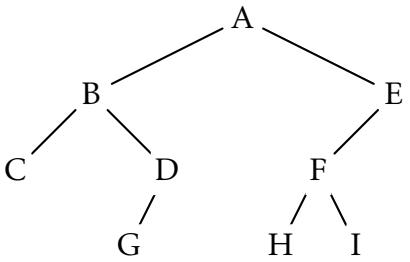
Arbres – Exercices de bac

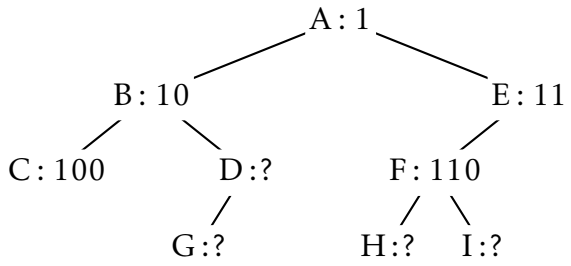
Exercice 1 : *Dans ce sujet, on utilisera la convention suivante : la hauteur d'un arbre binaire ne comportant qu'un nœud est 1.*



1) Déterminer la taille et la hauteur de l'arbre binaire ci-contre.

2) On décide de numéroter en binaire les nœuds d'un arbre binaire de la façon suivante :
   - la racine correspond à 1 ;
   - la numérotation pour un fils gauche s'obtient en ajoutant le chiffre 0 à droite au numéro de son père ;
   - la numérotation pour un fils droit s'obtient en ajoutant le chiffre 1 à droite au numéro de son père ;

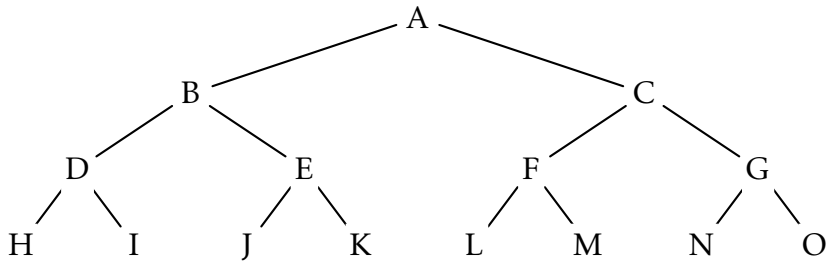
   Par exemple, dans l'arbre ci-contre, on a utilisé ce procédé pour numéroter les nœuds A, B, C, E et F.



   a) Dans l'exemple précédent, quel est le numéro en binaire associé au nœud G ?
   b) Quel est le nœud dont le numéro en binaire vaut 13 en décimal ?
   c) En notant h la hauteur de l'arbre, sur combien de bits seront numérotés les nœuds les plus en bas ?
   d) Justifier que pour tout arbre de hauteur h et de taille $n \geq 2$, on a :

$$h \leq n \leq 2^h - 1$$

3) Un arbre binaire est dit complet si tous les niveaux de l'arbre sont remplis.



Arbre binaire complet

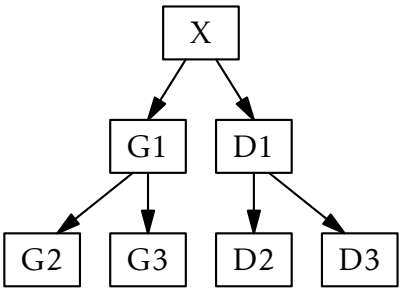
On décide de représenter un arbre binaire complet par un tableau de taille $n+1$, où n est la taille de l'arbre, de la façon suivante :
- La racine a pour indice 1 ;
- Le fils gauche du nœud d'indice i a pour indice $2 \times i$;
- Le fils droit du nœud d'indice i a pour indice $2 \times i + 1$;
- On place la taille n de l'arbre dans la case d'indice 0.

   a) Déterminer le tableau qui représente l'arbre binaire complet de l'exemple précédent.
   b) On considère le père du nœud d'indice i avec $i \geq 2$. Quel est son indice dans le tableau ?

4) On se place dans le cas particulier d'un arbre binaire de recherche complet où les nœuds contiennent des entiers et pour lequel la valeur de chaque noeud est supérieure à celles des noeuds de son fils gauche, et inférieure à celles des noeuds de son fils droit.

   Écrire une fonction `recherche` ayant pour paramètres un arbre `arbre` et un élément `element`. Cette fonction renvoie `True` si `element` est dans l'arbre et `False` sinon. L'arbre sera représenté par un tableau comme dans la question précédente.

