Mémo n. 6

Passeport pour la suite ...

I32 Preuves et Analyses d'algorithmes

P. Véron

Un certain nombre des notions présentées resteront certainement encore assez floues à la fin de ce module. Cependant ceux d'entre vous qui continueront dans la "voie informatique" reverront une nouvelle fois de façon plus approfondie certains des sujets traités. En tout état cause, il est indispensable en cette fin de premier semestre, après avoir suivi (ou subi !) les 3 modules I11, I21, I31, et I32 que vous soyez capables d'écrire les yeux fermés les trois algoritmes suivants : somme d'une suite de n entiers, recherche d'un élément dans une suite, recherche du plus petit élément d'une suite. Vous trouverez ci-dessous ces trois algorithmes avec en prime leurs preuves et l'analyse de leurs temps d'exécution.

Somme des éléments d'un tableau

		Coût	fois
1	Algo somme		
2	**données**		
3	A : *tableau de n entiers*		
4	s, i : *entiers*		
5	**début**		
6	*lire*(A)	c_1	1
7	$s \leftarrow 0$	c_2	1
8	$i \leftarrow 1$	c_3	1
9	$\{s = \sum_{j=1}^{i-1} A[j]$ *et* $0 \leqslant i \leqslant n+1\}$		
10	**tant que** $i \leqslant n$ **faire**	c_4	$n+1$
11	$\{s = \sum_{j=1}^{i-1} A[j]$ *et* $0 \leqslant i \leqslant n\} \Rightarrow$		
12	$\{s + A[i] = \sum_{j=1}^{i} A[j]$ *et* $0 \leqslant i \leqslant n\}$	c_5	n
13	$s \leftarrow s + A[i]$		
14	$\{s = \sum_{j=1}^{i} A[j]$ *et* $0 \leqslant i \leqslant n\} \Rightarrow$		
15	$\{s = \sum_{j=1}^{i+1-1} A[j]$ *et* $0 \leqslant i+1 \leqslant n+1\}$	c_6	n
16	$i \leftarrow i + 1$		
17	$\{s = \sum_{j=1}^{i-1} A[j]$ *et* $0 \leqslant i \leqslant n+1\}$		
18	**fintq**		
19	$\{s = \sum_{j=1}^{i-1} A[j]$ *et* $0 \leqslant i \leqslant n+1$ *et* $i \geqslant n+1\} \Rightarrow$		
20	$\{i = n+1$ *et* $s = \sum_{j=1}^{n} A[j]\}$		
21	*ecrire*(s)	c_7	1
22	**fin**		

La preuve d'arrêt de cet algorithme se fait en montrant que $\{n+1-i \geqslant 0\}$ est un invariant de boucle et que le schéma suivant est correct : $\{n+1-i = n_0\}$ $s \leftarrow s + A[i]$, $i \leftarrow i+1$ $\{n+1-i < n_0\}$. On a $T(n) = c_1 + c_2 + c_3 + c_4 + c_7 + (c_4 + c_5 + c_6)n = \Theta(n)$.

