Examen 2I003

Mercredi 4 Janvier 2017, 2 heures
aucun document autorisé

Exercice 1 – Le pic

On dit qu'un tableau `T[0..n-1]` de n entiers admet un *pic* s'il existe un indice p (compris au sens large entre 0 et $n-1$) tel que :
– le sous-tableau `T[0..p]` est strictement croissant
– le sous-tableau `T[p..n-1]` est strictement décroissant.
Par exemple :
– `[2,3,5,6,8,9,7,1]` admet un pic en position 5,
– `[45,13,6,4,2]` admet un pic en position 0,
– `[1,3,5,7,23]` admet un pic en position 4.
Dans cet exercice on considère des tableaux de n entiers qui admettent un pic et on étudie des algorithmes qui calculent la position de ce pic.

Le pic en itératif

On dispose de la fonction `Lambda(T, k)` ainsi définie, pour $0 \leq k \leq n-1$:

```
def Lambda(T, k):
    n = len(T)
    if k == n - 1:
        return True
    return T[k] > T[k + 1]
```

On considère la fonction `picIte(T)` ainsi définie :

```
def picIte(T):
    k = 0
    while not(Lambda(T, k)):
        k = k + 1
    return k
```

Question 1

Donner le résultat de `picIte(Ex)`, avec `ExT = [1,3,5,7,8,13,16,17,19,23,45,11,6,4]`.

<u>**Solution**</u>:
Le résultat de `picIte(ExT)` est 10.

Question 2

Montrer que `picIte(T)` se termine.

<u>**Solution**</u>:
k augmente de 1 chaque itration donc :
Ñ ou bien k prend une valeur k_0 telle que $T[k_0] > T[k_0+1]$ et alors `Lambda(T,k)` prend la valeur `True`,
Ñ ou bien k atteint la valeur $n-1$ et alors `Lambda(T,k)` prend la valeur `True`.
Dans les deux cas `picIte(T)` se termine.

Question 3

On note k_i la valeur de k à la fin de la i-ème itération (k_0 vaut 0). Montrer par récurrence sur i que, à la fin de la i-ème itération, $k_i = i$ et `T[0..i]` est strictement croissant.

Solution:
C'est vrai à la fin de l'itération 0 : $k_0 = 0$ et `T[0..0]` est strictement croissant. Supposons que ce soit vrai à la fin de la i-ème itération. À la fin de l'itération $i+1$, $k_{i+1} = k_i + 1 = i + 1$. Puisqu'il y a eu une $(i+1)$-ème itération, c'est que `Lambda(T,i)` avait la valeur `False` donc $T[i] \leq T[i+1]$, d'où $T[i] < T[i+1]$ (puisque le tableau croît strictement, puis décroît strictement). Par conséquent `T[0..i+1]` est strictement croissant.

Question 4

Déduire des deux questions précédentes que `picIte(T)` calcule la position du pic de T.

Solution:
Soit k la valeur renvoyée par `picIte(T)` alors :
– `T[0..k]` est strictement croissant,
– il n'y a pas de nouvelle itération donc `Lambda(T,k)` a la valeur `True`, soit parce que $k = n-1$ (dans ce cas le tableau est strictement croissant et on a un pic en $n-1 = k$), soit parce que $T[k] > T[k+1]$ et on a un pic en k.

Question 5

Quelle est la complexité (en nombre de comparaisons entre éléments du tableau) de `picIte(T)` dans le meilleur cas ? Dans le pire cas ?

Solution:
Il y a au minimum 1 comparaison (dans le cas d'un tableau strictement décroissant) et au maximum $n-1$ comparaisons (dans le cas d'un tableau strictement croissant). La complexité est donc en $\Omega(1)$ dans le meilleur cas et en $O(n)$ dans le pire cas.

