Numéro d'anonymat :

Examen 2I003

Mardi 15 Janvier 2019, 2 heures
aucun document autorisé

Exercice 1 – Tas

Dans tout cet exercice, les arbres binaires sont étiquetés sur $\mathbb{N}$.

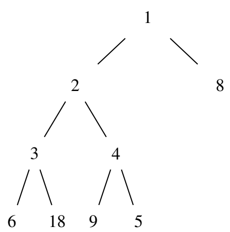
Question 1

Rappeler la définition **inductive** d'un arbre binaire étiqueté sur $\mathbb{N}$. Rappeler la définition d'un arbre binaire parfait. Rappeler la définition d'un tas.

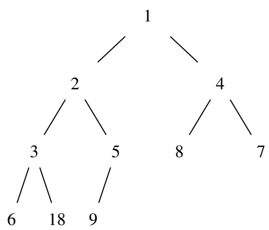
Question 2

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.

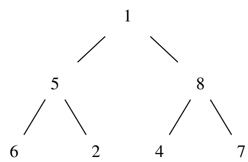
Arbre T_1 :



Arbre T_2 :



Arbre T_3 :

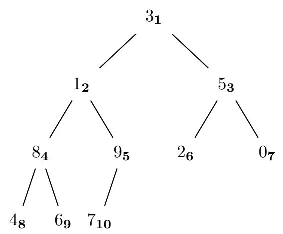


On rappelle qu'un arbre parfait de taille n peut être représenté au moyen d'un tableau $A[0..N]$, avec $N \geq n$, tel que :

— $A[0]$ contient la taille n de T

— les cases $A[1..n]$ sont remplies en parcourant T de gauche à droite, niveau par niveau.

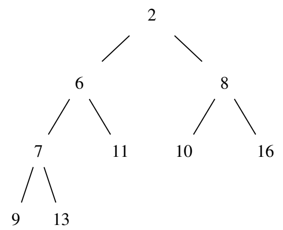
Ainsi, on peut numéroter les nœuds de l'arbre parfait T au moyen de leur position dans le tableau A comme illustré dans l'exemple suivant :



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

Question 3

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 4
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. Quelle en est la complexité en nombre de comparaisons dans le pire cas ?
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.

On dispose d'une procédure `echanger(A,i,j)` qui échange les valeurs en position i et en position j dans le tableau A.

On considère les procédures `ordonner` et `faireTas` suivantes :

```
def ordonner(A,n,i):
  if 2*i <= n:
    if 2*i + 1 <= n:
      if A[2*i] < A[2*i + 1]:
        j = 2*i
      else:
        j = 2*i + 1
    else:
      j = 2*i
    if A[i] > A[j]:
      echanger(A,i,j)
      ordonner(A,n,j)

def faireTas(A):
  n = len(A) - 1
  A[0] = n
  i = n // 2
  c = 0
  print('Pour␣c␣=',c,':␣i=', i, 'et␣A␣=', A)
  while i >= 1:
    c = c + 1
    ordonner(A,n,i)
    i = i - 1
    print('Pour␣c␣=',c,':␣i=', i, 'et␣A␣=', A)
  print('Tableau␣A␣:', A)
```

Rappel : `//` est l'opérateur de la division entière (par exemple `6//2` vaut 3, `7//2` vaut 3).

On admet que `faireTas(A)` transforme `A[1..n]` en un tas.

Question 5
Exécuter `faireTas(A)` avec `A=[0,3,1,5,8,9,2,0,4,6,7]`, en écrivant tous les affichages et en dessinant l'arbre parfait associé au tableau, à chaque étape.

Définition. La *profondeur* d'un nœud x dans un arbre binaire T est le nombre de nœuds se trouvant sur le chemin qui va de la racine à x. Par exemple, la racine est à profondeur 1, les fils de la racine sont à profondeur 2, etc.

Question 6
Notons $c(n)$ le nombre de comparaisons entre clés effectuées par `faireTas(A)` dans le pire cas.