Recherche d'un élément

		Coût	fois
1	Algo recherche		
2	**données**		
3	A : *tableau de n entiers*		
4	x, i : *entiers*		
5	*oncontinue* : *booléen*		
6	**début**	c_1	1
7	*lire(A)*	c_2	1
8	*lire(x)*	c_3	1
9	*oncontinue* $\leftarrow (x \neq A[1])$	c_4	1
10	$i \leftarrow 2$		
11	$\{oncontinue \Leftrightarrow (\forall j,\ 1 \leqslant j < i,\ A[j] \neq x)\}$ *et* $\{0 \leqslant i \leqslant n+1\}$		
12	**tant que** $i \leqslant n$ *et oncontinue* **faire**	c_5	t
14	$\{\forall j,\ 1 \leqslant j < i,\ A[j] \neq x\}$ *et* $\{0 \leqslant i \leqslant n\} \Rightarrow$		
15	$\{(x \neq A[i]) \Leftrightarrow (\forall j,\ 1 \leqslant j \leqslant i,\ A[j] \neq x)\}$ *et* $\{0 \leqslant i \leqslant n\}$		
16	*oncontinue* $\leftarrow (x \neq A[i])$	c_6	$t-1$
17	$\{oncontinue \Leftrightarrow (\forall j,\ 1 \leqslant j \leqslant i,\ A[j] \neq x)\}$ *et* $\{0 \leqslant i \leqslant n\} \Rightarrow$		
18	$\{oncontinue \Leftrightarrow (\forall j,\ 1 \leqslant j < i+1,\ A[j] \neq x)\}$ *et* $\{0 \leqslant i+1 \leqslant n+1\}$		
19	$i \leftarrow i+1$	c_7	$t-1$
20	$\{oncontinue \Leftrightarrow (\forall j,\ 1 \leqslant j < i,\ A[j] \neq x)\}$ *et* $\{0 \leqslant i \leqslant n+1\}$		
21	**fintq**		
22	**si** *non*(*oncontinue*) **alors**	c_8	1
23	*ecrire("x appartient au tableau")*	c_9	0 ou 1
24	**sinon**		
25	*ecrire("x n'appartient au tableau")*	c_{10}	0 ou 1
26	**finsi**		
27	**fin**		

D'après l'invariant de boucle, étant donné qu'à la sortie de la boucle on a $i \leqslant n+1$, si le booléen *oncontinue* est faux ceci signifie qu'il existe un entier j, $1 \leqslant j \leqslant n$ tel que $A[j] = x$. Ceci explique pourquoi on teste la valeur de ce booléen à la sortie de la boucle. La preuve d'arrêt est identique à celle de l'algorithme *somme*, en ce qui concerne l'invariant à utiliser. On a

$$c_1+c_2+c_3+c_4-c_6-c_7+c_8+(c_5+c_6+c_7)t \leqslant T(n) \leqslant c_1+c_2+c_3+c_4-c_6-c_7+c_8+c_9+c_{10}+(c_5+c_6+c_7)t$$

où t est le nombre de fois où le test de la boucle est exécuté. Dans une telle situation nous devons analyser deux cas distincts :

Cas favorable : Le booléen *oncontinue* est faux dès son initialisation (i.e $x = A[1]$), on effectue alors le test de la boucle qu'une seule fois, on a alors $t = 1$. D'où $\exists \alpha > 0$, tel que $T_{\min}(n) \geqslant \alpha$ pour toute valeur de n. D'où $\exists A(= \alpha) > 0$, $\exists n_0 = 0$ tels que $\forall n \geqslant n_0$, $0 \leqslant A \leqslant T_{\min}(n) \leqslant T(n)$, donc $T(n) = \Omega(1)$.

Cas défavorable : Le booléen *oncontinue* est vrai pour toute valeur de i (i.e l'élément x ne fait pas partie du tableau), on effectue alors le test de la boucle n fois, i.e. $t = n$. D'où $\exists \alpha' > 0$, $\beta' \geqslant 0$ tels que $T_{\max}(n) \leqslant \alpha' n + \beta'$. Or $\forall n \geqslant 1$, $\alpha' n + \beta' \leqslant (\alpha' + \beta')n$. Donc $\exists B(= \alpha' + \beta') > 0$, $\exists n_0 = 1$ tels que $\forall n \geqslant n_0$, $T_{\max}(n) \leqslant Bn$. Or $T(n) \leqslant T_{\max}(n)$, d'où $T(n) = \mathcal{O}(n)$.

Ainsi on a $T(n) = \Omega(1)$ et $T(n) = \mathcal{O}(n)$.

