Correction TD3

Stéphane Devismes

6 novembre 2023

1 Exercice 1

1.1 Question (a) (spécification)

Soit T un tableau d'entiers de $N \in \mathbb{N}^*$ cases indicées de 1 à N.

La fonction $IVM(T)$ renvoie un entier naturel min tel que $min \in [1..N] \wedge \forall j \in [1..N], T[min] \leq T[j]$.

1.2 Question (b) (idée)

Si min est l'indice de la plus petite valeur stockée entre les indices 1 et $i-1$ du tableau T avec $1 < i \leq N$, alors la plus petite valeur stockée dans T entre les indices 1 et i est soit à l'indice min soit à l'indice i.

- Comment initialise-t-on ? l'indice 1 est l'indice de la plus petite valeur stockée entre les indices 1 et 1 $(= 2-1)$. Donc $min = 1$ et $i = 2$.
- Comment s'arrête-t-on ? Si $i = N+1$, alors min est l'indice de la plus petite valeur stockée entre les indices 1 et $N+1-1 = N$ du tableau T.

1.3 Question (c) (algorithme)

Algorithme 1 Fonction IVM avec résultat entier naturel.

Entrées

T : tableau d'entiers indicés de 1 à N avec $N \in \mathbb{N}^*$

Variables

min, i : entiers naturels

Programme

1: $min = 1$; $i = 2$
   /* P_0 : $i \in [2..N+1] \wedge min \in [1..i[\wedge (\forall j \in [1..i[, T[min] \leq T[j])$ */
2: **Tant que** $i \leq N$ **faire**
   /* P_1 : $i \in [2..N] \wedge min \in [1..i[\wedge (\forall j \in [1..i[, T[min] \leq T[j])$ */
3: **Si** $T[i] < T[min]$ **alors**
   /* P_2 : $i \in [2..N] \wedge (\forall j \in [1..i], T[i] \leq T[j])$ */
4: $min \leftarrow i$
   /* P_3 : $i \in [2..N] \wedge min \in [1..i] \wedge (\forall j \in [1..i], T[min] \leq T[j])$ */
5: **Fin Si**
   /* P_4 : $i \in [2..N] \wedge min \in [1..i] \wedge (\forall j \in [1..i], T[min] \leq T[j])$ */
6: $i \leftarrow i + 1$
   /* P_5 : $i \in [2..N+1] \wedge min \in [1..i[\wedge (\forall j \in [1..i[, T[min] \leq T[j])$ */
7: **Fin Tant que**
   /* P_6 : $min \in [1..N] \wedge (\forall j \in [1..N], T[min] \leq T[j])$ */
8: `renvoyer` min

1.4 Question (d) (terminaison)

On pose le prédicat suivant :

$\forall k \in \mathbb{N}$, $PA(k)$: (($k = N - i + 1$ au moment d'exécuter la ligne 2) $\Rightarrow$ (le nombre de tours de boucle est fini)).

Montrons par récurrrence que $PA(k)$ est vérifié pour tout $k \in \mathbb{N}$.

Initialisation : Si $k = 0$ au moment d'exécuter la ligne 2, on a $N - i + 1 = 0$. Donc, $i > N$ (précisément $i = N + 1$) et on entre pas dans la boucle. Ainsi, $PA(0)$ est vérifié.

Hérédité : Il faut montrer que $\forall k \in \mathbb{N}$, $(PA(k) \vDash PA(k+1))$. (hérédité simple)

On précise ce que signifie $PA(k+1)$:

$PA(k+1)$: (($k + 1 = N - i + 1$ au moment d'exécuter la ligne 2) $\Rightarrow$ (le nombre de tours de boucle est fini)

Si $k + 1 \neq N - i + 1$ au moment d'exécuter la ligne 2, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $PA(k+1)$ est trivialement vérifié.

Nous étudions maintenant le cas où $k + 1 = N - i + 1$ au moment d'exécuter la ligne 2. On pose α la valeur de i à ce moment là. Donc, $k + 1 = N - \alpha + 1$, ce qui implique $k = N - \alpha$. Puisque N est constant et qu'à la fin du tour de boucle, on a $i = \alpha + 1$, nous déduisons que $\alpha = i - 1$ et $k = N - i + 1$ à la fin du tour de boucle. D'où, lorsqu'on atteint la ligne 2 après le tour de boucle, on a $k = N - i + 1$: par hypothèse de récurrence, après un tour de boucle, le nombre de tours de boucle restant est fini. Ainsi, lorsque $k + 1 = N - i + 1$ au moment d'exécuter la ligne 2, le nombre de tours de boucle à exécuter est fini : $PA(k+1)$ est vrai. □

À part la boucle « tant que », toutes les instructions de l'algorithme sont élémentaires. Or, puisque $PA(k)$ est vrai pour tout $k \in \mathbb{N}$, le nombre de tours de boucle a exécuté est toujours fini et par suite, l'algorithme termine. □

1.5 Question (e) (bon résultat)

Soit

$$PBR \equiv i \in [2..N+1] \wedge min \in [1..i[\wedge (\forall j \in [1..i[, T[min] \leq T[j])$$

Nous allons démontrer par récurrence que $\forall t \in \mathbb{N}$, $P(t)$: le nombre de tours complets de boucle exécutés est $t \Rightarrow PBR$ est vrai au moment d'exécuter la ligne 2.