Le pic en récursif

On considère la fonction `picRec(T,i,j)` ainsi définie, pour $0 \leq i \leq j \leq n-1$:

```python
def picRec(T, i, j):
    print("Recherche du pic entre ", i, " et ", j)
    if i == j:
        return i
    m = (i + j) // 2
    print("m = ", m)
    if T[m] < T[m + 1]:
        return picRec(T, m + 1, j)
    return picRec(T, i, m)
```

On rappelle que `//` est l'opérateur de la division entière (par exemple : `7//2 = 3, 8//2 = 4`).

Question 6

Exécuter l'appel de `picRec(ExT,0,13)`, avec `ExT = [1,3,5,7,8,13,16,17,19,23,45,11,6,4]`, en donnant les affichages successifs et le résultat final.

Solution:

```
picRec(ExT, 0, 13)
Recherche du pic entre  0  et  13
m =  6
Recherche du pic entre  7  et  13
m =  10
Recherche du pic entre  7  et  10
m =  8
```

```
Recherche du pic entre  9  et  10
m =  9
Recherche du pic entre  10  et  10
```

Le résultat final est 10.

Question 7

Soit i, j des entiers naturels tels que $i < j$. On pose $m = (i + j)\ //\ 2$ et $t = j - i + 1$. Montrer que :

a) $i \leq m < j$,

b) $1 \leq m - i + 1 < t$ et $1 \leq j - m < t$.

Solution:

a) $i < j \Rightarrow 2i < i + j < 2j \Rightarrow i < \dfrac{i+j}{2} < j$.

Donc $m = (i + j)\ //\ 2 \leq \dfrac{i+j}{2} < j$.

Puisque $\dfrac{i+j}{2}$ est supérieur à l'entier i, sa partie entière est supérieure ou égale à i, donc $i \leq m$.

b) $i \leq m < j \Rightarrow i - i + 1 \leq m - i + 1 < j - i + 1 \Rightarrow 1 \leq m - i + 1 < t$.

$i \leq m < j \Rightarrow j - j < j - m \leq j - i \Rightarrow 0 < j - m < j - i + 1 \Rightarrow 1 \leq j - m < t$.

Question 8

Montrer, par récurrence forte sur $t = j - i + 1$, que `picRec(T,i,j)` se termine, pour $t \geq 1$.

Solution:

Par récurrence forte sur $t = j - i + 1$.

(B) Si $t = 1$ (*i.e.* si $i = j$) alors `picRec(T,i,j)` se termine (et renvoie i).

(I) Soit $t > 1$, supposons que `picRec(T,i,j)` se termine pour $j - i + 1 < t$. Soit i et j tels que $j - i + 1 = t$, alors $i < j$ et, d'après la question précédente, $1 \leq m - i + 1 < t$ et $1 \leq j - m < t$.

Si $T[m] < T[m + 1]$ alors `picRec(T,i,j)` fait un appel récursif à `picRec(T,m+1,j)`.

On a $j - (m + 1) + 1 = j - m \leq j - i$ donc $1 \leq j - (m + 1) + 1 < t$. Par conséquent `picRec(T,m+1,j)` se termine et `picRec(T,i,j)` aussi.

Dans l'autre cas `picRec(T,i,j)` fait un appel récursif à `picRec(T,i,m)`.

On a $1 \leq m - i + 1 < t$, donc `picRec(T,i,m)` se termine et `picRec(T,i,j)` aussi.

(C) On a donc prouvé que `picRec(T,i,j)` se termine.

Question 9

Montrer, par récurrence forte sur $t = j - i + 1$, que `picRec(T,i,j)` calcule la position du pic de `T[i..j]`.

Solution:

Par récurrence forte sur $t = j - i + 1$.

(B) Si $t = 1$ (*i.e.* si $i = j$) alors `picRec(T,i,i)` renvoie i, qui est bien la position du pic de `T[i..i]`.

(I) Soit $t > 1$, supposons que `picRec(T,i,j)` renvoie la position du pic de `T[i..j]`, pour $j - i + 1 < t$. Soit i et j tels que $j - i + 1 = t$.

