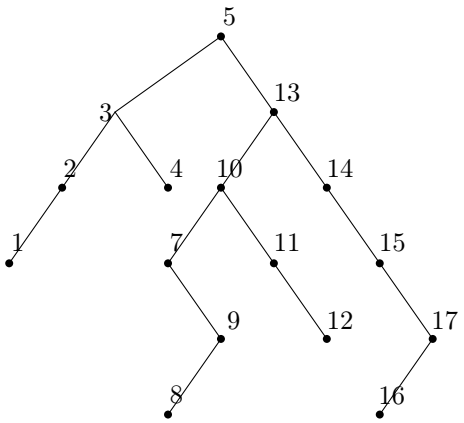
1 ABR

Question 1
Écrire une version itérative de la recherche d'un élément dans un ABR.
Justifiez que votre algorithme est dans $\theta(h)$ (où h est la hauteur de l'ABR).



Question 2 On veut supprimer l'étiquette 13 dans l'ABR ci dessus.

- indiquez les différentes étapes de l'**algorithme vu en cours**
- indiquer comment modifier le dessin pour représenter le résultat obtenu par l'algorithme

Question 3 On suppose que des entiers compris entre 1 et 1000 sont disposés dans un arbre binaire de recherche et que l'on souhaite trouver le nombre 363. Parmi les séquences suivantes, lesquelles **ne** pourraient **pas** être la suite des nœuds parcourus ? Justifiez chacune de vos réponses (ne pas justifier les séquences qui peuvent être la suite des nœuds parcourus).

1. 2, 252, 401, 398, 330, 344, 397, 363.
2. 924, 220, 911, 244, 898, 258, 362, 363.
3. 925, 202, 911, 240, 912, 245, 363.
4. 2, 399, 387, 219, 266, 382, 381, 278, 363.
5. 935, 278, 347, 621, 299, 392, 358, 363.

2 Codage d'une arborescence quelconque par une arborescence binaire.

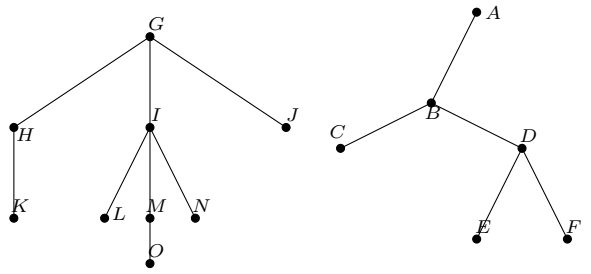
2.1 Le codage

On définit ci dessous une bijection $\mathcal{C}_d$ qui à toute arborescence (quelconque) A_Q fait correspondre une arborescence **binaire** $A_B = \mathcal{C}_d(A_Q)$.
On note $\mathcal{C}_d(S_Q)$ le sommet de A_B qui correspond au sommet S_Q de A_Q.

Définition de $\mathcal{C}_d$

1. l'image (par $\mathcal{C}_d$) de la racine de A_Q est la racine de A_B
2. le fils ainé de tout sommet S_Q de l'arborescence à coder devient (par $\mathcal{C}_d$) le fils gauche de $\mathcal{C}_d(S_Q)$
3. le frère cadet immédiatement à droite de tout sommet S_Q de l'arborescence à coder devient (par $\mathcal{C}_d$) le fils droit de $\mathcal{C}_d(S_Q)$

`Question 1` : coder l'arborescence de gauche et décoder l'arborescence de droite



2.2 Complexité en volume

`Question 2 :`
Comparer la classe Θ de la taille d'une arborescence quelconque avec celle de son codage

2.3 Traversées

`Question 3 :`
Pour quelle(s) traversée(s) le résultat de la traversée d'une arborescence quelconque et le résultat de la traversée du codage de cette arborescence sont-ils les mêmes ? (aucune justification n'est demandée)

3 Notation O et Θ

1. Pour cette question, on suppose vérifiée la loi empirique qui dit que la puissance d'un ordinateur double tous les 10 ans.
   Dans 100 ans, par combien aura été multipliée la puissance d'un ordinateur (à la louche : un "1" suivi du nombre suffisant de "0")
   Et dans 200 ans ?