Initialisation : Lorsqu'on arrive en ligne 2 après 0 tour de boucle, on a P_0. Donc, on a trivialement PBR (P_0 est identique à PBR). Ainsi, $P(0)$ est vérifié.

Hérédité : Il faut montrer que $\forall t \in \mathbb{N}$, $(P(t) \vDash P(t+1))$. (hérédité simple)

On précise ce que signifie $P(t+1)$:

$P(t+1)$: le nombre de tours complets de boucle exécutés est $t + 1 \Rightarrow PBR$ est vrai au moment d'exécuter la ligne 2.

Supposons que le nombre de tours complets de boucle exécutés est $t + 1$. Considérons les t premiers tours de boucle. Puisque que $t \geq 0$, nous pouvons appliquer l'hypothèse de récurrence : lorsque le nombre de tours complets de boucle exécutés est t, on a PBR au moment d'exécuter la ligne 2.

Lorsque l'on commence le $t + 1^{\text{ème}}$ tour de boucle, on a de plus $i \leq N$ qui est vrai en ligne 2. Donc, on a P_1 qui est vrai après le test du « tant que » et donc également en ligne 3. Deux cas sont alors possible en fonction du test en ligne 3 :

- $T[i] < T[min]$: Par transitivité et P_1, on a $\forall j \in [1..i], T[i] \leq T[j]$ et $i \in [2..N]$ en ligne 3 (P_2). Donc, on a $(\forall j \in [1..i], T[min] \leq T[j])$, $i \in [2..N]$ et $min \in [1..i]$ après la ligne 4 (P_3). Ainsi, P_4 est vérifié après le « si ».
- $T[i] \geq T[min]$: En ligne 3, on a $min \in [1..i]$ et $i \in [2..N]$, de plus, de P_1 et $T[i] \geq T[min]$, on déduit aussi $\forall j \in [1..i], T[min] \leq T[j]$. Ainsi, P_4 est vérifié après le « si ».

Donc, P_4 est vérifié avant l'exécution de la ligne 6. Or, cela implique que P_5 est vrai après l'exécution de la ligne 6 et donc à la fin du tour de boucle, mais aussi quand on revient en ligne 2. D'où, $P(t+1)$ est vrai. □

Nous pouvons maintenant utiliser $P(t)$ pour établir le bon résultat. D'après la section 1.4, on sait que la boucle « tant que » termine. Donc, il existe $t_f \in \mathbb{N}$ tel que après t_f tours complets de boucle exécutés, le test $i \leq N$ est faux en ligne 2, ou de manière équivalente $i > N$ est vrai en ligne 2.

— Si $t_f = 0$ alors puisque N est un entier naturel supérieur ou égal à 1, $i > N$ et $i = 2$, on a $N = 1$ et donc $i = N + 1$ en ligne 2 après t_f tours complets de boucle exécutés.

— Sinon, au début du $t_f{}^{\text{ème}}$ tour de boucle, on a $i \leq N$ en ligne 2. Or, après le tour de boucle, on a $i > N$ en ligne ligne 2. Enfin, la valeur de i est incrémenté de 1 à chaque tour de boucle. D'où, $i = N + 1$ en ligne 2 après t_f tours complets de boucle exécutés.

Ainsi, dans tous les cas, on a $i = N + 1$ en ligne 2 après t_f tours complets de boucle exécutés.

De $PBR(t_f)$ et $i = N + 1$, on déduit que P_6 est vrai au moment du renvoi de min, ce qui donne le bon résultat de l'algorithme. □

1.6 Question (f) (complexité)

On pose le prédicat suivant :

$\forall k \in \mathbb{N}$, $PNT(k)$: (($k = N - i + 1$ au moment d'exécuter la ligne 2) ⇒ (la boucle « tant que » termine après k tours)).

Montrons par récurrrence que $PNT(k)$ est vérifié pour tout $k \in \mathbb{N}$.

Initialisation : Si $k = 0$ au moment d'exécuter la ligne 2, on a $N - i + 1 = 0$. Donc, $i > N$ (précisément $i = N + 1$) et on entre pas dans la boucle. Ainsi, la boucle « tant que » termine après 0 tour : $PNT(0)$ est vrai.