Si $T[m] < T[m + 1]$ alors le pic de `T[i..j]` n'est pas dans `T[i..m]`, il est donc dans `T[m+1..j]`. Par hypothèse de récurrence, `picRec(T,m+1,j)` renvoie la position du pic de `T[m+1..j]`, et donc la position du pic de `T[i..j]`.

Dans l'autre cas $T[m] > T[m + 1]$ donc le pic de `T[i..j]` n'est pas dans `T[m+1..j]`, il est donc dans `T[i..m]`. Par hypothèse de récurrence, `picRec(T,i,m)` renvoie la position du pic de `T[i..m]`, et donc la position du pic de `T[i..j]`.

(C) On a donc prouvé que `picRec(T,i,j)` renvoie la position du pic de `T[i..j]`.

Question 10

On considère la fonction `pic(T)` ainsi définie :

```
def pic(T):
    return picRec(T, 0, len(T)-1)
```

Déduire des deux questions précédentes que `pic(T)` calcule la position du pic de T.

Solution:

D'après les deux questions précédentes, `pic(T)` calcule la position du pic de `T[0..len(T)-1]`, et donc la position du pic de T.

Question 11

Quelle est la complexité (en nombre de comparaisons entre éléments du tableau) de `pic(T)` ? Justifier la réponse.

Solution:

Notons $c(n)$ la complexité de `pic(T)` pour un tableau de taille n, alors $c(1) = 0$ et $c(n) = 1 + c(n/2)$ pour $n > 1$.
Supposons que n soit de la forme 2^k alors :
$c(n) = c(2^k) = 1 + c(2^{k-1}) = 2 + c(2^{k-2}) = \dots k + c(2^{k-k}) = k + c(1) = k = \log_2(n)$.

La complexité de `pic(T)` est en $\Theta(\log n)$.

Exercice 2 – Graphes et arbres - 10 points

Dans cet exercice, $G = (V, E)$ désigne un graphe non orienté. $n = |V|$ désigne le nombre de sommets, et $m = |E|$ le nombre d'arêtes.

Question 1

Dans cette question, on suppose que $G = (V, E)$ est défini par $V = \{1, 2, 3, 4, 5\}$ et $E = \{\{1, 2\}, \{1, 3\}, \{1, 5\}, \{3, 4\}, \{2, 4\}\}$. Que valent m et n ? Représentez la matrice sommets-sommets M de G.

Solution:

Le graphe est composé de $n = 5$ sommets et $m = 5$ arêtes. M est une matrice 5×5 définie par

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Question 2

1. Qu'est ce qu'un graphe connexe ? Est-ce que le graphe de la question précédente est connexe ?
2. Qu'est ce qu'un arbre ? Est-ce que le graphe de la question précédente est un arbre ? Quelles sont les arêtes que l'on peut rajouter ou supprimer pour obtenir un arbre ?

Solution:

1. Un graphe non orienté $G = (V, E)$ est connexe si, pour tout couple de sommets $(u, v) \in V^2$, il existe une chaîne de u à v. Le graphe de la question précédente est connexe.
2. Un arbre est un graphe non orienté connexe et sans cycle. Le graphe précédent n'est pas un arbre du à la présence de cycles. Si on enlève par exemple l'arête $\{1, 3\}$ on obtient un arbre.

Question 3

Soit la relation $\mathcal{R}$ définie sur V^2 par $u\mathcal{R}v$ ssi $u = v$ ou si il existe une arête entre u et v.

1. Donnez la définition de $\mathcal{R}$ pour le graphe de la question 1.

2. Décrire le principe d'un algorithme qui permet de calculer la fermeture transitive $\mathcal{R}'$ de $\mathcal{R}$ dans le cas général. Ne pas écrire l'algorithme en (pseudo)-code. Quelle est sa complexité ?
3. Que vérifie $\mathcal{R}'$ si G est connexe ? En déduire un algorithme qui teste si un graphe est connexe. Quelle est sa complexité ?

