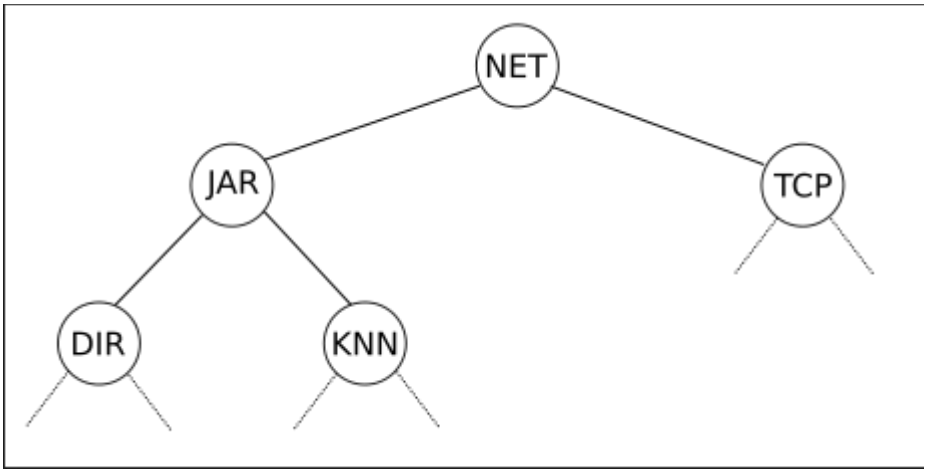
ACT2 ARBRES BINAIRES DE RECHERCHE

1 Bibliothèque

Imaginons une bibliothèque contenant un extrêmement grand nombre de livres. Cette bibliothèque est organisée de la manière suivante :

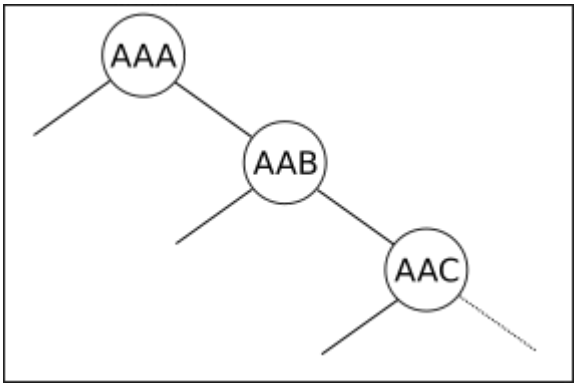
- Il y a 17 576 pièces différentes.
- Chaque pièce est repérée par une suite de trois lettres, et dans cette pièce sont rangés tous les livres dont les titres commencent par ces trois lettres.
- Chaque pièce possède deux sorties, une à droite et une à gauche.
- La sortie de gauche mène **toujours** soit à une salle dont les trois lettres sont situées avant dans l'ordre alphabétique, soit nulle part.
- La sortie de droite mène **toujours** soit à une salle dont les trois lettres sont situées après dans l'ordre alphabétique, soit nulle part.

Une représentation de cette bibliothèque peut être donnée sous la forme d'un arbre binaire tel que le suivant :



1. Où sont situés les livres dont le titre commence par `KNU ? UDP ? JET ?`
2. Pourquoi y a-t-il 17 576 pièces différentes ?

Un autre répartition possible serait :

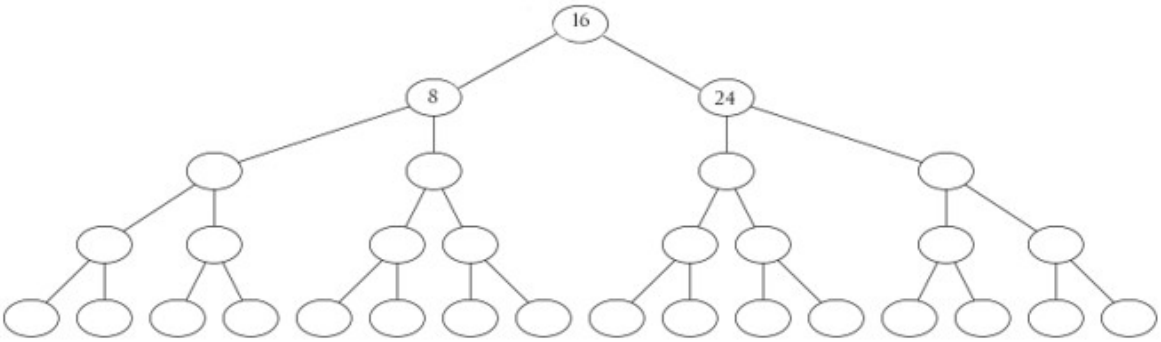


3. Pourquoi cette répartition n'est pas optimale par rapport à la précédente ?

2 Choix d’un nombre le plus rapidement possible...

Pearl choisit un nombre entier compris entre 1 et 31. Bob cherche à trouver le plus rapidement possible ce nombre en posant des questions à Pearl.

