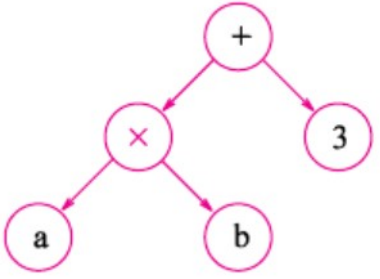
Algorithmie

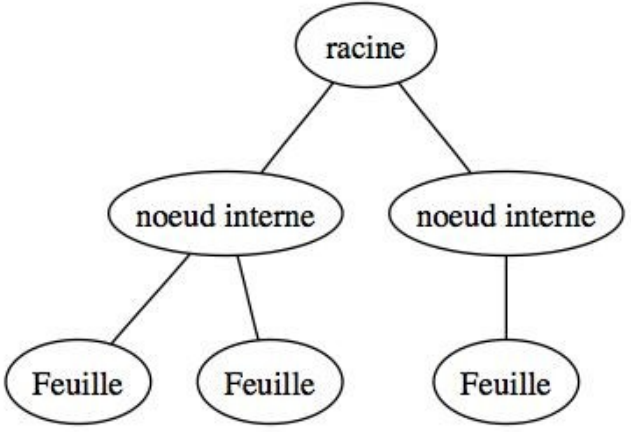
Les arbres

Les arbres

- <u>Présentation :</u>
  Un structure arborescente permet de présenter une information organisée de façon hiérarchique (poules pour les coupes du monde de rugby, arbres généalogiques, systèmes de fichiers,…) que l'on présente par des arbres.
  Il existe plusieurs formes d'arbres mais ce cours se consacre aux arbres binaires



a x b + 3



Arbre enraciné

Les arbres binaires

Un organisateur de tournoi de rugby recherche la meilleure solution pour afficher les potentiels quarts de final, demi-finales et finale :

Au départ nous avons 4 poules de 4 équipes. Les 4 équipes d'une poule s'affrontent dans un mini championnat (3 matchs par équipe). À l'issue de cette phase de poule, les 2 premières équipes de chaque poule sont qualifiées pour les quarts de finale.

Dans ce qui suit, on désigne les 2 qualifiés par poule par :

- Poule 1 => 1er Eq1 ; 2e Eq8
- Poule 2 => 1er Eq2 ; 2e Eq7
- Poule 3 => 1er Eq3 ; 2e Eq6
- Poule 4 => 1er Eq4 ; 2e Eq5

En quart de final, on va avoir :

- quart de finale 1 => Eq1 contre Eq5
- quart de finale 2 => Eq2 contre Eq6
- quart de finale 3 => Eq3 contre Eq7
- quart de finale 4 => Eq4 contre Eq8

Les arbres binaires

Un organisateur de tournoi de rugby recherche la meilleure solution pour afficher les potentiels quarts de final, demi-finales et finale :