Hérédité : Il faut montrer que $\forall k \in \mathbb{N}$, $(PNT(k) \vDash PNT(k+1))$. (hérédité simple)

On précise ce que signifie $PNT(k+1)$:

$PNT(k+1)$: (($k + 1 = N - i + 1$ au moment d'exécuter la ligne 2) ⇒ (la boucle « tant que » termine après $k + 1$ tours))

Si $k + 1 \neq N - i + 1$ au moment d'exécuter la ligne 2, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $PNT(k+1)$ est trivialement vérifié.

Nous étudions maintenant le cas où $k + 1 = N - i + 1$ au moment d'exécuter la ligne 2. On pose α la valeur de i à ce moment là. Donc, $k + 1 = N - \alpha + 1$, ce qui implique $k = N - \alpha$. Puisque N est constant et qu'à la fin du tour de boucle, on a $i = \alpha + 1$, nous déduisons que $\alpha = i - 1$ et $k = N - i + 1$ à la fin du tour de boucle mais aussi après le premier tour de boucle lorsqu'on atteint la ligne 2. Donc, par hypothèse de récurrence, la boucle « tant que » termine après k tours de boucle supplémentaires. Au total, lorsque $k + 1 = N - i + 1$ au moment d'exécuter la ligne 2, la boucle « tant que » termine après $k + 1$ tours de boucle : $PNT(k+1)$ est vrai. □

À part la boucle « tant que », toutes les instructions de l'algorithme prennent un temps constant. De plus, la première fois qu'on atteint la ligne 2, on a $N - i + 1 = N - 1$. Donc, $N - i + 1 = N - 1$ et $PNT(N-1)$ impliquent que la complexité en temps de notre algorithme est en $\Theta(N)$ pour un tableau de N cases. □

2 Exercice 2

2.1 Question (a)

Soit T un tableau d'entiers de N cases indicées de 1 à N. Soit Ti la configuration initiale T. Soit Tf la configuration finale de T. On a les deux post-conditions suivantes :

1. Tf est une permutation de Ti : $\forall v \in Ti, (|Ti|_v = |Tf|_v)$.
2. Tf est trié par ordre croissant : $\forall i \in [1..N-1], Tf[i] \leq Tf[i+1]$.

2.2 Question (b) (idée)

Soit $d \in [1..N]$. Nous allons utiliser la fonction *Echange* et la variante de IVM suivante : la fonction $IVM\_REC(T, d)$ renvoie un entier naturel min tel que $min \in [d..N] \wedge (\forall j \in [d..N], T[min] \leq T[j])$. IVM_REC s'exécute en $O(N)$ (*cf.* exercice 1).

Si, pour un entier naturel $k < N$, on a $\forall i \in [1..k-1], \forall j \in [i..N], Tf[i] \leq Tf[j]$, il suffit d'échange les valeurs d'indices k et $IVM\_REC(T, k)$ pour obtenir $\forall i \in [1..k], \forall j \in [i..N], Tf[i] \leq Tf[j]$.

— Comment initialise-t-on ? Pour $k = 1$, on a $\forall i \in [1..k-1], \forall j \in [i..N], Tf[i] \leq Tf[j]$ qui est trivialement vrai.

— Comment s'arrête-t-on ? Pour $k = N$, $\forall i \in [1..k-1], \forall j \in [i..N], Tf[i] \leq Tf[j]$ implique que le tableau est trié.

2.3 Question (c) (algorithme)

Algorithme 2 Procédure de tri.

Entrées-Sorties

T : tableau d'entiers indicés de 1 à N

Variables

min, k : entiers naturels

Programme

1: $k = 1$
   /* $P_0 : k \in \mathbb{N}^* \wedge \forall i \in [1..k-1], \forall j \in [i..N], T[i] \leq T[j]$ */
2: **Tant que** $k < N$ **faire**
   /* $P_1 : k \in [1..N-1] \wedge \forall i \in [1..k-1], \forall j \in [i..N], T[i] \leq T[j]$ */
3: $min \leftarrow IVM\_REC(T, k)$
   /* $P_2 : P1 \wedge min \in [k..N] \wedge j \in [k..N], T[min] \leq T[j]$ */
4: $Echange(T, k, min)$
   /* $P_3 : k \in [1..N-1] \wedge \forall i \in [1..k], \forall j \in [i..N], T[i] \leq T[j]$ */
5: $k \leftarrow k + 1$
   /* $P_4 : k \in [1..N] \wedge \forall i \in [1..k-1], \forall j \in [i..N], T[i] \leq T[j]$ */
6: **Fin Tant que**
   /* $P_5 : \forall i \in [1..N-1], \forall j \in [i..N], T[i] \leq T[j]$ */

2.4 Question (d) (terminaison)

On pose le prédicat suivant :

$\forall t \in \mathbb{N}$, $PA(t)$: (($t = N - k$ au moment d'exécuter la ligne 2) $\Rightarrow$ (le nombre de tours de boucle est fini)).

