Examen 2I003

Jeudi 17 décembre 2015, 2 heures
aucun document autorisé

Exercice 1 – Arbres binaires et tas - 4.5 points

Dans cet exercice, T désigne un arbre binaire, $n(T)$ son nombre de nœuds et $h(T)$ sa hauteur.

Question 1

Donnez la définition inductive de l'ensemble AB des arbres binaires dont les nœuds sont étiquetés par des éléments de $\mathcal{A}$. On rappelle que l'arbre vide est un arbre binaire. Donnez la définition (non inductive) d'un arbre parfait et d'un tas.

Solution:

L'ensemble AB des *arbres binaires* est définie par :

Base $\emptyset \in AB$

Ind. Si D et G sont des éléments de AB, pour tout $a \in \mathcal{A}$, $(a, G, D) \in AB$.

Un *arbre parfait* est un arbre binaire dont tous les niveaux sont pleins, sauf éventuellement le dernier qui est rempli le plus à gauche. Un *tas* est un arbre binaire parfait dont les étiquettes croissent de la racine vers les feuilles.

Question 2

Démontrez par induction sur la définition inductive de AB que, pour tout $T \in AB$, $h(T) \leq n(T) \leq 2^{h(T)} - 1$. En déduire que, pour tout arbre parfait T, $2^{h(T)-1} \leq n(T) < 2^{h(T)}$.

Solution:

Par induction sur la définition inductive des arbres binaires.

Base Si $T = \emptyset$ alors $h(T) = 0, n(T) = 0$ et $2^{h(T)} - 1 = 0$ donc la propriété est vraie.

Induction Soit G et D deux arbres binaires, supposons que la propriété soit vraie pour G et D. Soit $T = (x, G, D)$ alors :

$$h(T) = 1 + max(h(G), h(D)) \leq 1 + max(n(G), n(D)) \leq 1 + n(G) + n(D) = n(T)$$

$$\begin{aligned} n(T) &= 1 + n(G) + n(D) \\ &\leq 1 + 2^{h(G)} - 1 + 2^{h(D)} - 1 \\ &\leq 2.2^{max(h(G),h(D))} - 1 \\ &\leq 2^{1+max(h(G),h(D))} - 1 \\ &\leq 2^{h(T)} - 1 \end{aligned}$$

Conclusion La propriété est vraie pour tout arbre binaire.

Tour arbre parfait T est binaire, et donc $n(T) \leq 2^{h(T)} - 1 < 2^{h(T)}$. Les $h(T) - 1$ premiers niveaux de T sont entièrement remplis et contiennent donc exactement $2^{h(T)-1} - 1$ nœuds. Il y a au moins 1 nœud au niveau $h(T)$ donc $2^{h(T)-1} \leq n(T)$.

Question 3

1. Triez par ordre décroissant en utilisant un tri par tas les valeurs de la liste $L = (37, 42, 10, 2, 28, 16)$. Vous donnerez uniquement les valeurs successives du tableau courant après chaque opération de construction ou de destruction du tas.

2. Quelle est la complexité du tri par tas dans le pire des cas ? Justifiez votre réponse.

Solution:

1. Les valeurs successives du tableau sont :

1	37	42	10	2	28	16
2	37	42	10	2	28	16
3	10	42	37	2	28	16
4	2	10	37	42	28	16
5	2	10	37	42	28	16
6	2	10	16	42	28	37
5	10	28	16	42	28	2
4	16	28	37	42	10	2
3	28	42	37	16	10	2
2	37	42	28	16	10	2
1	42	37	28	16	10	2

2. La complexité dans le pire des cas du tri par tas est en $\mathcal{O}(n \log n)$. Chaque insertion et suppression dans un tas est en $\mathcal{O}(\log n)$. Il y a exactement $2(n-1)$ opérations sur le tas qui possède au plus n éléments, la complexité est donc bien dans le pire des cas en $\mathcal{O}(n \log n)$.

Exercice 2 – Graphes et arbres - 4 points

Dans cet exercice, $G = (V, E)$ est un graphe non orienté.

Question 1

Rappelez la définition du degré $d_G(x)$ pour tout $x \in V$. Calculer $d_G(x)$ pour le graphe $G = (V, E)$ avec $V = \{1, 2, 3, 4, 5\}$ et $E = \{\{1, 2\}, \{2, 3\}, \{1, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$.