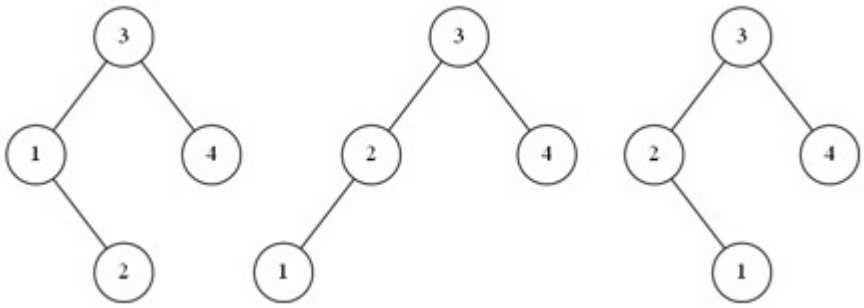
1. Quelle stratégie vue en première permettra à Bob de trouver le plus rapidement possible le nombre choisi par Pearl ?
2. On peut représenter cette stratégie à l’aide d’un arbre binaire. Compléter cet arbre :



3. Combien d’étapes sont nécessaires au maximum pour trouver le nombre qu’a choisi Pearl ?

4. En vous inspirant de l’exemple précédent, construire un arbre binaire permettant de trouver rapidement une lettre dans l’alphabet.

Exemples : Parmi les trois arbres suivants, lesquels sont des arbres binaires de recherche ?



Dans quel ordre sont visités les nœuds de ces arbres en parcours infixe ? Que remarque-t-on ?

3 Application : Clinique vétérinaire

Un vétérinaire voudrait stocker les fiches médicales de ses patients, et, plutôt que d'utiliser un tableau ou une liste, on se propose d'utiliser un ***arbre binaire***.

La fiche contiendra différentes informations sur l'animal ; on utilisera son nom comme clé, que l'on triera selon l'ordre alphabétique croissant.

Le vétérinaire reçoit sa première patiente, qui répond au nom de Gaufrette. Comme sa fiche sera le premier nœud de notre arbre, elle en devient automatiquement la racine. Puis le vétérinaire reçoit les animaux dans l’ordre suivant afin de les soigner : Charlie, Médor, Flipper, Bubulle et Augustin.

Construire l’arbre binaire de recherche associé à cette séquence.

4 Recherche dans un ABR

A retenir :

Pour chercher une valeur dans un ABR, on la compare à la valeur de la racine. Si elle est égale, on a trouvé, sinon on se dirige vers le sous-arbre gauche (si la valeur cherchée est plus petite), ou le sous-arbre droit (si la valeur est plus grande), et on recommence de manière récursive.

Cette recherche se décrit ainsi de manière récursive :

- Si l'arbre est vide, la valeur n'est pas dans l'arbre !
- Sinon, l'arbre contient au moins un nœud. On compare donc la valeur cherchée avec celle de la racine de l'arbre :
  - Si la valeur est plus petite, on continue la recherche dans le sous-arbre gauche (de manière récursive)
  - Si la valeur est plus grande, on continue la recherche dans le sous-arbre droit
  - Sinon on a trouvé la valeur.

Ecrire l'algorithme de la méthode **recherche(self, arbre)** qui renvoie **True** si la valeur est dans l'arbre, **False** sinon.

L'implémenter dans une classe ABR, qui reprend les fonctionnalités de la classe AB.

5 Ajout d'un élément dans un ABR

Ajouter un élément dans un ABR repose que le même principe que la recherche d'un élément :

- Si le nouvel élément est plus petit que la valeur du nœud en cours, on va à gauche
- Si le nouvel élément est plus grand que la valeur du nœud en cours, on va à droite
- Quand on arrive à un arbre vide, on ajoute un nouveau nœud.

Ecrire l'algorithme correspondant à l'ajout d'un élément dans un ABR.

L'implémenter dans la classe ABR.