Montrons par récurrrence que $PA(t)$ est vérifié pour tout $t \in \mathbb{N}$.

Initialisation : Si $t = 0$ au moment d'exécuter la ligne 2, on a $N - k = 0$. Donc, $k \geq N$ (précisément $k = N$) et on entre pas dans la boucle. Ainsi, $PA(0)$ est vérifié.

Hérédité : Il faut montrer que $\forall t \in \mathbb{N}$, $(PA(t) \vDash PA(t+1))$. (hérédité simple)

On précise ce que signifie $PA(t+1)$:

$PA(t+1)$: (($t + 1 = N - k$ au moment d'exécuter la ligne 2) $\Rightarrow$ (le nombre de tours de boucle est fini)

Si $t + 1 \neq N - k$ au moment d'exécuter la ligne 2, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $PA(t+1)$ est trivialement vérifié.

Nous étudions maintenant le cas où $t + 1 = N - k$ au moment d'exécuter la ligne 2. On pose α la valeur de k à ce moment là. Donc, $t + 1 = N - \alpha$, ce qui implique $t = N - \alpha - 1 = N - (\alpha + 1)$. Puisque N est constant et qu'à la fin du tour de boucle, on a $k = \alpha + 1$, nous déduisons que $t = N - k$ à la fin du tour de boucle. D'où,

lorsqu'on atteint la ligne 2 après le tour de boucle, on a $t = N - k$: par hypothèse de récurrence, après un tour de boucle, le nombre de tours de boucle restant est fini. Ainsi, lorsque $t + 1 = N - k$ au moment d'exécuter la ligne 2, le nombre de tours de boucle à exécuter est fini : $PA(t + 1)$ est vrai. □

À part la boucle « tant que », toutes les instructions de l'algorithme sont soit élémentaires soit des appels de fonctions qui terminent par hypothèse. Or, puisque $PA(t)$ est vrai pour tout $t \in \mathbb{N}$, le nombre de tours de boucle a exécuté est toujours fini et par suite, l'algorithme termine. □

2.5 Question (e) (bon résultat)

Vérifions tout d'abord qu'à chaque fois que l'on utilise les fonctions IVM_REC ou $Echange$, leurs pré-conditions sont vérifiées.

Considérons tout d'abord la fonction IVM_REC. Nous remarquons que k est initialisé à 1 et ne décroît jamais. Or, on exécute IVM_REC, k est passé en paramètre et à ce moment là $k < N$ (grâce au test de boucle). Donc, IVM_REC est toujours appelé avec un indice de tableau valide.

Considérons tout d'abord la fonction $Echange$. La variable Min est un indice du tableau juste avant l'appel à $Echange$ car il est affecté à l'aide de IVM_REC qui est toujours appelé avec un indice de tableau valide. Enfin, l'autre paramètre $Echange$ lors de son appel est k qui est un indice valide du tableau, comme vu dans le point précédent.

Remarque 1. *Lorsque les fonctions IVM_REC ou Echange sont utilisées, leurs pré-conditions sont vérifiées.*

Nous démontrons maintenant le bon résultat de la procédure de tri. Tout d'abord, si $N = 0$ alors le tableau est trivialement trié après l'appel. Concentrons nous maintenant sur le cas $N \in \mathbb{N}^*$.

Soit

$$PBR \equiv k \in [1..N] \wedge \forall i \in [1..k-1], \forall j \in [i..N], T[i] \leq T[j]$$

Nous allons démontrer par récurrence que $\forall t \in \mathbb{N}$, $P(t)$: le nombre de tours complets de boucle exécutés est $t \Rightarrow PBR$ est vrai au moment d'exécuter la ligne 2.

Initialisation : Lorsqu'on arrive en ligne 2 après 0 tour de boucle, on a $k \in [1..N]$ puisque $k = 1$ et $N \in \mathbb{N}^*$, par hypothèse. De plus, puisque $k = 1$, on a $\forall i \in [1..k-1], \forall j \in [i..N], T[i] \leq T[j]$ qui est trivialement vérifié. Donc, on a trivialement PBR et $P(0)$ est vérifié.

Hérédité : Il faut montrer que $\forall t \in \mathbb{N}$, $(P(t) \vDash P(t+1))$. (hérédité simple)

On précise ce que signifie $P(t + 1)$:

$P(t + 1)$: le nombre de tours complets de boucle exécutés est $t + 1 \Rightarrow PBR$ est vrai au moment d'exécuter la ligne 2.

Supposons que le nombre de tours complets de boucle exécutés est $t + 1$. Considérons les t premiers tours de boucle. Puisque que $t \geq 0$, nous pouvons appliquer l'hypothèse de récurrence : lorsque le nombre de tours complets de boucle exécutés est t, on a PBR au moment d'exécuter la ligne 2.

