Numéro d'anonymat :

Examen 2I003

Jeudi 21 Juin 2018, 2 heures
aucun document autorisé

Exercice 1 – Problème de décomposition - 12 points

Préliminaires

On dit qu'une suite d'entiers $(M_0, \ldots, M_{n-1})$ est *supercroissante* si tous ses éléments sont strictement positifs et si chaque élément est strictement supérieur à la somme de ses précédents : $M_i > M_0 + \ldots + M_{i-1}$ pour tout $i = 1, \ldots, n-1$.

Étant donné un entier naturel s et une suite $M = (M_0, \ldots, M_{n-1})$ supercroissante, on cherche à savoir s'il existe n valeurs binaires $x_0, \ldots, x_{n-1}$ ($x_i = 0$ ou 1) telles que $s = x_0 M_0 + \ldots + x_{n-1} M_{n-1}$.
Dans cet exercice on appellera ce problème : *problème de décomposition*.

Question 1
Pour chacune des suites ci-dessous, dire si elle est supercroissante, en justifiant la réponse :
a) $(1, 3, 4, 7, 10)$; b) $(2, 3, 8, 14, 31)$; c) $(1, 2, 4, 8)$.

Question 2
On considère la suite supercroissante $M = (1, 3, 5, 10, 20)$. Pour chacun des entiers suivants, dire si le problème de décomposition admet une solution :
$s_1 = 0$; $s_2 = 26$; $s_3 = 17$.
S'il existe une solution, donner les valeurs des x_i.

On considère la fonction :

```
def plusGrandInd(M,s):
    n = len(M)
    k = 1
    while (k < n) and M[k] <= s:
        k = k + 1
    return k - 1
```

où M est un tableau de nombres rangés en ordre croissant et s un nombre supérieur ou égal à $M[0]$.
On rappelle que `len(M)` est le nombre d'éléments du tableau M.

Question 3
On note k_i la valeur de k à la fin de l'itération i. Initialement, c'est-à-dire à la fin de l'itération 0, $k_0 = 1$.

1. Montrer, par récurrence sur i, qu'à la fin de l'itération i, on a $M[j] \leq s$ pour tout j tel que $0 \leq j < k_i$.

2. En déduire que `plusGrandInd(M,s)` retourne le plus grand indice p tel que $M[p] \leq s$.

Question 4
Calculer la complexité de `plusGrandInd` dans le meilleur cas et dans le pire cas, en précisant quel est le meilleur cas et quel est le pire cas pour un tableau de taille n.

Question 5
Soit $M = (M_0, \ldots, M_{n-1})$ une suite supercroissante.

1. Quelle est la plus petite valeur non nulle de s pour laquelle le problème de décomposition admet une solution ?
2. Quelle est la plus grande valeur de s pour laquelle le problème de décomposition admet une solution ?

On considère la fonction :

```
def existeDec(M, s):
    print ("Valeur␣de␣s␣:␣", s)
    if s == 0:
        return True
    if s < M[0]:
        return False
    p = plusGrandInd(M, s)
    print ("Valeur␣de␣p␣:␣", p)
    if s - M[p] >= M[p]:
        return False
    return existeDec(M, s - M[p])
```

où M est un tableau représentant une suite supercroissante et s un entier naturel.

Question 6
Exécuter l'appel de `existeDec(ExM, 42))`, avec `ExM = [2,3,8,14,31]`, en donnant les affichages successifs et le résultat final.

Question 7
Soit $M = (M_0, \ldots, M_{n-1})$ une suite supercroissante et s un entier naturel non nul. Soit p le plus grand indice tel que $M_p \leq s$.

1. Montrer que, si $s = x_0M_0 + \ldots + x_{n-1}M_{n-1}$ avec $x_i \in \{0, 1\}$ pour tout $i \in \{0, \ldots n-1\}$, alors :

   (a) $x_j = 0$ si $j > p$,

   (b) $x_p = 1$.

   Indication : faire ces deux preuves en utilisant un raisonnement par l'absurde.

2. En déduire que :

   (a) si $s - M_p \geq M_p$ alors le problème de décomposition n'a pas de solution pour (M, s) (faire un raisonnement par la contraposée) ;

   (b) si $s - M_p < M_p$ alors le problème de décomposition a une solution pour (M, s) ssi il en a une pour $(M, s - M_p)$.