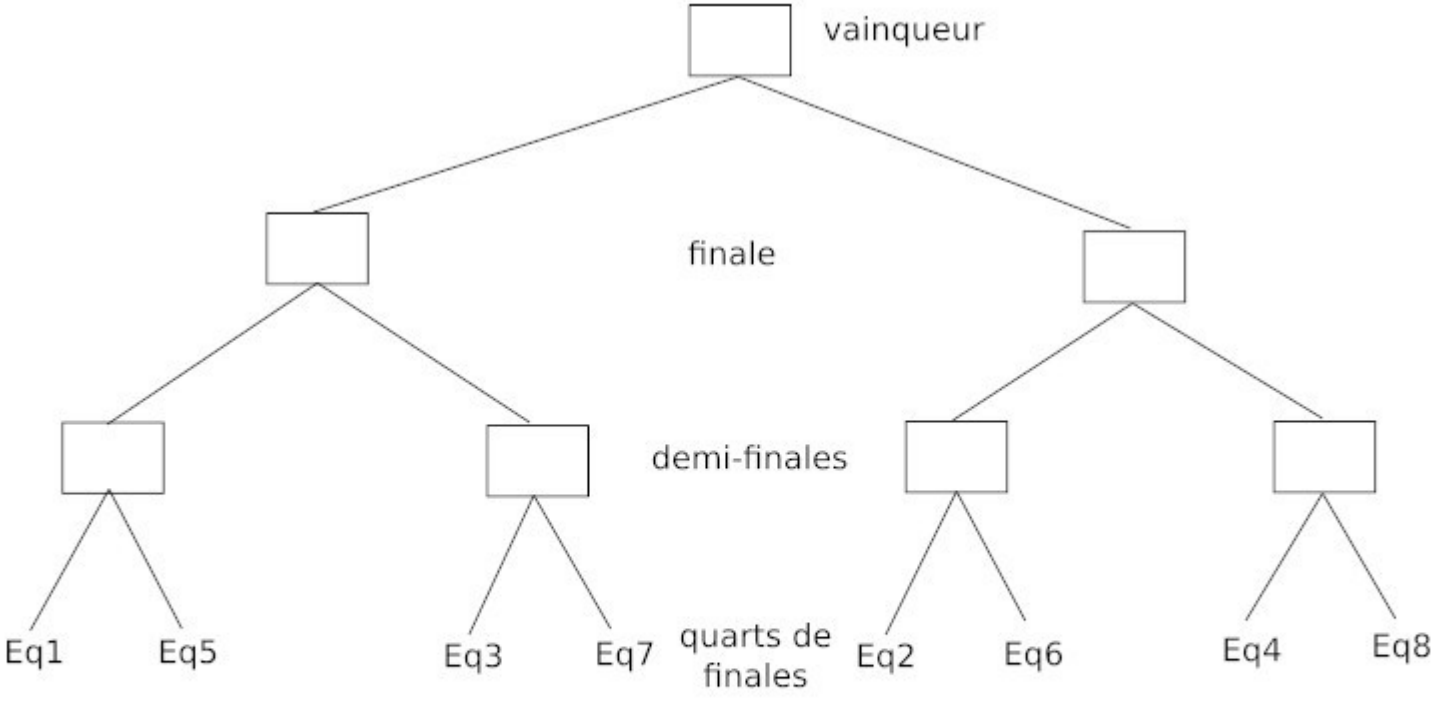
Au départ nous avons 4 poules de 4 équipes. Les 4 équipes d'une poule s'affrontent dans un mini championnat (3 matchs par équipe). À l'issue de cette phase de poule, les 2 premières équipes de chaque poule sont qualifiées pour les quarts de finale. Pour les demi-finales on aura :

demi-finale 1 => vainqueur quart de finale 1 contre vainqueur quart de finale 3
demi-finale 2 => vainqueur quart de finale 2 contre vainqueur quart de finale 4

L'organisateur du tournoi affiche les informations ci-dessus le jour du tournoi. Malheureusement, la plupart des spectateurs se perdent quand ils cherchent à déterminer les potentielles demi-finales (et ne parlons pas de la finale !)