1. Calculer $c(1)$, $c(3)$.
2. Dans cette question, on considère des arbres parfaits qui sont complets à tous les niveaux. La taille d'un tel arbre est de la forme $n = 2^h - 1$.
   (a) Montrer que, si $n = 2^h - 1$ et $n' = 2^{h+1} - 1$ alors $c(n') = c(n) + 2n$.
   Indication : si i est la position d'un nœud à profondeur k dans l'arbre parfait associé à un tableau A de taille $n = 2^h - 1$ alors `ordonner(A,n,i)` fait au plus $2(h - k)$ comparaisons entre clés.
   (b) En déduire que, si $n = 2^h - 1$, alors $c(n) = 2(n - h)$ (faire une récurrence sur $h \geq 1$).

3. Dans le cas général, la taille n d'un arbre parfait de hauteur h est telle que : $2^{h-1} < n \leq 2^h - 1$. En utilisant le résultat précédent et le fait que $c(n)$ est croissante, montrer que la complexité pire cas de `faireTas(A)` est en $O(n)$.

Exercice 2 – Somme et multiplication de polynômes

Dans cet exercice, $\mathcal{A}(x)$ et $\mathcal{B}(x)$ désignent deux polynômes de même degré $n-1 \geq 0$. On pose $\mathcal{A}(x) = \sum_{i=0}^{n-1} a_i x^i$ et $\mathcal{B}(x) = \sum_{i=0}^{n-1} b_i x^i$. Pour alléger les notations, les polynômes pourront également être désignés par respectivement $\mathcal{A}$ et $\mathcal{B}$. Ces polynômes sont stockés sous la forme de deux listes de n éléments $A = (a_0, a_1, \cdots, a_{n-1})$ et $B = (b_0, b_1, \cdots, b_{n-1})$.

Soit $W = (w_0, w_1, \cdots, w_{m-1})$ une liste de m entiers associée au polynôme $\mathcal{W}(x) = \sum_{i=0}^{m-1} w_i x^i$. Pour tout couple $(j,k) \in \mathbb{N} \times \mathbb{N}^\star$ avec $0 \leq j < k \leq m$, on pose en utilisant la convention du langage python, $W[j:k] = (w_j \cdots, w_{k-1})$. Le polynôme correspondant à la liste $W[j:k]$ est alors $\sum_{i=j}^{k-1} w_i x^{i-j}$.

Question 1

1. Dans cette question, on pose $\mathcal{A}(x) = 2 + 3x + x^2 + 4x^3$ et $\mathcal{B}(x) = 3 + 2x + 2x^2 + 5x^3$. Donnez les listes A et B.
2. On suppose dans cette question que $W = (4, 1, 9, 3, 2, 1, 8, 0, 2)$. Que vaut $W[2:6]$? Quel est le polynôme associé à $W[2:6]$?

Question 2

Soient les fonctions `somme(A,B)`, `multUnevaleur(v,A)` et `difference(A,B)` qui construisent une nouvelle liste correspondant respectivement aux polynômes $\mathcal{A}(x) + \mathcal{B}(x)$, $v \times \mathcal{A}(x)$ et $\mathcal{A}(x) - \mathcal{B}(x)$ sans se soucier du degré du polynôme obtenu qui est toujours stocké dans une liste de n éléments.

1. Quelle est la complexité de ces 3 fonctions si les listes sont stockées dans des tableaux ? Justifiez vos réponses.
2. Même question si les listes sont stockées dans des listes simplement chaînées. Justifiez vos réponses.

Question 3
Soit la fonction `sommeFacteur(W,A,B,k)` qui retourne la liste associée au polynôme $\mathcal{W}(x) + a_k x^k \times \mathcal{B}(x)$ pour $0 \leq k < n$. Quelle est la liste retournée par l'appel `sommeFacteur(W,A,B,1)` pour les listes $W = (4, 1, 9, 3, 2, 1, 8, 0, 2)$, $A = (2, 3, 1, 4)$ et $B = (3, 2, 2, 5)$?

On s'intéresse par la suite à calculer le produit $\mathcal{W}(x) = \mathcal{A}(x) \times \mathcal{B}(x)$. On rappelle que $\mathcal{A}(x)$ et $\mathcal{B}(x)$ sont des polynômes de degré $n - 1 \geq 0$. Pour cela, on considère le code python suivant :

```
def produit(A,B):
    n=len(A)
    W = [0]*(2*n-1)
    k=0
    while (k<len(A)) :
            W = sommeFacteur(W,A,B,k)
            print(W)
            k=k+1
    return W
```

Le fonction `len(A)` (resp. `len(B)`) retourne le nombre d'eléments de la liste A (resp. B). L'instruction `W = [0]*(2*n-1)` construit une liste de $2n - 1$ termes nuls.