Solution:

1. $\mathcal{R} = \{(1,1),(2,2),(3,3),(4,4),(5,5),(1,2),(2,1),(1,3),(3,1),(1,5),(5,1),(3,4),(4,3),(4,2),(2,4)\}$
2. Soit la matrice $M_{\mathcal{R}}$ de taille $n \times n$ définie par $M_{\mathcal{R}}[u,v] = 1$ si $u\mathcal{R}v$. Il suffit de calculer la matrice $M_{\mathcal{R}'} = \sum_{i=1}^{n} M_{\mathcal{R}}^i$ de manière booléene. On obtient ainsi $M[u,v] = 1$ ssi il existe une chaîne de u à v dans G. La complexité de cet algorithme est en $\Theta(n^3)$.
3. G est connexe si et seulement si, pour tout couple de sommet $(u,v) \in V^2$, $u\mathcal{R}v$. Il suffit donc de tester que la matrice $M_{\mathcal{R}'}$ obtenue précédemment est composée de 1, ce qui est en $\Theta(n^2)$. La complexité totale de l'algorithme est en $\Theta(n^3)$, puisqu'il faut calculer $M_{\mathcal{R}'}$.

Question 4

On rappelle qu'un graphe non orienté $G = (V,E)$ est minimal connexe si il est connexe et que, pour tout arête $e \in E$, $G' = (V, E - \{e\})$ ne l'est pas.

1. Représentez un graphe de 5 sommets qui est connexe, mais pas minimal connexe.
2. Démontrez que, si G est un arbre, alors G est minimal connexe. Utilisez la contraposée.
3. Démontrez que si G est minimal connexe, alors G est un arbre. Utilisez la contraposée.

Solution:

1. Un cycle n'est pas minimal connexe. Si on enlève n'importe quel arête, il y a toujours une chaîne entre tout couple de sommets.
2. Supposons que G ne soit pas minimal connexe. Si G n'est pas connexe, alors ce n'est pas un arbre.
   Supposons maintenant que G est connexe. Soit alors $e = \{u,v\}$ une arête de E et $G' = (V, E - \{e\})$ tel que G' soit connexe. Alors, il existe une chaîne μ de u à v dans G'. En rajoutant l'arête e à μ, on obtient un cycle dans G. On en déduit que G n'est pas un arbre.
3. Supposons que G n'est pas un arbre. Si G n'est pas connexe, alors il n'est pas minimal connexe.
   Supposons maintenant que G est connexe, mais qu'il possède un cycle c. Soit $e = \{u,v\}$ une arête de c. Le graphe $G' = (V, E - \{e\})$ est toujours connexe, puisque tout chaîne de G qui passe par l'arête e peut remplacer e par la sous-chaîne de G' correspondant à c de u à v. Donc, G n'est pas minimal connexe.

Question 5

1. Rappelez sans démonstration la relation entre le nombre de sommets et le nombre d'arêtes d'un graphe G connexe.
2. Démontrez que si $m = n - 1$ et que G est connexe, alors G est un arbre. La démonstration doit être effectuée par l'absurde.
3. Que pensez-vous de la réciproque (sans démonstration).
4. En déduire un algorithme simple qui teste si un graphe G est un arbre.

Solution:

1. Si G est connexe, alors $m \geq n - 1$.
2. Par l'absurde, supposons que G est un graphe connexe tel que $m = n-1$ et qui n'est pas un arbre. G est connexe, on en déduit qu'il n'est pas minimal connexe (sinon ce serai un arbre). Il existe donc une arête $e = \{u,v\}$ tel que $G' = (V, E - \{e\})$ est connexe. Si m' désigne le nombre d'arêtes de G', on a alors que $m' = n-2$. D'après la propriété du cours rappelée précédemment, G' n'est pas connexe, ce qui est contraire aux hypothèses.
3. La réciproque est vérifiée (résultat du cours).
4. Il suffit de tester que G est connexe en utilisant l'algorithme de question 3 et vérifier que $m = n - 1$.