Lorsque l'on commence le $t + 1^{\text{ème}}$ tour de boucle, on a de plus $k < N$ qui est vrai en ligne 2. Donc on a P_1 qui est vrai après le test du « tant que » et donc également en ligne 3. De plus, $min \in [k..N]$ et $\forall i \in [k..N]$, $T[min] \leq T[i]$ après la ligne 3 (P_2). Ainsi, après la ligne 4, on a $\forall i \in [k..N]$, $T[k] \leq T[i]$ et P_1, donc P_3. Ainsi, après la ligne 5 et donc à la fin du tour de boucle, on a P_4, c'est-à-dire PBR. D'où, $P(t + 1)$ est vrai. □

Nous pouvons maintenant utiliser $P(t)$ pour établir le bon résultat. D'après la section 1.4, on sait que la boucle « tant que » termine. Donc, il existe $t_f \in \mathbb{N}$ tel que après t_f tours complets de boucle exécutés, le test $k < N$ est faux en ligne 2, ou de manière équivalente $k \geq N$ est vrai en ligne 2.

- Si $t_f = 0$ alors puisque N est un entier naturel supérieur ou égal à 1, $k \geq N$ et $k = 1$, on a $N = 1$ et donc $k = N$ en ligne 2 après t_f tours complets de boucle exécutés.
- Sinon, au début du $t_f{}^{\text{ème}}$ tour de boucle, on a $k < N$ en ligne 2. Or, après le tour de boucle, on a $k \geq N$ en ligne ligne 2. Enfin, la valeur de k est incrémenté de 1 à chaque tour de boucle. D'où, $k = N$ en ligne 2 après t_f tours complets de boucle exécutés.

Ainsi, dans tous les cas, on a $k = N$ en ligne 2 après t_f tours complets de boucle exécutés.

De $PBR(t_f)$ et $k = N$, on déduit que P_5 est vrai à la fin de l'algorithme, ce qui donne le bon résultat de l'algorithme. □

2.6 Question (f) (complexité)

On pose le prédicat suivant :

$\forall t \in \mathbb{N}$, $PNT(t)$: (($t = N - k$ au moment d'exécuter la ligne 2) $\Rightarrow$ (la boucle « tant que » termine après t tours)).

Montrons par récurrrence que $PNT(t)$ est vérifié pour tout $t \in \mathbb{N}$.

Initialisation : Si $t = 0$ au moment d'exécuter la ligne 2, on a $N - k = 0$. Donc, $k \geq N$ (précisément $k = N$) et on entre pas dans la boucle. Ainsi, la boucle « tant que » termine après 0 tour : $PNT(0)$ est vrai.

Hérédité : Il faut montrer que $\forall t \in \mathbb{N}$, $(PNT(t) \vDash PNT(t+1))$. (hérédité simple)

On précise ce que signifie $PNT(t+1)$:

$PNT(t+1)$: (($t + 1 = N - k$ au moment d'exécuter la ligne 2) $\Rightarrow$ (la boucle « tant que » termine après $t + 1$ tours))

Si $t + 1 \neq N - k$ au moment d'exécuter la ligne 2, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $PNT(t + 1)$ est trivialement vérifié.

Nous étudions maintenant le cas où $t + 1 = N - k$ au moment d'exécuter la ligne 2. On pose α la valeur de k à ce moment là. Donc, $t+1 = N-\alpha$, ce qui implique $t = N-\alpha-1 = N-(\alpha+1)$. Puisque N est constant et qu'à la fin du tour de boucle, on a $k = \alpha+1$, nous déduisons que $t = N-k$ à la fin du tour de boucle mais aussi après le premier tour de boucle lorsqu'on atteint la ligne 2. Donc, par hypothèse de récurrence, la boucle « tant que » termine après t tours de boucle supplémentaires. Au total, lorsque $t + 1 = N - k$ au moment d'exécuter la ligne 2, la boucle « tant que » termine après $t + 1$ tours de boucle : $PNT(t + 1)$ est vrai. □

À part la boucle « tant que », toutes les instructions de l'algorithme prennent un temps constant, sauf IVM_REC qui est en $O(N)$. De plus, la première fois qu'on atteint la ligne 2, on a $N - k = N - 1$. Donc, $N - k = N - 1$ et $PNT(N - 1)$ impliquent que l'algorithme exécute $N - 1$ tour de boucles dont l'instruction la plus coûteuse est en $O(N)$. Ainsi, la complexité en temps de notre algorithme est en $O(N^2)$ pour un tableau de N cases. □

3 Exercice 3

Similaire à l'exercice 2 : il faut utiliser une fonction $IVM3(T, f)$ avec $f \in [1..N]$ qui renvoie un entier naturel min tel que $min \in [1..f] \wedge (\forall j \in [1..f], T[min] \leq T[j])$.

