LU2IN003
Arbres binaires (suite) semaine 7

Objectif(s)

✯ Les exercices à faire en priorité sont ceux de la feuille nommée TD7-exos-simples

Exercice(s)

1 Arbres H-équilibrés

Exercice 1 – Arbres binaires et nombres de Fibonacci

On rappelle la définition des nombres de Fibonacci : $F_0 = 0$, $F_1 = 1$ et $F_k = F_{k-1} + F_{k-2}$ si $k \geq 2$.
On donne la définition inductive suivante des *arbres de Fibonacci* :

— $T_0 = \emptyset$, $T_1 = \bullet$
— $T_k = (\bullet, T_{k-1}, T_{k-2})$ si $k \geq 2$

Question 1
Dessiner T_2, T_3, T_4, T_5.

Question 2
Montrer que $n(T_k) = F_{k+2} - 1$, pour tout $k \in \mathbb{N}$ par induction structurelle.

Question 3
Montrez par induction structurelle que, pour tout $k \geq 0$, $h(T_k) = k$ et que T_k est H-équilibré.

Exercice 2 – Hauteur d'un arbre H-équilibré

On note u_h la taille minimale d'un arbre H-équilibré de hauteur h.

Question 1
Calculer u_0, u_1, u_2, u_3, u_4.

Question 2
Montrer que la suite u_h est croissante et que $u_h = u_{h-1} + u_{h-2} + 1$ si $h \geq 2$. En déduire que $u_h = F_{h+2} - 1$, pour tout $h \in \mathbb{N}$.

Question 3

Montrer par récurrence que $F_{h+2} \geq \Phi^h$ avec $\Phi = \dfrac{1+\sqrt{5}}{2}$ (sachant que $\Phi \sim 1,61803$ est tel que $\Phi^2 - \Phi - 1 = 0$).
En déduire que la hauteur h d'un arbre H-équilibré est en $O(\log n)$.

2 Arbres parfaits et tas

Exercice 3 – Hauteur et taille d'un arbre parfait

Question 1
Montrer que la taille n d'un arbre parfait de hauteur h vérifie $2^{h-1} \leq n < 2^h$.

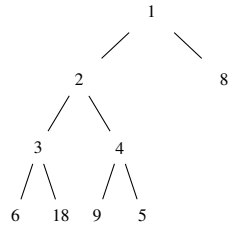
Question 2
En déduire la valeur de h en fonction de n. Quelle est la hauteur d'un arbre parfait de 10000 sommets ?

Exercice 4 – Opérations sur les tas

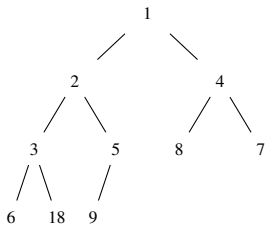
Dans cet exercice on considère des arbres binaires étiquetés par des nombres.
Un *tournoi* est un arbre binaire dont les étiquettes croissent de la racine vers les feuilles. Un *tas* est un tournoi parfait.
Le *parcours par niveau* d'un arbre parfait est la liste obtenue en parcourant les nœuds de l'arbre niveau par niveau et de gauche à droite.

Question 1
Pour chacun des trois arbres binaires T_1, T_2, T_3 suivants, dire s'il est parfait, s'il est un tas. Justifier les réponses.

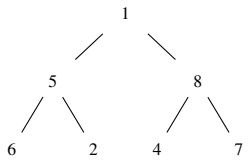
Arbre T_1 :



Arbre T_2 :

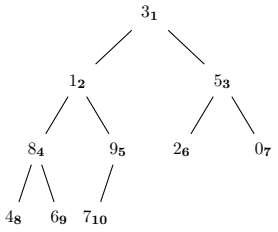


Arbre T_3 :

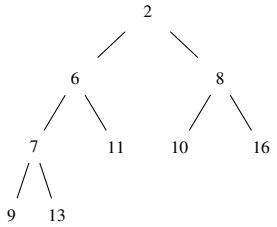


On rappelle qu'un arbre parfait de taille n peut être représenté au moyen d'un tableau $A[0..N]$, avec $N \geq n$, tel que :
- $A[0]$ contient la taille n de T
- les cases $A[1..n]$ sont remplies en parcourant T de gauche à droite, niveau par niveau.

Ce tableau A est appelé *parcours par niveau* de l'arbre parfait T.
On peut numéroter les nœuds de l'arbre parfait T au moyen de leur position dans le tableau A comme illustré dans l'exemple suivant :



Le tableau associé à est $[10, 3, 1, 5, 8, 4, 9, 2, 0, 4, 6, 7]$



Question 2
On considère le tas ExT :

1. Donner le tableau ExA associé à ExT.
2. Réaliser l'insertion de la clé 5 dans le tas ExT, en maintenant la structure de tas. Décrire brièvement (deux phrases maximum) l'algorithme utilisé. Donner le tableau associé au résultat.
3. Réaliser la suppression de la clé minimale dans le tas ExT, en maintenant la structure de tas. Décrire brièvement (deux phrases maximum) l'algorithme utilisé. Donner le tableau associé au résultat.