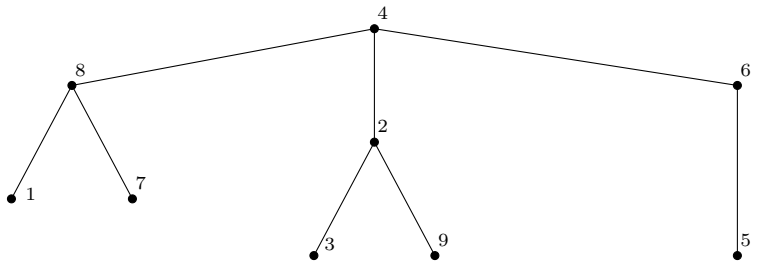
2. pour ces questions, on suppose **de plus** que la taille maximale d'une donnée fournie à un algorithme double aussi tous les 10 ans.
   Dans combien de temps aura été multipliée par 1000 la durée d'exécution sur une donnée de taille maximale (qui change donc au cours du temps) d'un algorithme dont la complexité, en fonction de la taille n de la donnée, appartient à la classe
   (a) $O(logn)$
   (b) $O(\sqrt{n})$
   (c) $O(n)$
   (d) $O(n^2)$
   (e) $O(n^3)$
   (f) $O(2^n)$
   (g) $O(nlog(n))$

   La réponse peut être
   — jamais
   — dans environ ... années
   — on ne peut pas le calculer
   mais on demande à chaque fois une courte justification.

   Indication : en 10 ans, comment aura évolué le temps de calcul d'une donnée de taille maximale ?

4 Un problème sur les arborescences (quelconques)

On veut représenter une arborescence de n sommets numérotés de 1 à n par un tableau `P` de n éléments indicé de 1 à n.
La case i du tableau `P` représentera le sommet i et contiendra l'indice du père de ce sommet (la case représentant la racine contiendra la valeur 0).
Par exemple l'arborescence



sera représentée par le tableau `P`

8	4	2	0	6	4	8	4	2

le sommet 1 a pour père 8, le sommet 2 a pour père 4 ...

Le problème

On veut un algorithme qui étant donné un tableau `P` de taille n donnée, calcule le nombre de feuilles de l'arborescence que représente `P` .

Question 4

sans utiliser aucune autre structure de données donner un algorithme en $\theta(n^2)$. Justifier sa complexité.

Question 5

Donner un algorithme en $\theta(n)$ qui resolve le problème.

On utilisera la structure de graphe vue en cours, et chaque sommet sera repéré par son indice dans le tableau des sommets, qui sera le même que son indice dans le tableau `P`.

Pour manipuler la structure de graphe vue en cours, utiliser les primitives (chacune en $\theta(1)$)

- *CreerGrapheVide(int n)* : fabrique et renvoie un graphe de n sommets isolés
- *AjouterSuccesseur*(`G`, `SD`, `SA`) :
  modifie le graphe `G` passé en paramètre en ajoutant un arc de `SD` à `SA`.
- *FeuilleP* (`S`,`A`) indique si le sommet S est une feuille dans le graphe A.

Préciser les entrées et les sorties de chacun de vos sous algorithmes et justifiez leur complexité.

5 Algorithmes de Prufer

5.1 Codage

Données:
$n \in \mathbb{N}^*$, on appellera $\mathcal{I}$ l'intervalle des entiers entre 1 et n
$\mathcal{A} = (X,\ E)$ un arbre (quelconque) de n sommets dont chaque sommet porte un *numéro* de $\mathcal{I}$

Résultat: $\mathcal{S}$ une suite de $n-2$ entiers, tous dans $\mathcal{I}$, avec éventuellement des répétitions

$\mathcal{S} \leftarrow$ la suite vide;
tant que *il reste plus de deux sommets dans* $\mathcal{A}$ **faire**
  Choisir la feuille de numéro maximal de $\mathcal{A}$;
  Rajouter à droite de $\mathcal{S}$ le numéro du voisin de cette feuille;
  Retirer de $\mathcal{A}$ cette feuille (et l'arête qui la relie à son voisin)
fin
retourner $\mathcal{S}$

5.1.1 Problème

Proposer (et justifier) une classe de complexité (la meilleure possible)
- dans le cas où vous ne disposez comme structure de données
  - pour $\mathcal{A}$ que d'un tableau (indicé par $\mathcal{I}$) de listes de successeurs
  - pour $\mathcal{S}$ que d'un tableau de n entiers (et d'un indice)
  - et comme **seule** structure suplémentaire d'un tableau de n booléens

- dans le cas où vous avez droit aux structures de votre choix (qu'il vous appartiendra alors de décrire avec leurs utilisations).