Si, pour un entier naturel $k \geq 1$, on a $\forall i \in [k + 1..N], \forall j \in [1..i], T[j] \geq T[i]$, il suffit d'échange les valeurs d'indices k et $IVM3(T, k)$ pour obtenir $\forall i \in [k..N], \forall j \in [1..i], T[j] \geq T[j]$.

- Comment initialise-t-on ? Pour $k = N$, on a $\forall i \in [k + 1..N], \forall j \in [1..i], T[j] \geq T[j]$ qui est trivialement vrai.
- Comment s'arrête-t-on ? Pour $k = 1$, $\forall i \in [2..N], \forall j \in [1..i], T[j] \geq T[j]$ implique que le tableau est trié.

L'algorithme, ses preuves de terminaison et bon résultat, ainsi que sa complexité sont similaires à l'exercice 2.

4 Exercice 4

Cf., fichier à part.

5 Exercice 5

5.1 Question (a)

Soit T un tableau d'entiers de N cases indicées de 1 à N. Soit Ti la configuration initiale T. Soit Tf la configuration finale de T. On a les deux post-conditions suivantes :

1. Tf est une permutation de Ti : $\forall v \in Ti, (|Ti|_v = |Tf|_v)$.
2. Tf est trié par ordre croissant : $\forall i \in [1..N-1], Tf[i] \leq Tf[i+1]$.

5.2 Question (b) (idée)

Soit $d \in [1..N]$. Nous allons utiliser la fonction *Echange* et la variante de IVM suivante : la fonction $IVM\_REC(T, d)$ renvoie un entier naturel min tel que $min \in [d..N] \wedge (\forall j \in [d..N], T[j] \geq T[min])$. IVM_REC s'exécute en $O(N)$ (*cf.* exercice 1).

Supposons que le tableau est trié jusqu'à la case $d-1$, c'est-à-dire, $\forall i \in [1..d-1], \forall j \in [i..N], T[j] \geq T[i]$. Il suffit d'échanger la valeur de la case d et la valeur minimale entre les cases d et N ($IVM\_REC(T, d)$), puis de continuer récursivement le traitement sur les cases de $d+1$ à N.

— Quand arrêter les appels récursifs ? Quand $d = N$.

— Comment on commence ? On appelle avec $d = 1$.

5.3 Question (c) (algorithme)

Algorithme 3 Procédure de tri, Tri.

Entrées-Sorties

T : tableau d'entiers indicés de 1 à N

Programme

1: $TriRec(T, 1)$ /* $P0 : d \in [1..N] \wedge (\forall i \in [1..0], \forall j \in [i..N], T[j] \geq T[i])$ */

Algorithme 4 Procédure de tri récursif, $TriRec$.

Entrées-Sorties

T : tableau d'entiers indicés de 1 à N

d : entier naturel non-nul /* $1 \leq d \leq N$ */

Variables

min : entier naturel non-nul

Programme

/* $H : d \in [1..N] \wedge (\forall i \in [1..d-1], \forall j \in [i..N], T[j] \geq T[i])$ */

1: **Si** $d < N$ **alors**

/* $P1 : d \in [1..N-1] \wedge (\forall i \in [1..d-1], \forall j \in [i..N], T[j] \geq T[i])$ */

2: $min \leftarrow IVM\_REC(T, d)$

/* $P2 : P1 \wedge min \in [d..N] \wedge (\forall j \in [d..N], T[j] \geq T[min])$ */

3: $Echange(T, min, d)$

/* $P3 : d+1 \in [2..N] \wedge (\forall i \in [1..d], \forall j \in [i..N], T[j] \geq T[i])$ */

4: $TriRec(T, d+1)$

/* $P4 : \forall i \in [1..N], \forall j \in [i..N], T[j] \geq T[i]$ */

5: **Fin Si**

5.4 Question (d) (terminaison)

Nous allons tout d'abord démontrer par récurrence que $\forall i \in \mathbb{N}^*, IndiceOk(i)$: s'il y a au moins i appels à $TriRec(T, d)$, alors le $i^{\text{ème}}$ appel à $TriRec(T, d)$ est effectué avec $d \in [1..N]$.

Initialisation : Lors du premier appel à $TriRec(T, d)$, on a $d = 1$, *cf.* ligne 1 de l'algorithme Tri. Ainsi, $IndiceOk(1)$ est trivialement vérifiée.

Hérédité : Il faut montrer que $\forall i \in \mathbb{N}^*$, $(IndiceOk(i) \vDash IndiceOk(i+1))$. (hérédité simple)

On précise ce que signifie $IndiceOk(i+1)$:

s'il y a au moins $i+1$ appels à $TriRec(T, d)$, alors le $i^{\text{ème}}$ appel à $TriRec(T, d)$ est effectué avec $d \in [1..N]$.

S'il y a moins de $i+1$ appels à $TriRec(T, d)$, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $IndiceOk(i+1)$ est trivialement vérifié.

