Suite du TD1

Algo TD2 le 9 février 2021

6.1.2

$u_n = 2u_{n-1} + 1 \qquad u_0 = 2$

$= 2(2u_{n-2} + 1) + 1$

$= 2(2(2u_{n-3} + 1) + 1) + 1$

$= 2^3 u_{n-3} + \underbrace{1 + 2 + 4}_{2^3 - 1}$

$= \dots$

$= 2^n \times u_0 + 2^n - 1$

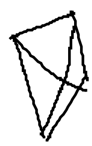
$= 2 \times 2^n + 2^n - 1$

$= 3 \cdot 2^n - 1$

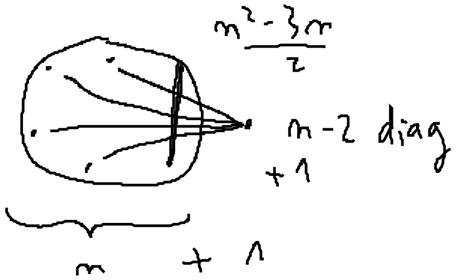
Exo 9

Diagonales d'un polygone à n côtés : $d(n) = \frac{n^2 - 3n}{2}$

Base : un quadrilatère a 2 diagonales et $\frac{n^2 - 3n}{2} = \frac{16 - 12}{2} = 2$.



Induction :



$d(n+1) = \frac{n^2 - 3n}{2} + n - 2 + 1$

$= \frac{1}{2}(n^2 - 3n + 2n - 2)$

$= \frac{1}{2}(\underbrace{n^2 + 2n + 1} - 3n - 3)$

$= \frac{1}{2}\left[(n+1)^2 - 3(n+1)\right]$

Exo 14

(Q1) $u_n = \sum_{i=0}^{n-1} u_i$, $u_0 = 1$

2) Mq $u_n = 2^{n-1}$ $\forall n \in \mathbb{N}^*$

Base : $u_1 = \sum_0^0 u_i = u_0 = 1 = 2^0$

Induction : $u_{n+1} = \sum_{i=0}^{n+1-1} u_i = \underbrace{\sum_{i=0}^{n-1} u_i}_{=u_n} + u_n = 2u_n \overset{hyp}{=} 2 \times 2^{n-1} = 2^n$

Récurrence faible car on se base sur u_n seulement.

1) $u_{n+1} = \sum_{i=0}^{n+1-1} u_i = \underbrace{u_0}_{=1} + \underbrace{\sum_{i=1}^{n} 2^{i-1}}_{2^n - 1} = 2^n$

Récurrence forte car on se base sur tous les u_i, $i \leq n$

Notations de Landau

APCR = à partir d'un certain rang

"grand O" : $f \in O(g) \Leftrightarrow \exists D \in \mathbb{R}^{+*}_{>0}$, $f \leq D \cdot g$ APCR

$\exists x_0 \in \mathbb{R}, \forall x \geq x_0, f(x) \leq D \cdot g(x)$

oméga Ω : $f \in \Omega(g) \Leftrightarrow \exists C \in \mathbb{R}_+^*$, $f \geq C g$ APCR

théta Θ : $f \in \Theta(g) = O(g) \cap \Omega(g)$

$\Leftrightarrow \exists D, C \in \mathbb{R}_+^*$, $C \cdot g \leq f \leq D \cdot g$ apcr.

("petit o" : négligeable $f \in o(g) \Leftrightarrow \frac{f}{g} \xrightarrow[+\infty]{} 0$ ex : $f(x) = x^7$, $g(x) = e^x$ $\frac{f}{g} = \frac{x^7}{e^x} \longrightarrow 0$)

Exo 17

Q1) i) Mq $n^2 \in O(10^{-5} n^3)$

méthode 1 : $\lim_{n \to +\infty} \frac{n^2}{10^{-5} n^3} = 0$ donc $n^2 \in o(10^{-5} n^3) \subset O(10^{-5} n^3)$

méthode 2 : On a $n^2 \leq 10^5 \times 10^{-5} n^3$ $\forall n \in \mathbb{N}$

ii) Mq $\underbrace{25n^4 - 19n^3 + 13n^2}_{f(n)} \in O(\underbrace{n^4}_{g(n)})$

- $\frac{f}{g} \xrightarrow[+\infty]{} 25$ donc d'après le cours, $f \in \Theta(g) \subset O(g)$

- $f(n) = 25n^4 - 19n^3 + 13n^2 \leq (25+13)n^4 = \underbrace{38}_{D}\underbrace{n^4}_{g(n)}$

iii) Mq $2^{n+100} \in O(2^n)$

- $f(n) = 2^{n+100} = 2^{100} \cdot 2^n \leq 2^{100} \cdot g(n)$ donc $f \in O(g)$
(et même $f \in \Theta(g)$)

Q2

$O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(2^n)$
$\subset O(n^3)$

Exo 18

1) Mq $f+g \in \Theta(\max(f,g))$, $f,g : \mathbb{N} \to \mathbb{N}$ ou $\mathbb{R}^+$