Recherche du plus petit élément dans un tableau

		Coût	fois
1	Algo minimum		
2	**données**		
3	A : *tableau de n entiers*		
4	min, i : *entiers*		
5	**début**		
6	*lire(A)*	c_1	1
7	$min \leftarrow A[1]$	c_2	1
8	$i \leftarrow 2$	c_3	1
9	**tant que** $i \leqslant n$ **faire**	c_4	$n+1$
10	**si** $A[i] < min$ **alors**	c_5	n
11	$min \leftarrow A[i]$	c_6	t
12	**finsi**		
13	$i \leftarrow i+1$	c_7	1
14	**fintq**		
15	*ecrire(min)*	c_8	1
16	**fin**		

La preuve de ce programme se fait en démontrant que l'assertion

$$\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n+1\}$$

est un invariant de boucle. Cette assertion est clairement vérifiée avant la boucle puisque $i = 2$ (on suppose que l'on travaille sur un tableau possédant au moins un élément !). En rentrant dans la boucle on a donc l'assertion

$$\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n+1 \text{ et } i \leqslant n\}$$

soit

$$\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n\}.$$

Pour démontrer que l'assertion initiale est un invariant de boucle, il nous reste donc à démontrer que

$\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n\}$
Si
$\vdots$
finsi
$i \leftarrow i+1$
$\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n+1\}$.

L'instruction étant composée d'un choix simple, il nous faut donc montrer tout d'abord (cf. memo 2) :

a) $\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n\}$ et $\{A[i] < min\}$ $min \leftarrow A[i]$ $\{min = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\}$,

b) $\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n\}$ et $\{A[i] \geqslant min\} \Rightarrow \{min = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\}$.

Le point b) est évident, reste à prouver le point a). On a

$$\{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n\} \text{ et } \{A[i] < min\} \;\Rightarrow\; \{A[i] = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\}$$

D'où

$$\{A[i] = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\}\ min \leftarrow A[i]\ \{min = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\}$$

Après l'instruction **finsi** on a donc

$$\{min = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\}$$

or

$$\{min = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\} \Rightarrow \{min = \min_{1 \leqslant j \leqslant i+1-1} A[j] \text{ et } 0 \leqslant i+1 \leqslant n+1\}$$

et

$$\{min = \min_{1 \leqslant j \leqslant i+1-1} A[j] \text{ et } 0 \leqslant i+1 \leqslant n+1\}\ i \leftarrow i+1\ \{min = \min_{1 \leqslant j \leqslant i-1} A[j] \text{ et } 0 \leqslant i \leqslant n+1\}$$

L'assertion est donc bien un invariant et à la sortie de la boucle on obtient donc

$$\{min = \min_{1 \leqslant j \leqslant i} A[j] \text{ et } 0 \leqslant i \leqslant n\} \text{ et } \{i > n\}$$

soit

$$\{min = \min_{1 \leqslant j \leqslant n} A[j]\}$$

La preuve d'arrêt est identique à celles des deux algorithmes précédents, la condition `Si` ... **finsi** ne modifiant pas la valeur de i.

On a $T(n) = c_1 + c_2 + c_3 + c_4 + c_7 + c_8 + (c_4 + c_5)n + c_6 t$. Il nous faut donc discuter des cas favorables et défavorables afin de se débarasser de la variable t qui représente le nombre de fois où est exécutée l'instruction correspondante dans l'algorithme.

Cas favorable : L'instruction n'est jamais exécutée, i.e le test est toujours faux (ce qui signifie que le plus petit élément du tableau se trouve en première position). On a donc $t = 0$ et $T_{\max}(n) = \alpha + \beta n \leqslant (\alpha + \beta)n$ pour $n \geqslant 1$. D'où $\exists B(= \alpha + \beta) > 0$, $\exists n_0 = 1$ tels que $\forall n \geqslant n_0$, $0 \leqslant T(n) \leqslant T_{\max}(n) \leqslant Bn$. D'où $T(n) = \mathcal{O}(n)$.

Cas défavorable : L'instruction est exécutée à chaque passage dans la boucle, i.e. le test est toujours vrai (ce qui signifie ici que le tableau est forcément rangé dans l'ordre décroissant). On a donc $t = n$ et $T_{\min}(n) = \alpha + \beta' n \geqslant \beta' n \geqslant 0$. Donc $\exists A(= \beta') > 0$, $\exists n_0 = 0$ tels que $\forall n \geqslant n_0\ T(n) \geqslant T_{\min}(n) \geqslant An \geqslant 0$. D'où $T(n) = \Omega(n)$.

On a donc $T(n) = \Omega(n)$ et $T(n) = \mathcal{O}(n) \Leftrightarrow T(n) = \Theta(n)$.