Solution:

Le degré de tout sommet $x \in V$ est le nombre de sommets de V qui sont adjacents à x. Les degrés du graphe G donné sont :

x	1	2	3	4	5
$d_G(x)$	2	3	3	3	1

Question 2

Démontrez par récurrence sur le nombre d'arêtes que, pour tout graphe non orienté $G = (V, E)$, $\sum_{x \in V} d_G(x) = 2|E|$. En déduire que tout graphe possède un nombre pair de sommets de degré impair.

Solution:

La propriété à démontrer s'énonce par :
$\Pi(m)$, $m \geq 0$: pour tout graphe $G = (V, E)$ non orienté de m arêtes, $\sum_{x \in V} d_G(x) = 2|E|$. La propriété se démontre par récurrence faible.

Base Si $m = 0$, pour tout graphe G sans arête, $d_G(x) = 0$. $\Pi(0)$ est donc bien vérifiée.

Induction Soit $m > 0$ tel que $\Pi(m-1)$ soit vérifiée. Soit $G = (V, E)$ un graphe de m arêtes, $a = \{u, v\} \in E$, $E' = E - \{a\}$ et $G' = (V, E')$.

G' est un graphe qui possède $m-1$ arêtes, donc par récurrence, $\sum_{e \in E} d_{G'}(e) = 2(m-1)$. D'autre part, on observe que $d_G(u) = d_{G'}(u) + 1$, $d_G(v) = d_{G'}(v) + 1$, et que $\forall x \in V - \{u, v\}$, $d_G(x) = d_{G'}(x)$. On en déduit que

$$\sum_{x \in V} d_G(x) = 2 + \sum_{x \in V} d_{G'}(x) = 2(m-1) + 2 = 2m$$

et $\Pi(m)$ est vérifiée.

Conclusion La propriété est démontrée par récurrence faible.
On a $\sum_{x\in V} d_G(x) = \sum_{x\in V, d_G(v)pair} d_G(x) + \sum_{x\in V, d_G(x)impair} d_G(x) = 2.|E|$. Donc, la somme des degrés impairs est pair. On en déduit que l'on a un nombre pair de sommets de degré impair.

Question 3

Quelle est la définition d'un graphe connexe ? Quelle est la définition d'un graphe minimal connexe ? Quelle est la définition d'un arbre ?

Solution:
Un graphe G est *connexe* si pour tout couple de sommets $(u, v) \in V^2$, il existe une chaîne de u à v. Un graphe G est *minimal connexe* si pour toute arête $e = \{u, v\}$, le sous-graphe $G' = (V, E - \{e\})$ n'est pas connexe. Un *arbre* est un graphe non orienté connexe et sans cycle.

Question 4

Démontrez par l'absurde que, si G est un arbre, alors G est minimal connexe. Que pensez-vous de la réciproque ? Justifiez votre réponse.

Solution:
Supposons par l'absurde que G est un arbre qui n'est pas minimal connexe. Par définition d'un arbre G est connexe. Comme il n'est pas minimal connexe, il existe une arête $e = \{u, v\}$ telle que $G' = (V, E - \{e\})$ est également connexe. Il existe donc une chaîne ν dans G' de v à u. On en déduit que $e.\nu$ est un cycle de G, ce qui est impossible.
La réciproque est vraie, il s'agit du théorème du cours sur la caractérisation d'un arbre.

Exercice 3 – Tri par sélection du min et du max - 12 points

On rappelle que l'indexation des tableaux commence à 0. Si $T = [a_0, \ldots, a_i, \ldots, a_j, \ldots a_{n-1}]$, on note $T[i..j] = [a_i, \ldots, a_j]$. Dans tout l'exercice, T est un tableau d'entiers dont la taille est notée n.
On dispose de la fonction `indMinMax(T,g,d)` ainsi définie, pour $0 \leq g \leq d \leq n-1$:

```
def indMinMax(T, g, d):
    a = g ; z = g ; i = g + 1
    while i <= d:
        if T[i] < T[a]:
            a = i
        elif T[i] >= T[z]:
            z = i
        i = i + 1
    return (a, z)
```

La fonction `indMinMax(T,g,d)` renvoie le couple (a, z) où a est le plus petit indice appartenant à $[g..d]$ tel que $T[a]$ est le minimum de $T[g..d]$ et z est le plus grand indice appartenant à $[g..d]$ tel que $T[z]$ est le maximum de $T[g..d]$.

