Arbres binaires (suite) semaine 8

Exercice(s)

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 $k \in \mathbb{N}$.

Question 3
Montrer que les arbres de Fibonacci sont H-équilibrés.

Exercice 2 – Arbres H-équilibrés

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 $O(\log n)$.

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 – Tri par 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

1. On considère les listes $L_1 = (1, 4, 3, 6, 9, 7, 12, 8, 7, 9, 10, 10)$ et $L_2 = (5, 6, 12, 4, 3, 8, 10, 7, 15, 2)$. Dessiner les arbres parfaits T_1 et T_2 dont L_1 et L_2 sont les parcours par niveau. T_1 est-il un tas ? et T_2 ? Quels échanges peut-on effectuer pour se ramener à un tas le cas échéant ?
2. Une liste triée est-elle nécessairement le parcours par niveau d'un tas ?
3. Où peut-on trouver un élément minimum dans un tas ?
4. Où peut-on trouver un élément maximum dans un tas ?
5. Donner la hauteur h d'un tas de n éléments.

Question 2

1. Décrire un algorithme d'insertion d'un élément dans un tas.
2. Décrire un algorithme de suppression du minimum dans un tas.

On rappelle qu'un arbre parfait de taille n peut être représenté par 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.

Question 3

Soit A un tableau représentant un arbre parfait. Donner une condition nécessaire et suffisante pour que A 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 4

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 5

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

Question 6

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.

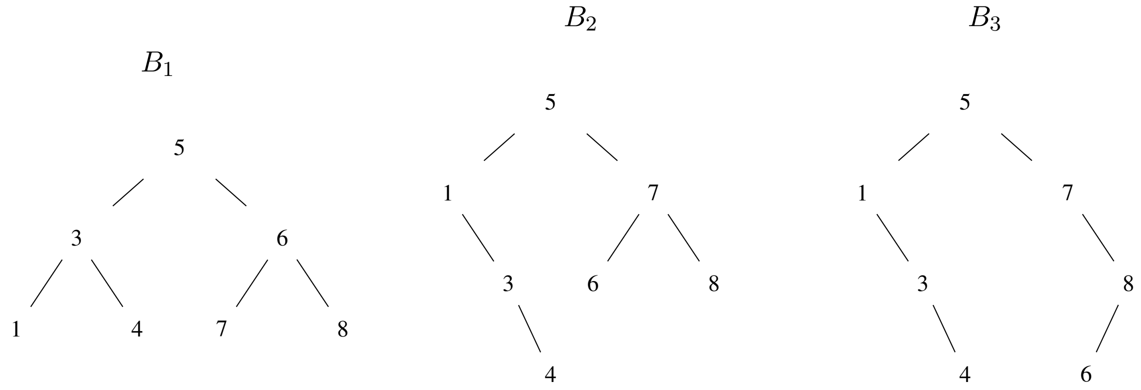
Question 7

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

Exercice 5 – 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 6 – Parcours d'un ABR

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 ? Que faut il faire pour le démontrer ?

Question 2

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 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 3

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

Question 4

Montrer que, si le parcours infixe des clefs d'un arbre binaire T est rangé en ordre strictement croissant alors T est un

ABR. On observe qu'il s'agit ici de la réciproque de la question précédente. Cette propriété peut être démontrée par récurrence sur la taille de l'arbre.

Question 5

1. Écrire une définition de la fonction qui calcule la liste résultant du parcours infixe d'un arbre binaire.
2. Écrire une définition de la fonction qui teste si une liste est rangée en ordre strictement croissant.
3. Utiliser ces deux fonctions pour écrire une définition de la fonction qui teste si un arbre binaire est un ABR. Quelle est la complexité ?

Question 6

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 7 – Construction d'un ABR et recherche de l'élément minimum

Question 1

On rappelle l'algorithme d'insertion dans un ABR :

```
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))
```

(On dispose d'une fonction `ABfeuille(x)` qui retourne un ABR réduit à une feuille de clé x.)
Construire l'ABR obtenu par insertion succéssive des clefs 32, 7, 23,64, 18, 28, 41 et 58. Est-ce que cet ABR est unique ? Que se passe t'il si on change l'ordre des clefs ?
Montrer que `ABRinsertion(x,T)` se termine et retourne un ABR dont les clés sont x et les clés de T.

Question 2

Si T est un ABR non vide, ou se trouve les plus petit élément ?

Question 3

On considère l'algorithme de recherche du minimum dans un ABR non vide :

```
def ABRmin(T):
  if estABvide(T.gauche):
    return T.clef
  return ABRmin(T.gauche)
```

Montrer que `ABRmin(T)` se termine et retourne la plus petite clé de T.

Exercice 8 – Suppression d'une clef dans un ABR

Question 1

Construire un ABR par insertion successive des clefs suivantes : 25, 50, 7, 35, 15, 12, 5, 42.

Question 2

Écrire une définition récursive de la fonction qui renvoie le maximum d'un ABR.

Question 3

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

Question 4

Supprimez la racine de l'ABR de la 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 5

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. Qeulles est sa complexité ?