2I003 – Interrogation longue 1 (1h) – corrigé

19 octobre 2017

L'interrogation est notée sur 25 points. Elle consiste en un problème composé de deux parties. La deuxième partie dépend de la première, mais **chaque question peut être traitée en admettant les questions précédentes**.
Le but du problème est d'étudier un algorithme qui calcule la signature d'une permutation.

1 Première partie (11pts).

Soient deux entiers $0 \leq j < n$ et `T` un tableau de taille n contenant des entiers tous distincts. Dans cette partie préliminaire, on étudie une fonction `nbSup` qui, étant donnés `T` et j, doit retourner le nombre d'incides $i \in \{0, \ldots, j-1\}$ tels que `T[i] > T[j]`.
La fonction `nbSup` est définie comme suit :

```
def nbSup(T,j):
    res = 0
    i = 0
    while i < j:
        if T[i] > T[j]:
            res = res + 1
        i = i + 1
    return res
```

Question 1 (2pts). Montrer que la fonction `nbSup(T,j)` se termine pour tout tableau `T` de taille n et tout $j \in \{0, \ldots, n-1\}$.

Solution : Soit $j \in \{0, \ldots, n-1\}$. À l'exécution de `nbSup(T,j)`, la boucle **`while` est effectuée j fois**, et toutes les autres instructions sont **élémentaires**, donc la fonction se termine bien.

Pour $i \in \{0, \ldots, j\}$, on définit r_i comme la valeur de `res` à la fin du i-ème tour de boucle (c'est-à-dire, après que l'instruction `i = i + 1` ait été effectuée pour la i-ème fois). Par convention, $r_0 = 0$. On définit aussi la propriété

$$\Pi(i) : \text{« } r_i \text{ est le nombre d'éléments } k \in \{0, \ldots, i-1\} \text{ tels que } \texttt{T[k] > T[j]} \text{ ».}$$

Question 2 (4pts). Montrer que pour tout $i \in \{0, \ldots, j\}$, la propriété $\Pi(i)$ est vérifiée.

Solution : On montre cela par récurrence faible.
• Initialisation. Pour $i = 0$, d'une part $r_0 = 0$, et d'autre part, le nombre de $k \in \{0, \ldots, -1\} = \varnothing$ vérifiant `T[k] > T[j]` est bien sûr 0. Donc $\Pi(0)$ est vérifiée.
• Hérédité. Soit $i \in \{0, \ldots, j-1\}$, et supposons $\Pi(i)$. Cela signifie que r_i est le nombre de $k \in \{0, \ldots, i-1\}$ tels que `T[k] > T[j]`. Dans le $(i+1)$-ème tour de boucle :
  – Si `T[i] > T[j]`, alors $r_{i+1} = r_i + 1$ d'après l'algorithme, et comme on a trouvé un nouvel élément supérieur à `T[j]`, $\Pi(i+1)$ est bien vérifiée dans ce cas.
  – Sinon, l'algorithme nous renseigne que $r_{i+1} = r_i$, et r_i compte toujours le nombre de $k \leq i$ tels que `T[k] > T[j]`, puisque `T[i] < T[j]`. Donc $\Pi(i+1)$ est aussi vrai dans ce cas.
• Conclusion. $\Pi(0)$ est vraie, et $\Pi(i) \Rightarrow \Pi(i+1)$, $\forall i \in \{0, \ldots, j-1\}$. Donc, par récurrence, $\Pi(i)$ est vraie pour tout $i \in \{0, \ldots, j\}$.

Question 3 (2pts). En déduire la validité de la fonction `nbSup`.

Solution : La dernière boucle est la j-ème, donc la propriété $\Pi(j)$ est vérifiée en fin de boucle. Celle-ci se traduit par : r_j est le nombre d'éléments $k \in \{0, \dots, j-1\}$ tels que `T[k] > T[j]`. Comme la fonction `nbSup(T,j)` se termine et retourne `res` $= r_j$, elle a bien le comportement attendu.

Question 4 (3pts). Exprimer en fonction de j la complexité exacte de `nbSup(T,j)`, en terme de nombre de comparaisons entre éléments du tableau.

Solution : Il y a j tours de boucle, et il y a une unique comparaison entre éléments du tableau par tour de boucle, donc la complexité exacte de `nbSup(T,j)` est j (si on veut l'exprimer approximativement, cela signifie que la complexité est en $\Theta(j)$).

2 Seconde partie (14pts).

Question 5 (1pt). Donner sans justifier la valeur de $(-1)^u$ suivant si u est pair ou impair.

Solution : $(-1)^u$ vaut 1 si u est pair, et -1 sinon.

On note S_n l'ensemble des permutations de $\{1, \dots, n\}$. Une permutation $\sigma \in S_n$ peut être représentée en machine par un tableau `T` tel que `T[i - 1]` $= \sigma(i)$ pour tout $i \in \{1, \dots, n\}$ (le décalage d'indice est dû au fait que le premier indice d'un tableau est 0).