Question 8

Prouver, par récurrence forte sur s, que `existeDec(M,s)` se termine et renvoie la valeur `True` si s admet une décomposition et la valeur `False` sinon.

Indication : pour la base, on étudiera le cas où $s = 0$ et le cas où $0 < s < M[0]$.

Question 9
Calculer la complexité de `existeDec` dans le meilleur cas et dans le pire cas, en précisant quel est le meilleur cas et quel est le pire cas pour un tableau de taille n. Justifier la réponse.

Exercice 2 – Arbres Binaires, Arbres Binaires de Recherche - 10 points

Question 1 – Question de cours

Soit T un arbre binaire.

1. Rappelez les définitions inductives d'un arbre binaire étiqueté sur un ensemble E, de sa hauteur $h(T)$ et de son nombre de nœuds $n(T)$.

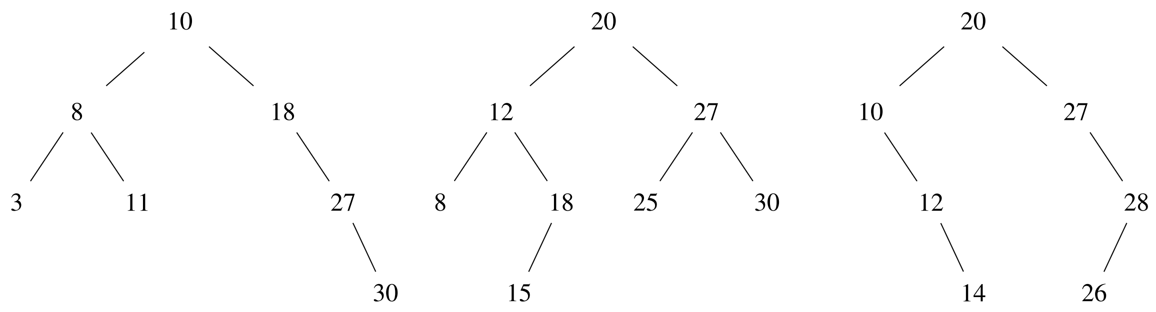
2. Montrez par <u>induction structurelle</u> que $h(T) \leq n(T) \leq 2^{h(T)} - 1$.

3. Quels sont les arbres binaires tels que $h(T) = n(T)$? Même question pour $n(T) = 2^{h(T)} - 1$.

Question 2 – Question de cours

1. Rappelez la définition <u>inductive</u> d'un arbre binaire de recherche.

2. Est-ce que les arbres suivants sont des arbres binaires de recherche. Dans la négative, justifiez votre réponse.



B_1 B_2 B_3

Question 3

Qu'affiche l'algorithme `mystere(B2,8,15)` où B_2 est l'arbre binaire de la question 2 ? Vous préciserez l'arbre des appels et les valeurs retournées à chaque appel.

La fonction `estABRvide(T)` retourne `true` si T est un arbre vide. $L_1 + L_2$ désigne la concaténation des listes L_1 et $L2$.

```
def mystere (T, a, b):
        if estABRvide(T):
                print "appel␣mystere␣(vide,",  a, ",",b, ")"
                print "appel␣mystere␣(vide,",  a, ",",b, ")␣retourne␣[]"
                return []
        L=[]
        print "appel␣mystere␣(", T.clef, ",", a,",", b, ")"
        if (a<T.clef):
                L=mystere(T.gauche, a,b)
        if (a<= T.clef) and (T.clef<=b):
                L=L+[T.clef]
        if (T.clef<b):
                L=L+mystere(T.droit, a,b)
        print  "l'appel␣mystere␣(", T.clef, a,b, ")␣retourne", L
        return L
```

Question 4

Pour tout arbre binaire de recherche T, soit la propriété $\Pi(T)$ suivante :

$\Pi(T)$: pour tout couple $(a, b) \in \mathbb{N}^2$ avec $a \leq b$, mystere(T, a, b) se termine et retourne toutes les clefs de T en ordre croissant qui sont dans l'intervalle $[a, b]$.

Démontez $\Pi(T)$ par induction structurelle sur l'ensemble des ABR.

Question 5
On souhaite implémenter la fonction `mystere` en utilisant des listes doublement chaînées circulaires.

1. Quelle est la complexité de la fusion de deux listes doublement chaînées circulaires ?
2. En déduire la complexité de la fonction `mystere` dans le pire cas et le meilleur des cas. Justifiez vos réponses.