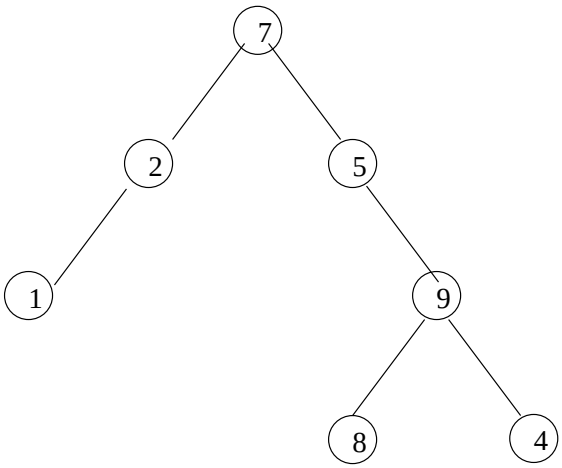
Arbres lexicographiques
Fils gauche frère droit

1. **Chemins de la racine aux feuilles**
   On considère un arbre binaire de hauteur inférieure à une constante MAX. Ecrire une fonction `void Chemin(Arbre a)` qui permet l'affichage de tous les chemins menant de la racine aux feuilles dans l'ordre d'un parcours en profondeur gauche.
   Pour l'arbre :



   On affichera

```
7 2 1
7 5 9 8
7 5 9 4
```

   On appellera une fonction annexe `AfficheCheminAux(Arbre A, int Buffer M[],int indice)`
   où `Buffer` est un tableau de taille `MAX` déclaré dans la fonction `Chemin` et `indice` indique la première case vide de `Buffer`.

2. **Arbre fils gauche frère droit -lexique de mots**
   Un arbre peut être utilisé pour représenter une collection de mots de sorte qu'il soit très efficace de vérifier si une séquence donnée de caractères est un mot valide ou pas. Une des façons de représenter une

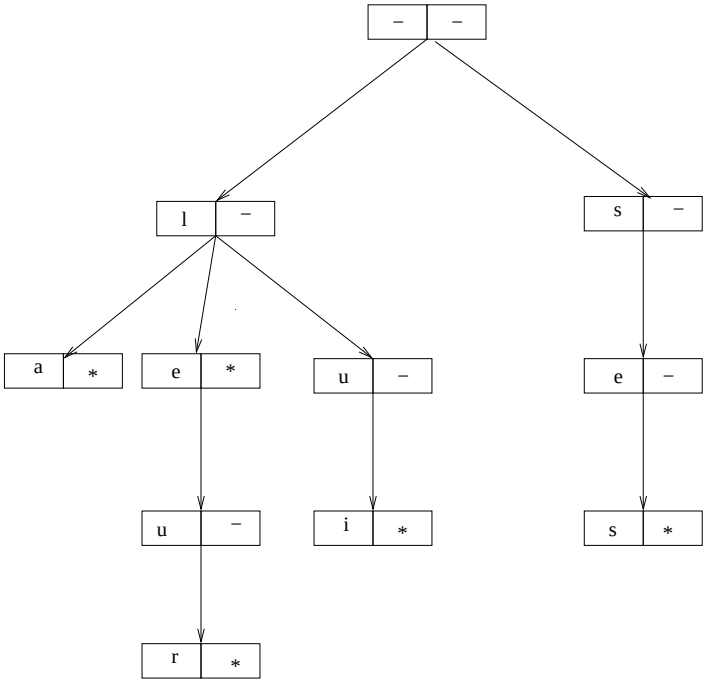


Figure 1: Trie.

telle collection utilise un arbre que l'on appelle Trie dont chaque noeud peut avoir autant d (voir la figure 1 où * indique la fin d'un mot de la collection ici {`la, le, leur, lui, ses`}).

Le problème du nombre variable de fils d'un nœud peut être résolu par la représentation est appelée fils gauche-frère droit et dont le type est défini par :

```
typedef struct noeud {
  char lettre;
  struct noeud *filsGauche, *frereDroit;
} Noeud,*Arbre;
```

Un exemple est donné par la figure 2 ci-dessous pour la même collection de mots que précédemment.