Pour toute permutation $\sigma \in S_n$, on définit la **signature** de σ comme

$$\varepsilon(\sigma) := \prod_{1 \le i < j \le n} \text{signe}(\sigma(j) - \sigma(i)),$$

où la fonction signe est définie par $\text{signe}(x) := \begin{cases} -1 & \text{si } x < 0, \\ 1 & \text{si } x \ge 0. \end{cases}$

On **admet** que l'on a $\varepsilon(\sigma) = \prod_{j=2}^{n} (-1)^{u_j}$ où u_j est le nombre d'entiers $i \in \{1, \dots, j-1\}$ tels que $\sigma(i) > \sigma(j)$.

Soit enfin la fonction `signature(T,n)`, qui a pour but de retourner la signature d'une permutation $\sigma \in S_n$, qui encodée comme un tableau `T` de taille n.

```
def signature(T,n):
    if (n == 1):
        u = 0            # rajouté lors de l'interro
        res = 1
    else:
        res = signature(T,n-1)
        u = nbSup(T,n-1)
        if (u % 2 == 1):
            res = - res
    print("u = ", u, "et signature(", T, ", ", n, ") = ", res)
    return res
```

Question 6 (4pts). Exécuter `signature(T,n)` avec `T = [3,2,4,1]` et `n` $= 4$, puis donner sa valeur de retour.

Solution :

```
u = 0 et signature( [3, 2, 4, 1] , 1 ) = 1
u = 1 et signature( [3, 2, 4, 1] , 2 ) = -1
u = 0 et signature( [3, 2, 4, 1] , 3 ) = -1
u = 3 et signature( [3, 2, 4, 1] , 4 ) = 1
```

La valeur retournée est 1.

Question 7 (6pts). Démontrer la terminaison et la validité de la fonction `signature(T,n)`. On pourra montrer la propriété $\mathcal{P}(k)$: « `signature(T,`k`)` termine et renvoie $\prod_{j=2}^{k}(-1)^{u_j}$ » pour des valeurs appropriées de k.

Solution : On va montrer par récurrence faible la propriété suivante, pour tout $k \in \{1, \dots, n\}$:
$\mathcal{P}(k)$: « `signature(T,`k`)` se termine et renvoie $\prod_{j=2}^{k}(-1)^{u_j}$ ».
• Initialisation. Pour $k = 1$, la fonction `signature(T,1)` se termine bien, et renvoie 1. Par ailleurs, tout produit indexé par l'ensemble vide vaut 1. Donc $\mathcal{P}(1)$ est vérifiée.
• Hérédité. Soit $k \in \{2, \dots, n\}$, et supposons $\mathcal{P}(k-1)$. On veut montrer $\mathcal{P}(k)$. On remarque d'abord que la fonction `signature(T,k)` se termine, puisque la seule instruction non élémentaire qu'elle contient est l'appel à `signature(T,k-1)`, qui se termine bien par hypothèse de récurrence. Concernant la valeur de retour, d'après les questions 3 et 5, `signature(T,k)` renvoie le produit entre $(-1)^{u_k}$ et la valeur de retour de `signature(T,k-1)`. Or, cette dernière est égale à $\prod_{j=2}^{k-1}(-1)^{u_j}$ par hypothèse de récurrence. En effectuant le produit, on observe que `signature(T,k)` renvoie donc $\prod_{j=2}^{k}(-1)^{u_j}$, ce qui signifie que $\mathcal{P}(k)$ est satisfaite.
• Conclusion. On conclut que `signature(T,n)` renvoie bien le produit $\prod_{j=2}^{n}(-1)^{u_j}$, **qui est égal à la signature $\varepsilon(\sigma)$ de la permutation σ**, d'après l'énoncé.

Question 8 (3pts). Exprimer en fonction de n la complexité de `signature(T,n)` en nombre de comparaisons entre éléments du tableau `T` de taille n.

Solution : Les comparaisons entre éléments du tableau sont toutes contenues dans l'appel à `nbSup(T,n-1)`, qui en effectue $n-1$ d'après la question 4. Donc, si l'on note c_n la complexité de `signature(T,n)`, on a la **relation de récurrence** suivante :

$$\forall n > 1,\ c_n = c_{n-1} + (n-1)$$

Par ailleurs, on note que $c_1 = 0$. Donc,

$$\begin{aligned} c_n &= c_{n-1} + (n-1) = c_{n-2} + (n-2) + (n-1) = \cdots = c_1 + 1 + \cdots + (n-2) + (n-1) \\ &= 0 + \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2) \end{aligned}$$

Remarque : Pour ceux qui trouvent l'étape d'induction « $= \cdots =$ » trop informelle, sachez qu'il est possible de faire ce calcul plus proprement. Par exemple, on pourrait montrer par récurrence que $\forall i \in \{1, \dots, n-1\}$, on a $c_n = c_{n-i} + \sum_{j=n-i}^{n-1} j$. Puis on conclurait avec $i = n-1$.