Les arbres binaires

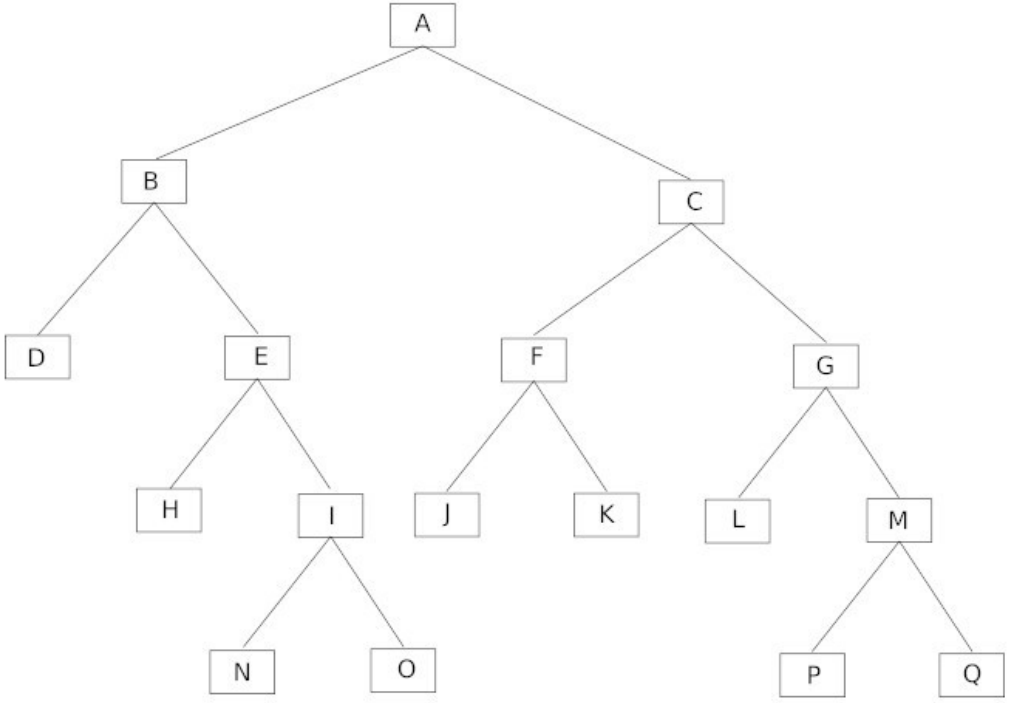
Pourtant, un simple graphique aurait grandement simplifié les choses :



Poule 1 => 1er Eq1 ; 2e Eq8
Poule 2 => 1er Eq2 ; 2e Eq7
Poule 3 => 1er Eq3 ; 2e Eq6
Poule 4 => 1er Eq4 ; 2e Eq5

Les arbres binaires

Un arbre binaire a, au maximum, 2 branches qui partent d'un élément



Les arbres binaires

- chaque élément de l'arbre est appelé nœud (par exemple : A, B, C, D,...,P et Q sont des nœuds)
- le nœud initial (ici A) est appelé nœud racine ou plus simplement racine
- On dira que le nœud E et le nœud D sont les fils du nœud B. On dira que le nœud B est le père des nœuds E et D
- Dans un arbre binaire, un nœud possède au plus 2 fils
- Un nœud n'ayant aucun fils est appelé feuille (exemples : D, H, N, O, J, K, L, P et Q sont des feuilles)
- À chaque nœud d'un arbre binaire, on associe une clé (on peut aussi utiliser le terme "valeur" à la place de clé), un "sous-arbre gauche" et un "sous-arbre droit" (exemple : à partir du nœud ayant pour clé C on va trouver un sous-arbre gauche composé des nœuds F, J et K et un sous-arbre droit composé des nœuds G, L, M, P et Q)
- Un arbre (ou un sous-arbre) vide est noté NIL (NIL est une abréviation du latin nihil qui veut dire "rien"). Par exemple, le sous-arbre gauche du nœud D est NIL (même chose pour son sous-arbre droit d'ailleurs puisque D est une feuille).
- On appelle arête le segment qui relie 2 nœuds.
- On appelle taille d'un arbre le nombre de nœuds présents dans cet arbre
- On appelle profondeur d'un nœud ou d'une feuille dans un arbre binaire le nombre de nœuds du chemin qui va de la racine à ce nœud. La racine d'un arbre est à une profondeur 1, et la profondeur d'un nœud est égale à la profondeur de son prédécesseur plus 1. Si un nœud est à une profondeur p, tous ses successeurs sont à une profondeur p+1. Exemples : profondeur de B = 2 ; profondeur de I = 4 ; profondeur de P = 5 ATTENTION : on trouve aussi dans certains livres la profondeur de la racine égale à 0 (on trouve alors : profondeur de B = 1 ; profondeur de I = 3 ; profondeur de P = 4). Les 2 définitions sont valables, il faut juste préciser si vous considérez que la profondeur de la racine est de 1 ou de 0.
- On appelle hauteur d'un arbre la profondeur maximale des nœuds de l'arbre. Exemple : la profondeur de P = 5, c'est un des nœuds les plus profond, donc la hauteur de l'arbre est de 5. ATTENTION : comme on trouve 2 définitions pour la profondeur, on peut trouver 2 résultats différents pour la hauteur : si on considère la profondeur de la racine égale à 1, on aura bien une hauteur de 5, mais si l'on considère que la profondeur de la racine est de 0, on aura alors une hauteur de 4