Exercice 2 : Un arbre binaire est soit vide, soit un nœud qui a une valeur et au plus deux fils (le sous-arbre gauche et le sous-arbre droit). X est un nœud, sa valeur est X.valeur G1 est le fils gauche de X, noté X.fils_gauche D1 est le fils droit de X, noté X.fils_droit. Un arbre binaire de recherche est ordonné de la manière suivante :



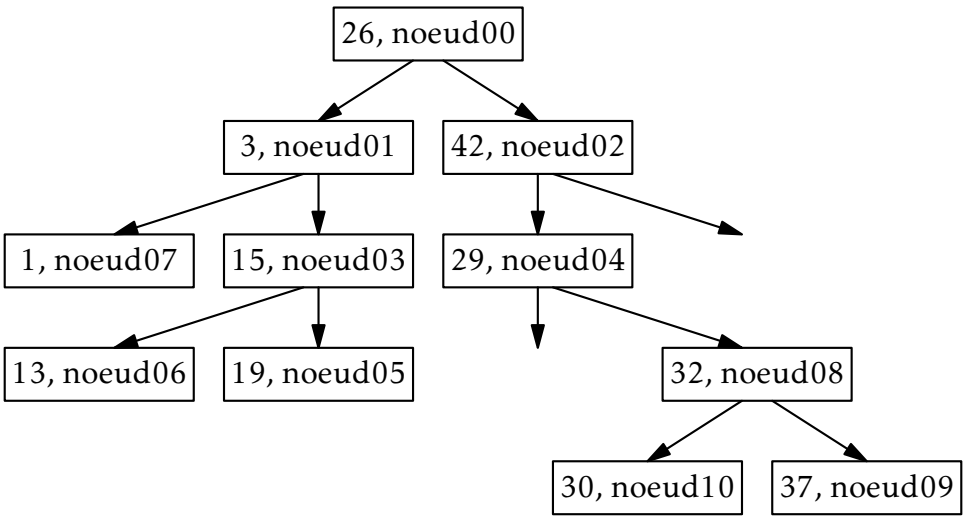
Pour chaque nœud X,

- les valeurs de tous les nœuds du sous-arbre gauche sont **strictement inférieures** à la valeur du nœud X
- les valeurs de tous les nœuds du sous-arbre droit sont **supérieures ou égales** à la valeur du nœud X

Ainsi, par exemple, toutes les valeurs des nœuds G1, G2 et G3 sont strictement inférieures à la valeur du nœud X et toutes les valeurs des nœuds D1, D2 et D3 sont supérieures ou égales à la valeur du nœud X.
Voici un exemple d'arbre binaire de recherche dans lequel on a stocké dans cet ordre les valeurs :

[26,3,42,15,29,19,13,1,32,37,30]

L'étiquette d'un nœud indique la valeur du nœud suivie du nom du nœud.
Les nœuds ont été nommés dans l'ordre de leur insertion dans l'arbre ci-dessous.
'29, noeud04' signifie que le nœud nommé noeud04 possède la valeur 29.



1) On insère la valeur 25 dans l'arbre, dans un nouveau nœud nommé noeud11.
   Recopier l'arbre binaire de recherche étudié et placer la valeur 25 sur cet arbre en coloriant en rouge le chemin parcouru.
   Préciser sous quel nœud la valeur 25 sera insérée et si elle est insérée en fils gauche ou en fils droit, et expliquer toutes les étapes de la décision.

2) Préciser toutes les valeurs entières que l'on peut stocker dans le nœud fils gauche du noeud04 (vide pour l'instant), en respectant les règles sur les arbres binaires de recherche ?

3) Voici un algorithme récursif permettant de parcourir et d'afficher les valeurs de l'arbre de recherche A.

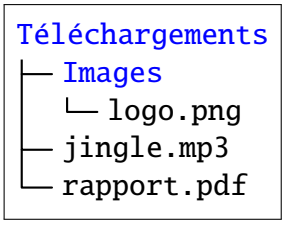
Algorithme Parcours(A) :
Afficher(A.valeur)
Parcours(A.fils_gauche)
Parcours(A.fils_droit)

   a) Écrire la liste de toutes les valeurs dans l'ordre où elles seront affichées.
   b) Choisir le type de parcours d'arbres binaires de recherche réalisé parmi les propositions suivantes : Préfixe, Suffixe ou Infixe.

4) En vous inspirant de l'algorithme précédent, écrire un algorithme Parcours2 permettant de parcourir et d'afficher les valeurs de l'arbre A dans l'ordre croissant.