Question 1

1. On considère le tableau `T0 = [6,8,1,7,4,3,9,5]`. Exécuter l'appel de`indMinMax(T0,0,7)`, en donnant les valeurs de a, z et i à la fin de chaque itération, ainsi que la valeur retournée par la fonction.
2. Calculer le nombre minimum de comparaisons entre éléments du tableau effectuées par `indMinMax(T,g,d)`. Donner un exemple de tableau pour lequel `indMinMax` effectue le minimum de comparaisons.
3. Calculer le nombre maximum de comparaisons entre éléments du tableau effectuées par `indMinMax(T,g,d)`. Donner un exemple de tableau pour lequel `indMinMax` effectue le maximum de comparaisons.

4. Calculer le nombre maximum d'affectations (aux variables a et z) effectuées par `indMinMax(T,g,d)`. Donner un exemple de tableau pour lequel `indMinMax` effectue le maximum d'affectations.

Solution:

1. `indMinMax(T0,0,7)`

```
a= 0 z= 0 i= 1
a= 0 z= 1 i= 2
a= 2 z= 1 i= 3
a= 2 z= 1 i= 4
a= 2 z= 1 i= 5
a= 2 z= 1 i= 6
a= 2 z= 6 i= 7
a= 2 z= 6 i= 8
```

   La valeur retournée est $(2, 6)$.

   Remarque : la première ligne correspond aux valeurs de a, z et i à la fin de la 0-ème itération, elle n'est pas obligatoire.

2. `indMinMax(T,g,d)` effectue au minimum une comparaison par itération et il y a $d - g$ itérations donc `indMinMax(T,g,d)` effectue au minimum $d - g$ comparaisons.

   C'est le cas lorsque le tableau est rangé en ordre strictement décroissant, par exemple : $T = [5, 4, 3, 2, 1]$, $g = 0$, $d = 4$.

3. `indMinMax(T,g,d)` effectue au maximum deux comparaisons par itération et il y a $d - g$ itérations donc `indMinMax(T,g,d)` effectue au maximum $2(d - g)$ comparaisons.

   C'est le cas lorsque le plus petit élément du tableau est en position g, par exemple : $T = [1, 4, 5, 2, 3]$, $g = 0$, $d = 4$.

4. `indMinMax(T,g,d)` effectue deux affectations initiales puis `indMinMax(T,g,d)` au maximum une affectation par itération et il y a $d - g$ itérations donc `indMinMax(T,g,d)` effectue au maximum $2 + (d - g)$ affectations.

   Quelques exemples ($g = 0$, $d = 4$) : $T = [5, 4, 3, 2, 1]$ ou $T = [1, 2, 3, 4, 5]$ ou $T = [1, 1, 1, 1, 1]$, etc.

On dispose aussi d'une procédure `permuter(T,g,d,a,z)`, pour $0 \leq g < d \leq n - 1$, $g \leq a \leq d$, $g \leq z \leq d$ et $a \neq z$, qui transforme le tableau T en un tableau T' tel que :
- $T'[g..d]$ est une permutation de $T[g..d]$
- $T'[g] = T[a]$ et $T'[d] = T[z]$
- $T'[0..(g-1)] = T[0..(g-1)]$ et $T'[(d+1)..(n-1)] = T[(d+1)..(n-1)]$.

On considère la procédure `placerMinMax(T,g,d)` ainsi définie, pour $0 \leq g < d \leq n - 1$:

```
def placerMinMax(T,g,d):
    (a, z) = indMinMax(T,g,d)
    print('a=', a, 'z=', z)
    permuter(T, g, d, a, z)
    print('T[',g,']=', T[g], 'T[',d,']=', T[d])
```

Question 2

1. On considère le tableau `T0=[6,8,1,7,4,3,9,5]` et l'appel `placerMinMax(T0,1,5)`. Donner les affichages de cet appel.

   Remarque : il est inutile de connaître en détail l'ordre des éléments de $T[g + 1..d - 1]$ après l'appel `permuter(T,g,d,a,z)` pour répondre à cette question.