Question 3
Soit L une liste de n entiers naturels.

1. Décrire brièvement (deux phrases maximum) l'algorithme permettant de trier la liste L en utilisant un tas.
2. Soit $L_0 = (5, 2, 3, 1, 4)$.
   (a) Construire le tas T_0 obtenu en insérant successivement les éléments de L_0 et le tableau A_0 associé à ce tas.
   (b) Détruire le tas T_0 par suppressions successives du minimum. On dessinera l'arbre obtenu à chaque suppression et on donnera le tableau associé à chaque tas.

Exercice 5 – Tri par tas : implémentation, complexité

Dans cet exercice on considère des arbres binaires étiquetés par des nombres.

Question 1
Soit A un tableau représentant un arbre parfait T. Donner une condition nécessaire et suffisante pour que T soit un tas.
Dans la suite de l'exercice un tableau représentant un tas sera appelé tas et on lui appliquera la terminologie des arbres binaires (feuille, fils, père, etc).

Question 2

1. Écrire le prédicat `estFeuille(A,i)` qui retourne vrai ssi $A[i]$ est une feuille du tas A.
2. Écrire le prédicat `aUnFilsDroit(A,i)` qui retourne vrai ssi $A[i]$ a un fils droit.
3. Écrire la fonction `Fg(A,i)` qui retourne l'indice du fils gauche de $A[i]$, en supposant qu'il existe.
4. Écrire la fonction `Fd(A,i)` qui retourne l'indice du fils droit de $A[i]$, en supposant qu'il existe.
5. Écrire la fonction `pere(A,i)` qui retourne l'indice du père de $A[i]$, en supposant qu'il existe.

Question 3
Écrire la procédure `inserer(x,A)` qui insère un nombre x dans un tas A. Calculer la complexité de cette opération.

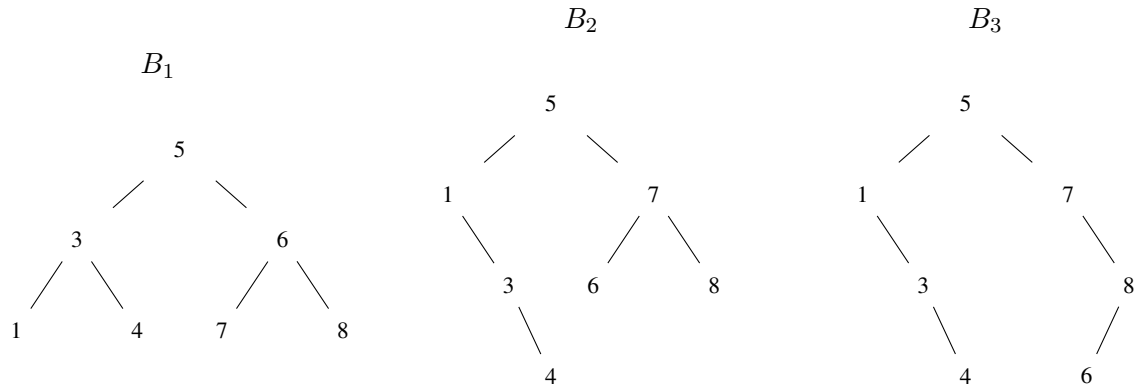
Question 4
Écrire la procédure `supprimerMin(A)` qui supprime un élément minimum du tas A. On placera cet élément minimum à la place de la feuille supprimée. Calculer la complexité de cette opération.

Question 5
Écrire la procédure `trier(A)` qui range en ordre décroissant $A[1..n]$. Cette procédure utilisera un tas. Calculer la complexité de cette opération.

3 Arbres binaires de recherche (ABR)

Exercice 6 – Définition des ABR

Question 1
Est-ce que les arbres B_1, B_2 et B_3 sont des ABR ? Justifier les réponses négatives.



Question 2
Soit E un ensemble ordonné. Rappelez la définition (non inductive) d'un ABR sur E, et la définition non inductive d'un ABR sur E. Démontrez que ces deux définitions sont équivalentes.
Pour cela, on note ABR(E) l'ensemble des ABR sur E par la définition non inductive, et $\mathcal{ABR}(E)$ l'ensemble des ABR sur E par la définition inductive, et on montre que ABR$(E) = \mathcal{ABR}(E)$.

Exercice 7 – Caractérisation d'un ABR

On rappelle que, si T est un ABR, `ABinfixe(T)` est une liste rangée en ordre strictement croissant.