Arbre binaire de recherche (ABR)

Arbre binaire de recherche

Un arbre binaire de recherche est un cas particulier d'arbre binaire. Pour avoir un arbre binaire de recherche :

- il faut avoir un arbre binaire !
- il faut que les clés de nœuds composant l'arbre soient ordonnables (on doit pouvoir classer les nœuds, par exemple, de la plus petite clé à la plus grande)
- soit x un nœud d'un arbre binaire de recherche. Si y est un nœud du sous-arbre gauche de x, alors il faut que y.clé $\leqslant$ x.clé. Si y est un nœud du sous-arbre droit de x, il faut alors que x.clé $\leqslant$ y.clé

Exercice 1

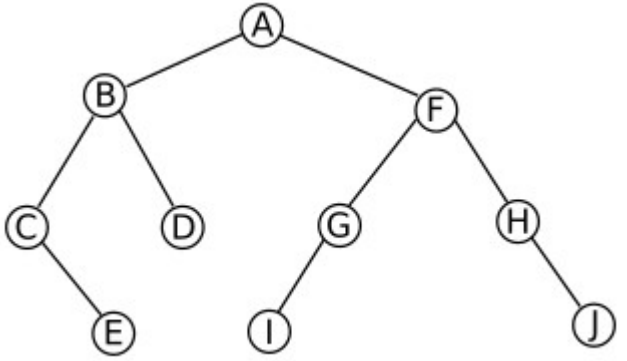
Étudiez cet algorithme :

```
VARIABLE
T : arbre
x : nœud


DEBUT
HAUTEUR(T) :
  si T ≠ NIL :
    x ← T.racine
    renvoyer 1 + max(HAUTEUR(x.gauche), HAUTEUR(x.droit))
  sinon :
    renvoyer 0
  fin si
FIN
```

N.B. la fonction max renvoie la plus grande valeur des 2 valeurs passées en paramètre (exemple : max(5,6) renvoie 6)

Cet algorithme est loin d'être simple, n'hésitez pas à écrire votre raisonnement sur une feuille de brouillon. Vous pourrez par exemple essayer d'appliquer cet algorithme sur l'arbre binaire ci-contre.

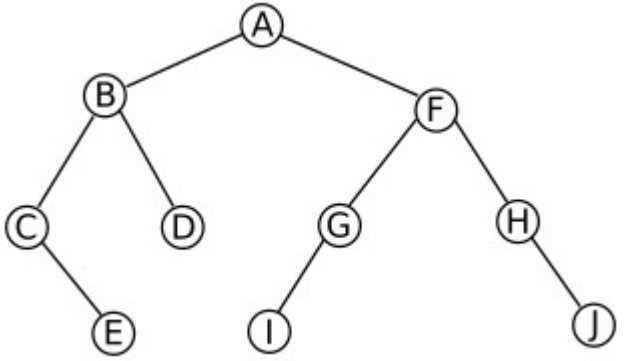


Exercice 2

Cet algo permet de calculer la taille d'un arbre

```
VARIABLE
T : arbre
x : noeud

DEBUT
TAILLE(T) :
  si T ≠ NIL :
    x ← T.racine
    renvoyer 1 + TAILLE(x.gauche) + TAILLE(x.droit)
  sinon :
    renvoyer 0
  fin si
FIN
```



Cet algorithme ressemble beaucoup à l'algorithme étudié à l'exercice 1, son étude ne devrait donc pas vous poser de problème.