2. Notons T' le tableau obtenu après l'exécution de `placerMinMax(T,g,d)`, pour $0 \leq g < d \leq n - 1$. Montrer que :

— $T'[g..d]$ est une permutation de $T[g..d]$,
— $T'[g]$ est le plus petit élément de $T'[g..d]$,
— $T'[d]$ est le plus grand élément de $T'[g..d]$,
— $T'[0..(g-1)] = T[0..(g-1)]$ et $T'[(d+1)..(n-1)] = T[(d+1)..(n-1)]$.

3. On admet que `permuter(T,g,d,a,z)` n'effectue aucune comparaison entre éléments du tableau et effectue un nombre d'affectations aux éléments de T en $\Theta(1)$. Quel est le nombre de comparaisons entre éléments du tableau effectuées par `placerMinMax(T,g,d)` ? Quel est le nombre d'affections aux éléments du tableau effectuées par `placerMinMax(T,g,d)` ?

Solution:
Remarque : a et z vérifient bien l'hypothèse $a \neq z$ imposée par `permuter(T,g,d,a,z)`. En effet a est le plus petit indice appartenant à $[g..d]$ tel que $T[a]$ est le minimum de $T[g..d]$ et z est le plus grand indice appartenant à $[g..d]$ tel que $T[z]$ est le maximum de $T[g..d]$. Si le minimum est différent du maximum alors il est évident que $a \neq z$. Si le minimum est égal au maximum alors tous les éléments de $T[g..d]$ sont égaux, donc $a = g$, $z = d$ et on a fait l'hypothèse $g < d$.

1.
```
placerMinMax(T0,1,5)
a = 2               z = 1
T[ 1 ] = 1          T[ 5 ] = 8
```

2. L'exécution de `placerMinMax(T,g,d)` calcule le plus petit indice a appartenant à $[g..d]$ tel que $T[a]$ est le minimum de $T[g..d]$ et le plus grand indice z appartenant à $[g..d]$ tel que $T[z]$ est le maximum de $T[g..d]$ puis fait un appel à `permuter(T,g,d,a,z)`. Cet appel place $T[a]$, qui est le plus petit élément de $T[g..d]$ (et donc de $T'[g..d]$) en position g et place $T[z]$, qui est le plus grand élément de $T[g..d]$ (et donc de $T'[g..d]$) en position d. Les autres propriétés sont conséquences des propriétés vérifiées par `permuter(T,g,d,a,z)`.
3. `placerMinMax(T,g,d)` effectue au moins $d - g$ comparaisons et au plus $2(d - g)$ comparaisons. Elle effectue un nombre d'affectations aux éléments de T en $\Theta(1)$.

On considère la procédure `trierGD(T, g, d)` ainsi définie, pour $0 \leq g \leq d \leq n-1$:

```
def trierGD(T, g, d):
    if g < d:
        print('appel avec g=', g, 'et d=',d)
        placerMinMax(T, g, d)
        trierGD(T, g + 1, d - 1)
        print('retour T =', T)
```

Question 3
Exécuter l'appel de `trierGD(T0,0,7)`, avec `T0=[6,8,1,7,4,3,9,5]`, en donnant les affichages successifs (ne pas oublier les affichages faits par `placerMinMax(T,g,d)` !).

Solution:

```
trierGD(T0,0,7)
appel avec g= 0 et d= 7
a = 2               z = 6
T[ 0 ] = 1          T[ 7 ] = 9
appel avec g= 1 et d= 6
a = 5               z = 1
T[ 1 ] = 3          T[ 6 ] = 8
appel avec g= 2 et d= 5
a = 4               z = 3
T[ 2 ] = 4          T[ 5 ] = 7
appel avec g= 3 et d= 4
```

```
a = 3                z = 4
T[ 3 ] = 5           T[ 4 ] = 6
retour T = [1, 3, 4, 5, 6, 7, 8, 9]
retour T = [1, 3, 4, 5, 6, 7, 8, 9]
retour T = [1, 3, 4, 5, 6, 7, 8, 9]
retour T = [1, 3, 4, 5, 6, 7, 8, 9]
```

Question 4

Montrer, par récurrence sur $d-g$, que la procédure `trierGD(T,g,d)` trie le sous-tableau $T[g..d]$ en ordre croissant et laisse les autres éléments de T inchangés.

Solution:

On prouve, par récurrence sur $k = d - g$ la propriété

$\Pi(k)$: `trierGD(T,g,d)` trie le sous-tableau $T[g..d]$ en ordre croissant et laisse les autres éléments de T inchangés.