Sinon, il y a au moins $i+1$ appels à $TriRec(T, d)$ et lors du $i^{\text{ème}}$ appel à $TriRec$, on a $d = \alpha$ avec $\alpha \in [1..N]$, par hypothèse de récurrence. Or, pour qu'il y ait un appel supplémentaire, il faut que α soit inférieur à N (*cf.* ligne 1). Donc, on a $\alpha \in [1..N-1]$. De plus, la variable d n'est pas modifié jusqu'au $i+1^{\text{ème}}$ appel à $TriRec(T, d)$ qui est donc effectué avec $d = \alpha + 1 \in [2..N]$. D'où, $IndiceOk(i+1)$ est vérifié.

□

D'après le résultat précédent, lors d'un appel à $TriRec(T, d)$, on a $d \in [1..N]$. Ensuite, d n'est pas modifié lors de l'appel et lorsqu'on appelle IVM_REC, on a $d < N$. Donc, IVM_REC est appelé avec un indice du tableau et ainsi renvoie un indice du tableau stocké dans min. En conséquence, $Echange$ est aussi appelé avec des indices du tableaux (min et d). D'où :

Remarque 2. *Lorsque les fonctions IVM_REC ou Echange sont utilisées, leurs pré-conditions sont vérifiées.*

Montrons maintenant qu'un appel $TriRec(T, d)$ avec $d \in [1..N]$ engendre un nombre fini d'appels récursifs. Pour cela, on pose le prédicat suivant :

$\forall k \in \mathbb{N}$, $PA(k)$: (($k = N - d$ au moment de l'appel à $TriRec(T, d)$) $\Rightarrow$ (le nombre d'appels récursifs engendrés par $TriRec(T, d)$ est fini)).

Montrons par récurrrence que $PA(k)$ est vérifié pour tout $k \in \mathbb{N}$.

Initialisation : Si $k = 0$ au moment de l'appel, on a $N - d = 0$. Donc, $d = N$ et il n'y a plus d'appel récursif. Ainsi, $PA(0)$ est vérifié.

Hérédité : Il faut montrer que $\forall k \in \mathbb{N}$, $(PA(k) \vDash PA(k+1))$. (hérédité simple)

On précise ce que signifie $PA(k+1)$:

$PA(k+1)$: (($k + 1 = N - d$ au moment de l'appel à $TriRec(T, d)$) $\Rightarrow$ (le nombre d'appels récursifs engendrés par $TriRec(T, d)$ est fini))

Si $k + 1 \neq N - d$ au moment de l'appel, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $PA(k+1)$ est trivialement vérifié.

Nous étudions maintenant le cas où $k + 1 = N - d$ au moment de l'appel à $TriRec(T, d)$. Lors de cet appel, on a $d < N$ (le fait que k et N soient des entiers naturels, nous permet de déduire $d < N$ de $N = d + k + 1$) on effectue un seul appel récursif : $TriRec(T, d+1)$. Or, de $k + 1 = N - d$, on déduit que $k = N - (d + 1)$. Donc, par hypothèse de récurrence, $TriRec(T, d+1)$ n'engendrera qu'un nombre fini d'appels récursifs. Puisque $TriRec(T, d)$ n'engendre qu'un seul appel de plus que $TriRec(T, d+1)$, on déduit que $TriRec(T, d)$ n'engendre qu'un nombre fini d'appels récursifs : $PA(k+1)$ est vrai. □

À part l'appel récursif $TriRec(T, d+1)$ lorsque $d < N$, toutes les instructions de $TriRec(T, d)$ sont soit élémentaires soit des appels de fonctions qui terminent par hypothèse. Or, puisque $PA(t)$ est vrai pour tout $t \in \mathbb{N}$, le nombre d'appels récursifs engendrés $TriRec(T, d)$ lorsque $d \in [1..N]$ est fini. Enfin, Tri consiste uniquement en un appel à $TriRec(T, 1)$. Donc, la procédure Tri termine.

□

5.5 Question (e) (bon résultat)

Tout d'abord, les actions sur le tableau sont uniquement des lectures et des échanges. Or, on a déjà pprouvé que lors des échanges, les paramètres étaient des indices de tableau. Donc, à la fin de l'exécution de la procédure Tri, T est bien une permutation de son état initial.

Nous allons maintenant démontrer qu'à la de la procédure Tri, T est trié dans l'ordre croissant. Soit

$$PBR(k) \equiv \forall i \in [1..k], \forall j \in [i..N], T[j] \geq T[i]$$

Nous allons démontrer par récurrence que $\forall t \in \mathbb{N}$, $P(t) : t = N - d \wedge PBR(d-1)$ au moment de l'appel à $TriRec(T,d) \Rightarrow PBR(N-1)$ est vrai juste après l'appel.

Initialisation : Lorsque $t = 0$, on a $d = N$.

Si $\neg PBR(d-1)$, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $P(0)$ est trivialement vérifié.