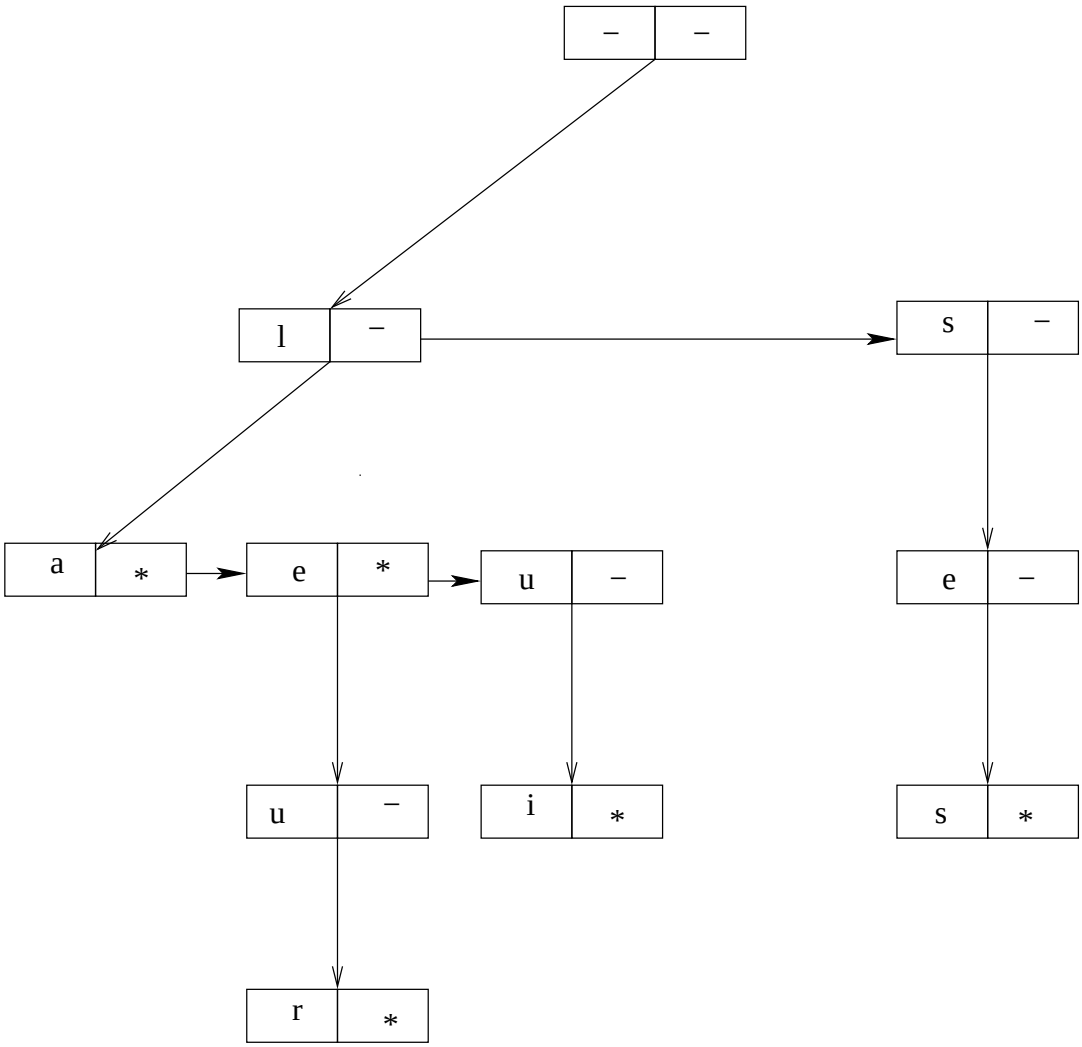


Figure 2: Arbre fils-gauche-frère-droit.

Les enfants d'un nœud sont reliés dans une liste chaînée, le champs `filsGauche` du nœud père pointe sur la tête de la liste de ses enfants.

Écrire les fonctions de manipulations d'un tel lexique:

- Recherche
- Nombre de mots
- affichage
- ajout

On considérera qu'un mot est un chaîne (de moins de MAX caractères) au sens du C (telle que saisie par `scanf("%s",Mot);`). On placera le marqueur '\0' comme feuille de l'arbre, ce qui évite d'utiliser un champs fin de mot.