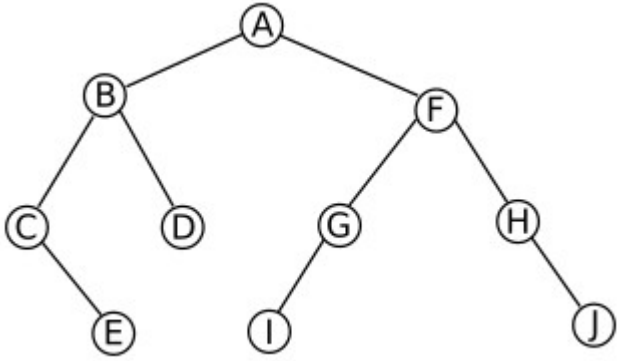
Appliquez cet algorithme à l'exemple ci-contre :

Exercice 3

Cet algo permet de calculer la taille d'un arbre

```
VARIABLE
T : arbre
x : noeud

DEBUT
PARCOURS-PREFIXE(T) :
  si T ≠ NIL :
    x ← T.racine
    affiche x.clé
    PARCOURS-PREFIXE(x.gauche)
    PARCOURS-PREFIXE(x.droit)
  fin si
FIN
```

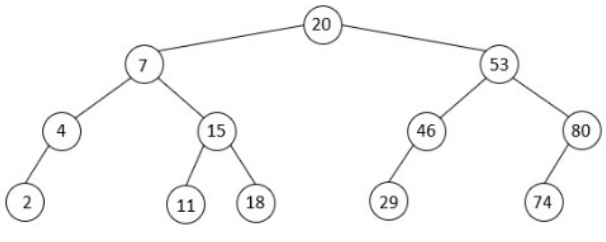


Vérifiez qu'en appliquant l'algorithme ci-dessus, l'arbre ci-dessous est bien parcouru dans l'ordre suivant : A, B, C, E, D, F, G, I, H, J

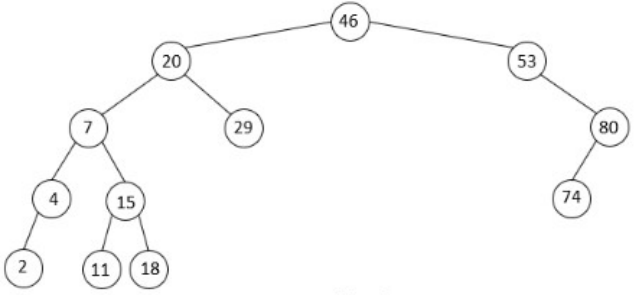
Exercice 4 *(sujet bac 2023)*

Dans cet exercice, la taille d'un arbre est égale au nombre de ses nœuds et on convient que la hauteur d'un arbre ne contenant qu'un nœud vaut 1.
On utilisera la définition suivante : un arbre binaire de recherche est un arbre binaire, dans lequel
• on peut comparer les valeurs des nœuds : ce sont par exemple des nombres entiers, ou des lettres de l'alphabet ;
• si x est un nœud de cet arbre et y est un nœud du sous-arbre gauche de x, alors il faut que
y.valeur < x.valeur
• si x est un nœud de cet arbre et y est un nœud du sous-arbre droit de x, alors il faut que
y.valeur ≥ x.valeur

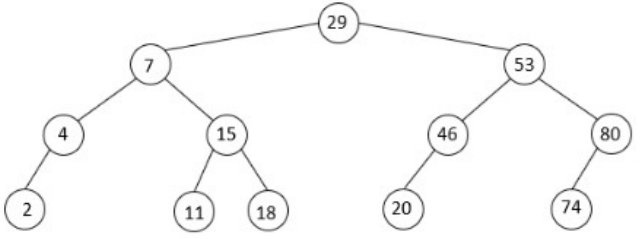
Parmi les trois arbres dessinés ci-dessous, recopier sur la copie le numéro correspondant à celui qui n'est pas un arbre binaire de recherche. Justifier.



Arbre 1



Arbre 2



Arbre 3