Question 4

Exécutez `produit(A,B)` pour les polynômes $\mathcal{A}(x) = 2 + 3x + x^2 + 4x^3$ et $\mathcal{B}(x) = 3 + 2x + 2x^2 + 5x^3$.

Question 5

1. En supposant que $a_{n-1} \neq 0$ et $b_{n-1} \neq 0$, quel est le degré du polynôme $\mathcal{W}(x)$ en fonction de n ? En déduire la taille de la liste associée W en fonction de n. Justifiez votre réponse.
2. Soit k_0 et W_0 les valeurs initiales de k et de W et pour $j \in \{1, \cdots, n\}$, k_j et W_j les valeurs de k et de W à la fin de la j-ième itération (les itérations sont numérotées de 1 à n). $\mathcal{W}_j(x)$ désigne le polynôme associé à W_j.
   Montrez par récurrence sur j que $k_j = j$ et que W_j est la représentation du polynôme $\mathcal{W}_j(x) = \sum_{i=0}^{j-1} a_i x^i \times \mathcal{B}(x)$.
3. En déduire la validité de la fonction `produit`.

Question 6

Supposons que l'appel `sommeFacteur(W,A,B,k)` est en $\Theta(n)$ pour des tableaux et des listes simplement chaînées. Quelle est la complexité de la fonction `produit` si les listes sont stockées dans des tableaux ? Quelle est la complexité de la fonction `produit` si les listes sont stockées dans des listes simplement chaînées ? Justifiez vos réponses.

Question 7

Par la suite, on souhaite écrire une fonction pour accélérer le calcul du produit de 2 polynômes de même degré. On suppose que n est une puissance de 2, soit il existe $\ell \in \mathbb{N}$ avec $n = 2^\ell$. Soient alors les polynômes $\mathcal{A}_0$ et $\mathcal{A}_1$ définis par $\mathcal{A}_0 = \sum_{i=0}^{\frac{n}{2}-1} a_i x^i$ et $\mathcal{A}_1 = \sum_{i=0}^{\frac{n}{2}-1} a_{i+\frac{n}{2}} x^i$. On a alors $\mathcal{A} = \mathcal{A}_0 + x^{\frac{n}{2}} \mathcal{A}_1$. De la même manière, $\mathcal{B}_0 = \sum_{i=0}^{\frac{n}{2}-1} b_i x^i$ et $\mathcal{B}_1 = \sum_{i=0}^{\frac{n}{2}-1} b_{i+\frac{n}{2}} x^i$. On a alors $\mathcal{B} = \mathcal{B}_0 + x^{\frac{n}{2}} \mathcal{B}_1$.

1. Dans cette question, on pose $\mathcal{A}(x) = 2 + 3x + x^2 + 4x^3$ et $\mathcal{B}(x) = 3 + 2x + 2x^2 + 5x^3$. Que valent les polynômes $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{B}_0$, et $\mathcal{B}_1$?
2. Démontrez que dans le cas général $\mathcal{A}_1 \times \mathcal{B}_0 + \mathcal{A}_0 \times \mathcal{B}_1 = (\mathcal{A}_0 + \mathcal{A}_1) \times (\mathcal{B}_0 + \mathcal{B}_1) - \mathcal{A}_0 \times \mathcal{B}_0 - \mathcal{A}_1 \times \mathcal{B}_1$. En déduire que $\mathcal{A} \times \mathcal{B} = x^n \mathcal{A}_1 \times \mathcal{B}_1 + x^{\frac{n}{2}} ((\mathcal{A}_0 + \mathcal{A}_1) \times (\mathcal{B}_0 + \mathcal{B}_1) - \mathcal{A}_0 \times \mathcal{B}_0 - \mathcal{A}_1 \times \mathcal{B}_1) + \mathcal{A}_0 \times \mathcal{B}_0$.

