol sur le site de l'apmep avec un passage sur le tri du bijoutier : http://www.apmep.asso.fr/spip.php?article3405
3. Le cours de Pierre Audibert (Paris 8) en ligne : http://www.ai.univ-paris8.fr/~audibert/ens/06-ARBREBINX.pdf