Sinon, par hypothèse $PBR(d-1)$ est vrai avec $d = N$, donc $PBR(N-1)$ est vrai au moment de l'appel et puisque le test en ligne 1 est faux, T n'est pas modifié et ainsi $PBR(N-1)$ reste vrai après l'appel. D'où, $P(0)$ est vérifié.

Hérédité : Il faut montrer que $\forall t \in \mathbb{N}$, $(P(t) \vDash P(t+1))$. (hérédité simple)

On précise ce que signifie $P(t+1)$:

$P(t+1) : t+1 = N - d \wedge PBR(d-1)$ au moment de l'appel à $TriRec(T,d) \Rightarrow PBR(N-1)$ est vrai juste après l'appel.

Si $t+1 \neq N - d \vee \neg PBR(d-1)$ au moment de l'appel à $TriRec(T,d)$, alors la prémisse de l'implication est fausse, donc l'implication est vraie et $P(t+1)$ est trivialement vérifié.

Nous étudions maintenant le cas où $t+1 = N-d \wedge PBR(d-1)$ au moment de l'appel à $TriRec(T,d)$. On a $t = N - (d+1)$ et si nous avons $PBR((d+1)-1) = PBR(d)$ au moment de l'appel à $TriRec(T, d+1)$, alors, par hypothèse de récurrence, $PBR(N-1)$ sera vrai juste après l'appel et ainsi $P(t+1)$ sera vérifié.

Puisque $d \in [1..N]$ (prouvé précédemment), N est un entier naturel et $N - d = t + 1 \in \mathbb{N}^*$, on a $d < N$ en ligne 1. Ensuite, après l'affectation en ligne 2, on a $min \in [d..N] \wedge (\forall j \in [d..N], T[j] \geq T[min])$ puisque IVM_REC est appelé avec T et d, ce dernier étant un indice valide. De même, puisque $Echange$ est appelé en ligne 3 avec T, min et d, ces deux derniers étant des indices valides, on a $\forall i \in [1..d], \forall j \in [i..N], T[j] \geq T[i]$, c'est-à-dire $PBR(d)$, lors de l'appel $TriRec(T, d+1)$ en ligne 4. D'où, $PBR(N-1)$ est vrai juste après cet appel et donc à la fin de l'appel $TriRec(T,d)$. Ainsi, $P(t+1)$ est vérifié.

T est trivialement trié (dans l'ordre croissant) si $N = 0$. Supposons maintenant que $N \in \mathbb{N}^*$. La procédure Tri consiste uniquement à exécuter $TriRec(T,1)$ et $t = N - 1 \in \mathbb{N}$ au moment de cet appel. De plus, $PBR(0)$ est trivialement vrai. Donc, puisque $\forall t \in \mathbb{N}$, $P(t)$, on a $PBR(N-1)$ qui est vrai juste après l'appel, ce qui implique que T est trié dans l'ordre croissant à la fin de la procédure Tri.

5.6 Question (f) (complexité)

Posons $k = N - d$. Soit $C(k)$ la fonction qui permet de calculer la complexité d'un appel de $TriRec(T,d)$. Tout d'abord, $t = N - d = 0$ lorsque $d = N$. Donc, $C(0) = 1$. Ensuite, récursivement, on a :
$C(0) = 1$
$C(1) = 4 + O(N) + C(0) = 5 + O(N)$
$C(2) = 4 + O(N) + C(1) = 9 + 2 \times O(N)$
...
D'où, $C(k) = 4 + O(N) + C(k-1) = 4 \times k + 1 + k \times O(N) = O(k.N)$.

Démontrons, par récurrence que $\forall k \in \mathbb{N}$, $C(k) = 4 \times k + 1 + k \times O(N)$.

Initialisation : Pour $k = 0$, on a $4 \times k + 1 + k \times O(N) = 4 \times 0 + 1 + 0 \times O(N) = 1 = C(0)$.

Hérédité : Il faut montrer que $\forall k \in \mathbb{N}$, $((C(k) = 4 \times k + 1 + k \times O(N)) \vDash (C(k+1) = 4 \times (k+1) + 1 + (k+1) \times O(N)))$. (hérédité simple)

$C(k+1) = 4 + O(N) + C(k)$. Or, par hypothèse de récurrence, $C(k) = 4 \times k + 1 + k \times O(N)$. Donc, $C(k+1) = 4 + O(N) + 4 \times k + 1 + k \times O(N) = 4 \times (k+1) + 1 + (k+1) \times O(N)$. □

$Tri(T)$ consiste en un appel à $TriRec(T, 1)$. Donc, le coût de $Tri(T)$ est $1 + C(N-1) = 1 + 4 \times (N-1) + 1 + (N-1) \times O(N) = O(N^2)$.

6 Exercice 6

Le principe est identique à celui de l'exercice 3.