Exercice 3 : *Cet exercice porte sur les structures de données (dictionnaires)*
Afin d'organiser les dossiers et les fichiers sur un disque dur, une structure arborescente est utilisée. Les fichiers sont dans des dossiers qui sont eux-mêmes dans d'autres dossiers, etc. Dans une arborescence, chaque dossier peut contenir des fichiers et des dossiers, qui sont identifiés par leur nom. Le contenu d'un dossier est modélisé par la structure de données **dictionnaire**. Les clés de ce dictionnaire sont des chaînes de caractères donnant le nom des fichiers et des dossiers contenus.

Le dossier appelé `Téléchargements` contient deux fichiers `rapport.pdf` et `jingle.mp3`, et un dossier `Images` contenant simplement le fichier `logo.png`. Il est représenté ci-contre.
Ce dossier `Téléchargements` est modélisé en Python par le dictionnaire suivant :




```
{"Images": {"logo.png": 36}, "rapport.pdf": 450, "jingle.mp3": 4800}
```

Les valeurs numériques sont exprimées en ko (kilo-octets). `"logo.png": 36` signifie que le fichier `logo.png` occupe un espace mémoire de 36 ko sur le disque dur.
On rappelle, ci-dessous, quelques commandes sur l'utilisation d'un dictionnaire :

- `dico = {}` crée un dictionnaire vide appelé `dico`,
- `dico[cle] = contenu` met la valeur contenu pour la clé `cle` dans le dictionnaire `dico`,
- `dico[cle]` renvoie la valeur associée à la clé `cle` dans le dictionnaire `dico`,
- `cle in dico` renvoie un booléen indiquant si la clé `cle` est présente dans le dictionnaire `dico`,
- `del dico[cle]` supprime la clé `cle` et sa valeur associée du `dico`,
- `dico.keys()` renvoie la liste des clés du dictionnaire `dico`.

L'**adresse** d'un fichier ou d'un dossier correspond au nom de tous les dossiers à parcourir depuis la racine afin d'accéder au fichier ou au dossier. Cette adresse est modélisée en Python par la liste des noms de dossier à parcourir pour y accéder.
Exemple : L'adresse du dossier : `/home/pierre/Documents/` est modélisée par la liste `["home", "pierre", "Documents"]`.

1) Dessiner l'arbre donné par le dictionnaire suivant, qui correspond au dossier `Documents` donné ci-dessous.

```
Documents = {
  "Administratif":{
    "certificat JDC.pdf ": 1500,
    "attestation recensement.pdf ": 850
  },
  "Cours": {
    "NSI": {
      "TP.html": 60,
      "dm.odt": 345
    },
    "Philo": {"Tractatus logico-philosophicus.epub": 2600}
  },
  "liste de courses.txt ": 24
}
```

2) a) On donne la fonction `Parcourir` suivante qui prend en paramètres un dossier racine et une liste représentant une adresse, et qui renvoie le contenu du dossier cible correspondant à l'adresse.
Exemple : Si la variable `Documents` contient le dictionnaire de l'exemple de la question 1 alors `Parcourir(Documents, ["Cours", "Philo"])` renvoie le dictionnaire `{"Tractatus logicophilosophicus.epub": 2600}`.
Compléter le code de la fonction :

```
def Parcourir(racine, adr):
    dossier = racine
    for nom_dossier in adr:
        dossier = . . . . . # À compléter
    return dossier
```

b) Soit la fonction suivante :

```
def Afficher(racine, adr, nom_fichier):
    dossier = Parcourir(racine, adr)
    print(dossier[nom_fichier])
```

Qu'affiche l'instruction `Afficher(Documents, ["Cours", "NSI"], "TP.html")` sachant que la variable `Documents` contient le dictionnaire de la question 1 ?

3) a) La fonction `Ajouter(racine, adr, nom_fichier, taille)` suivante ajoute au dictionnaire `racine`, à l'adresse `adr`, la clé `nom_fichier` associée à la valeur `taille`.
Une ligne de la fonction donnée ci-dessous contient une erreur. Laquelle ? Proposer une correction.

```
def Ajouter_fichier(racine, adr, nom_fichier, taille):
    dossier = Parcourir(racine, adr)
    taille = dossier[nom_fichier]
```

b) Écrire une fonction `Ajouter_dossier(racine, adr, nom_dossier)` pour créer un dictionnaire représentant un dossier vide appelé `nom_dossier` dans le dictionnaire racine à l'adresse `adr`.

4) Écrire une fonction `taille(dossier)` qui prend en paramètre un dictionnaire `dossier` modélisant le contenu du répertoire `dossier` et qui renvoie le total de l'espace mémoire occupé par les fichiers contenus dans le dossier. On considère que le répertoire `dossier` ne contient que des fichiers et **aucun sous-dossier**.