5.2 Décodage

Données:
$n \in \mathbb{N}^*$, on appellera $\mathcal{I}$ l'intervalle des entiers entre 1 et n
$\mathcal{S}$ une suite de $n-2$ entiers, tous dans $\mathcal{I}$, avec éventuellement des répétitions

Résultat: $\mathcal{A} = (X,\ E)$ un arbre dont chaque sommet porte un numéro de $\mathcal{I}$,

Fabriquer les n sommets isolés de $\mathcal{A}$, numérotés de 1 à n;
tant que *il reste des éléments dans* $\mathcal{S}$ *et plus de deux éléments dans* $\mathcal{I}$
faire
  Trouver le plus grand élément i de $\mathcal{I}$ n'apparaissant pas dans $\mathcal{S}$;
  Rajouter dans $\mathcal{A}$ une arête entre
    — le sommet de numéro i et
    — le sommet dont le numéro s est le premier élément de $\mathcal{S}$;
  Retirer i de $\mathcal{I}$ et s de $\mathcal{S}$
fin
retourner $\mathcal{A}$

Même problème.

6 Problème

Question 1

Dans l'arbre binaire de recherche que nous considérons dans ce problème, toutes les valeurs sont différentes. Pour un sommet S, on notera $Val(S)$ la valeur de ce sommet.

1. à quelle condition un sommet S est-il le sommet ayant la plus petite valeur ? Si S n'est pas le sommet qui a la plus petite valeur, et si toutes les valeurs sont différentes, on appelle $Pred(S)$ le sommet qui a la valeur imédiatement inférieure à celle de S
2. Dans le premier cas que nous examinons, S a une sous arborescence gauche
   (a) où se trouve alors $Pred(S)$? appelez ce sommet $Min(S)$ dans les explications suivantes par lesquelles vous allez démontrer votre intuition.
   (b) expliquez pourquoi pour tout sommet T
      i. qui se trouve dans la sous arborescence gauche de S, $T \neq Min(S) \Rightarrow T$ ne peut pas être le prédécesseur de S
      ii. qui se trouve dans la sous arborescence droite de S, T ne peut pas être le prédécesseur de S
      iii. qui se trouve sur le chemin de S à la racine, T ne peut pas être le prédécesseur de S (il faudra examiner deux cas, suivant que S est dans la sous arborescence droite ou gauche de T)
      iv. qui se trouve ailleurs dans l'arbre, T ne peut pas être le prédécesseur de S (il faudra considérer U l'ancêtre commun à S et T le plus bas dans l'arbre, puis examiner deux cas)
3. Dans le deuxième cas que nous examinons, S n'a pas de sous arborescence gauche :
   (a) indiquez alors où vous pensez que se trouve $Pred(S)$ et appelez le $Bas(S)$ dans les explications ultèrieures.
   (b) prouvez le de la même manière que pour le premier cas

Question 2

Dans un arbre binaire de recherche dont toutes les valeurs sont différentes, pour lequel on dispose des fonctions (en $\Theta(1)$)

- $racine?(S)$ qui indique si un sommet S est la racine de l'arbre
- $pere(S)$, définie pour les autres sommets que la racine et qui renvoie le père du sommet S
- $val(S)$ qui renvoie l'étiquette du sommet S
- $sad(S)$ qui renvoie la racine de la sous arborescence droite du sommet S
- $sag(S)$ qui renvoie la racine de la sous arborescence gauche du sommet S
- $fg?(S)$, définie pour les autres sommets que la racine et qui indique si un sommet S est le fils gauche de son père
- $fd?(S)$, définie pour les autres sommets que la racine et qui indique si un sommet S est le fils droit de son père
- $nonVide?(A)$ qui indique que l'arborescence A n'est pas vide

1. écrivez la fonction $Pred(S)$ (une solution avec des fonctions auxiliaires est préférable).
2. Donnez et justifiez la classe de complexité de votre fonction en fonction de la hauteur h de l'arbre.