Par la suite, on pose $\mathcal{P}_0 = \mathcal{A}_0 \times \mathcal{B}_0$, $\mathcal{P}_1 = \mathcal{A}_1 \times \mathcal{B}_1$ et $\mathcal{P}_2 = (\mathcal{A}_0 + \mathcal{A}_1) \times (\mathcal{B}_0 + \mathcal{B}_1)$. On obtient

$$\mathcal{A} \times \mathcal{B} = x^n \mathcal{P}_1 + x^{\frac{n}{2}}(\mathcal{P}_2 - \mathcal{P}_0 - \mathcal{P}_1) + \mathcal{P}_0.$$

Question 8

1. Dans cette question, on pose $\mathcal{A}(x) = 2 + 3x + x^2 + 4x^3$ et $\mathcal{B}(x) = 3 + 2x + 2x^2 + 5x^3$. Vérifiez que $\mathcal{P}_0 = 6 + 13x + 6x^2$, $\mathcal{P}_1 = 2 + 13x + 20x^2$, $\mathcal{P}_2 = 15 + 56x + 49x^2$ et $\mathcal{P}_2 - \mathcal{P}_0 - \mathcal{P}_1 = 7 + 30x + 23x^2$. En déduire que $\mathcal{A} \times \mathcal{B} = 6 + 13x + 13x^2 + 30x^3 + 25x^4 + 13x^5 + 20x^6$.
2. On suppose dans cette question que $\mathcal{A}(x) = a_0 + a_1 x$ et $\mathcal{B}(x) = b_0 + b_1 x$. Que valent les polynômes $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{B}_0$, et $\mathcal{B}_1$? Calculez $\mathcal{P}_0$, $\mathcal{P}_1$, $\mathcal{P}_2$.
3. Calculez $\mathcal{P}_0$, $\mathcal{P}_1$, $\mathcal{P}_2$ pour les couples de polynômes suivant :
   — $\mathcal{A}(x) = 2 + 3x$ et $\mathcal{B}(x) = 3 + 2x$
   — $\mathcal{A}(x) = 1 + 4x$ et $\mathcal{B}(x) = 2 + 5x$
   — $\mathcal{A}(x) = 3 + 7x$ et $\mathcal{B}(x) = 5 + 7x$

Question 9

Soit la fonction récursive `produitK(A,B)` suivante :

```
def produitK(A,B):
    print "produitK(",A,",",B, ")"
    n=len(A) ;
    A0=A[0:n/2] ; A1=A[n/2:n]
    B0=B[0:n/2] ; B1=B[n/2:n]
    if (n==1):
        W=[A[0]*B[0]]
    else:
        P0=produitK(A0, B0)
        P1=produitK(A1, B1)
        P2=produitK(somme(A0,A1), somme(B0,B1))
        W=calculProduit(P0,P1,P2)
    print A, '*', B, '=', W
    return W
```

L'appel `calculProduit(P0,P1,P2)` retourne la liste correspondant au polynôme $x^n\mathcal{P}_1 + x^{\frac{n}{2}}(\mathcal{P}_2 - \mathcal{P}_0 - \mathcal{P}_1) + \mathcal{P}_0$. Exécutez la fonction pour les listes $A = (2, 3, 1, 4)$ et $B = (3, 2, 2, 5)$. Vous donnerez la liste des affichages et l'arbre des appels.

Indication : Pour éviter les calculs inutiles, utilisez les résultats numériques des questions 7 et 8 et commencez par construire l'arbre des appels.

Question 10

Démontrez par récurrence la propriété suivante : $\Pi(\ell), \ell \geq 0$: pour tout couple de polynômes $\mathcal{A}$ et $\mathcal{B}$ de degré $n - 1 = 2^\ell - 1$, l'appel `ProduitK(A,B)` se termine et retourne $\mathcal{A} \times \mathcal{B}$.

Question 11 – Facultative

On suppose que `calculProduit(P0,P1,P2)` est de complexité $\Theta(n)$ pour des tableaux ou des listes simplements chaînées.

Démontrez que la complexité de la fonction `produitK(A,B)` pour deux polynômes de degré $n - 1$ vérifie $c(n) = 3c(\frac{n}{2}) + \alpha \times n$. En déduire que l'algorithme est $\mathcal{O}(\log_2(n) \times n^{\log_2(3)})$.

Indication : on peut remarquer que pour $\forall \ell \geq 0$, $3^\ell = 2^{\ell \log_2(3)}$.