Question 1

1. Dessiner trois ABR différents dont les clefs sont $3, 1, 4, 8, 5, 2, 7, 6$. Donner le parcours infixe de chacun d'eux.
2. Pouvez-vous dessiner deux ABR différents ayant la même forme et les mêmes clefs (deux à deux distinctes) ?

Question 2
Montrer que, si le parcours infixe des clefs d'un arbre binaire T est rangé en ordre strictement croissant alors T est un ABR. Cette propriété peut être démontrée par récurrence sur la taille de l'arbre.

Question 3

1. Décrire en une phrase le principe d'un algorithme qui teste si un arbre binaire est un ABR.
2. En supposant que les listes sont représentées par des listes circulaires doublement chaînées, calculer la complexité de cet algorithme.

Question 4
Soit P une liste dont les éléments sont deux à deux distincts. Montrer que, s'il existe un ABR dont le parcours préfixe est P, alors cet arbre est unique.

Exercice 8 – Liste des clés inférieures à une valeur

Question 1
On dispose de la fonction `ABinfixe(T)` qui retourne le parcours infixe de l'arbre binaire T. On rappelle que, si T est un ABR, `ABinfixe(T)` est une liste rangée en ordre strictement croissant.
On définit la fonction :

```
def listePlusPetits(T, x):
  if estABvide(T):
    return []
  if x <= T.clef:
    return listePlusPetits(T.gauche, x)
  return ABinfixe(T.gauche) + [T.clef] + listePlusPetits(T.droit, x)
```

Exécutez la fonction `ABinfixe(T)` sur l'arbre B_2 de l'exercice 5. Vous préciserez l'ordre des appels et les listes renvoyées pour chaque appel.

Question 2
Montrer que `listePlusPetits(T, x)` se termine et retourne la liste des clés de T qui sont inférieures à x, rangée en ordre croissant.

Exercice 9 – Exemple de manipulation d'un ABR

Question 1
L'ajout de nouvelles clefs dans un ABR se fait par insertion aux feuilles. Construire l'ABR Tex obtenu par insertion successive des clefs 32, 7, 23,64, 18, 28, 41 et 58. Est-ce que cet ABR est unique ? Que se passe-t'il si l'on change l'ordre des clefs ?

Question 2
Donner l'ABR obtenu en supprimant le maximum de Tex.

Question 3
Donner l'ABR obtenu en supprimant la racine de Tex.

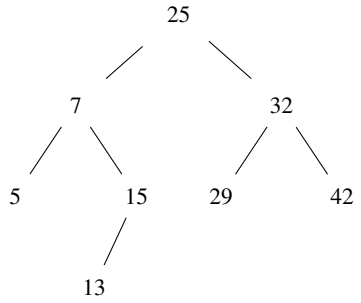
Question 4
Donner l'ABR obtenu en supprimant la clef 7 dans Tex.

Exercice 10 – Insertion d'une clef dans un ABR

Dans cet exercice, on étudie l'algorithme d'insertion d'une clef dans un ABR rappelé à la suite :

```
def ABRinsertion(x,T):
  if estABvide(T):
    return ABfeuille(x)
  if x == T.clef:
    return T
  if x < T.clef:
    return AB(T.clef,ABRinsertion(x,T.gauche),T.droit)
  return AB(T.clef,T.gauche,ABRinsertion(x,T.droit))
```

Question 1
On considère dans cette question l'ABR suivant. Donner son parcours infixe. Conclusion ? Construire l'ABR obtenu après l'insertion de la clef 31. Donnez alors le parcours infixe de ce nouvel ABR.



Question 2
Montrer que `ABRinsertion(x,T)` se termine et retourne un ABR dont les clés sont x et les clés de T.

Exercice 11 – Recherche et suppression du maximum dans un ABR

Dans cet exercice, on étudie les algorithmes de recherche du maximum et de suppression du maximum dans un ABR. Ces deux algorithmes sont rappelés ci-dessous :

— Recherche du maximum

```
def ABRmax(T):
  if estABvide(T.droit):
    return T.clef
  return ABRmax(T.droit)
```

— Suppression du maximum

```
def ABRsaufMax(T):
  if estABvide(T.droit):
    return T.gauche
  return AB(T.clef,T.gauche,ABRsaufMax(T.droit))
```

Question 1
Montrer que `ABRmax(T)` se termine et retourne la plus grande clé de T.

Question 2
Montrer que `ABRsaufMax(T)` se termine et retourne l'arbre obtenu en supprimant la plus grande clé de T.

Exercice 12 – Suppression d'une clef dans un ABR

Question 1
Quel est le principe d'un algorithme qui supprime la racine ? Écrire une définition de la fonction qui renvoie l'ABR obtenu en supprimant la racine d'un ABR.

Question 2
Quel est le principe d'une fonction qui supprime un élément de clef x fixée ? Écrire une définition récursive de la fonction qui renvoie l'ABR obtenu en supprimant une clef x dans un ABR. Qeulle est sa complexité ?