(B) Si $k <= 0$ (*i.e.* si $g \geq d$), la procédure laisse le tableau T inchangé ; le sous-tableau $T[g..d]$ est vide ou réduit à un seul élément, il est donc trié en ordre croissant.

(I) Soit g et d tels que $0 \leq g < d \leq n-1$, supposons que le résultat soit vrai pour $j < k = d - g$. L'exécution de `trierGD(T,g,d)` fait appel à `placerMinMax(T,g,d)` qui transforme le tableau T en un tableau T' tel que :

— $T'[g..d]$ est une permutation de $T[g..d]$,
— $T'[g]$ est le plus petit élément de $T'[g..d]$,
— $T'[d]$ est le plus grand élément de $T'[g..d]$,
— $T'[0..(g-1)] = T[0..(g-1)]$ et $T'[(d+1)..(n-1)] = T'[(d+1)..(n-1)]$.

Il y a ensuite un appel à `trierGD(T',g+1,d-1)` qui, par hypothèse de récurrence, trie le sous-tableau $T'[g+1..d-1]$ en ordre croissant. Par conséquent $T'[g..d]$ est trié en ordre croissant, et $T[g..d]$ aussi. Les autres éléments de T sont inchangés. Le résultat est donc vrai au rang k.

(C) La propriété est vraie pour $k \leq 0$ et se propage des rangs inférieurs à k au rang $k = d - g$, pour $0 \leq g \leq d \leq n-1$ donc elle est vraie pour tous g, d tels que $0 \leq g \leq d \leq n-1$.

Question 5

En utilisant la procédure `trierGD(T,g,d)`, définir une procédure `trier(T)` qui trie le tableau `T` en ordre croissant. Justifier la réponse.

Solution:

```
trierK(T) :
  trierGD(T,0,len(T)-1)
```

La procédure `trierGD(T,g,d)` trie le sous-tableau $T[g..d]$ en ordre croissant donc `trierK(T,0,len(T)-1)` trie le sous-tableau $T[0..n-1] = T$ en ordre croissant.

Question 6

1. Montrer que le nombre d'affectations aux éléments du tableau effectuées par `trier(T)` est en $\Theta(n)$.
2. Montrer que le nombre de comparaisons entre éléments du tableau effectuées par `trier(T)` est en $\Theta(n^2)$.

Solution:

1. Notons $c(k)$ le nombre d'affectations effectuées par `trierGD(T,g,d)` avec $k = d - g$. Il existe deux constantes c_1 et c_2 telles que $c(k-2) + c_1 \leq c(k) \leq c(k-2) + c_2$ si $k \geq 1$ (puisque le nombre d'affectations effectuées par `placerMinMax(T,g,d)` est en $\Theta(1)$). $c(k) = 0$ si $k \leq 0$. D'où $\frac{k}{2}c_1 \leq c(k) \leq \frac{k+1}{2}c_2$. Le nombre d'affectations aux éléments du tableau effectuées par `trier(T)` est donc en $\Theta(n)$.

2. Notons $d(k)$ le nombre de comparaisons effectués par `trierGD(T,g,d)` avec $k = d - g$. Alors :
   $d(k-2) + k \leq d(k) \leq d(k-2) + 2k$ si $k \geq 1$ et $d(k) = 0$ si $k \leq 0$.
   Donc
   $k + (k-2) + (k-4) + (k-6) + \ldots \leq d(k) \leq 2(k + (k-2) + (k-4) + (k-6) + \ldots)$
   Posons $S = k + (k-2) + (k-4) + (k-6) + \ldots$ alors $S + (k-1) + (k-3) + (k-5) + \ldots = \frac{k(k+1)}{2}$ et $S + (k-1) + (k-3) + (k-5) + \ldots = S + k - 1 + (k-2) - 1 + (k-4) - 1 + \ldots = S + S - p$ avec $p \approx \frac{k}{2}$. D'où $2S - p = \frac{k(k+1)}{2}$ et $S = (\frac{k(k+1)}{2} + p)/2$.
   Donc $(\frac{k(k+1)}{2} + p)/2 \leq d(k) \leq \frac{k(k+1)}{2} + p$, avec $p \approx \frac{k}{2}$.
   Le nombre de comparaisons entre éléments du tableau effectuées par `trier(T)` est donc en $\Theta(n^2)$.