$1 \times \max(f,g) \leq f+g \leq \max(f,g) + \max(f,g) = 2 \times \max(f,g)$

par définition, $f+g \in \Theta(\max(f,g))$

2) • soit $h \in O(f+g)$, Mq $h \in O(\max(f,g))$

- $\exists D \in \mathbb{R}^+$, $h \leq D \times (f+g)$ APCR
$\leq \underbrace{D \times 2}_{D'} \times \max(f,g)$ (question 1) donc $h \in O(\max)$

- soit $h \in O(\max(f,g))$
$\exists D \in \mathbb{R}^+$, $h \leq D \times \max(f,g)$ APCR
$\leq D \times (f+g)$ donc $h \in O(f+g)$

- Donc $O(f+g) = O(\max(f,g))$

Exo 19

Q1) 1) $u_n = u_{n-1} + n$, $u_0 = 0$

On trouve $u_n = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$ donc $u_n \in \Theta(n^2)$

or $\frac{\frac{n^2+n}{2}}{n^2} \xrightarrow[+\infty]{} \boxed{\frac{1}{2}}$

2) $u_n = 2u_{n-1} + 1$, $u_0 = 2$. $u_n = 3 \cdot 2^n - 1$ $\forall n \in \mathbb{N}^*$

$u_n = 3e^{n\ln 2} - 1$

$\lim_{n\to\infty} \frac{u_n}{n} = +\infty$: $u_n \in \Omega(n)$

$n \in o(u_n)$

$\lim_{n\to+\infty} \frac{u_n}{2^n} = \boxed{3}$ donc $u_n \in \Theta(2^n)$

$2\times 2^n \leq u_n \leq 4 \times 2^n$ APCR

TD 2

Problème : ex "trier"

Instance : "trier [2,4,1,3,0]"

Variables, opérations, contrôles

Terminaison : ça s'arrête

Validité : réponse correcte

Complexité : Θ(temps de calcul)

Invariant de boucle : vraie à chaque fin de boucle

Variant de boucle : valeur qui diminue à chaque boucle

Exo 1 (magenta) : Fonction factorielle

1) Terminaison : boucle exécutée n fois, une seule instruction

2) Soit tmp_i la valeur de tmp à la fin de chaque boucle

Mq $tmp_i = (i-1)!$

Base : $tmp_1 = 0! = 1$

Induction : Supposons que $tmp_i = (i-1)!$

Alors, après un tour de boucle en plus, $tmp_{i+1} = tmp_i \times i = (i-1)! \times i$

$= i! = ((i+1)-1)!$

3) On sort quand $i = n+1$, donc $tmp_{n+1} = n!$ est la valeur finale

Exo 2 (bleu) : Somme des éléments d'un tableau $n=4$

1) Exécutée n fois, avec 1 instruction : $i=0$, $i=1$, $\ldots$, $i=n-1$ (0, 1, 2, 3)

2) Mq $tmp_i = \sum_{j=0}^{i-1} tab[j]$ Base : $i=0$, $tmp_i = 0$ ok.

Induction : Supposons $tmp_i = \sum_{j=0}^{i-1} tab[j]$

Alors $tmp_{i+1} = tmp_i + tab[i] = \sum_{j=0}^{i-1} tab[j] + tab[i] = \boxed{\sum_{j=0}^{i} tab[j]}$

3) On sort de la boucle quand $i = nbElem = n$

donc $tmp_i = tmp_n = \sum_{j=0}^{n-1} tab[j]$ qui est la somme de tous les éléments de tab.

Remarque :
- la somme de rien vaut 0 car le neutre de l'addition est zéro. Ex: somme de (tab[j] pour j de 0 = -1) = 0
- le produit de rien vaut 1 car le neutre du produit est un. Ex: 0! = 1

Exo 3 (rouge)

1) Terminaison : on avance dans un tableau à n éléments donc il y a au plus n boucles de 1 opération.

2) Invariant : $\Pi(i)$: " $\forall j \in [0, i-1]$, $tab[j] \neq elem$ "

Base : $\Pi(0)$, tableau vide donc ok.

Induction : Supposons $\Pi(i)$, montrons $\Pi(i+1)$

On sait déjà que $tab[j] \neq elem$ $\forall j \leq i-1$

De plus, par condition de boucle, $tab[i] \neq elem$. D'où $\Pi(i+1)$

3) On sort quand $tab[i]$ vaut elem, et d'après $\Pi(i)$, tous les éléments précédents sont $\neq$ elem

Donc i est bien le plus petit indice.

Exercice 4 : même exercice sauf que le tableau est trié et ne contient peut-être pas elem

1) Il y a au plus n boucles (parcourir le tableau)

2) $\Pi(i)$: " $\forall j \in [\![0, i-1]\!]$, $tab[j] < elem$ "

Base : tableau vide, ok.

Induction : Supposons $\Pi(i)$, si $tab[i] \geqslant elem$, on sort
autrement $tab[i] < elem$, et on sait que $tab[j] < elem$ $\forall j \leqslant i-1$
donc $\Pi(i+1)$

3) On sort quand $tab[i] \geqslant elem$ et $\Pi(i)$ vérifiée

Condition : si $i < n$, et $tab[i] = elem$,
alors elem est dans le tableau à l'indice i

Sinon : on renvoie -1

Méthode pour les algorithmes itératifs :

1) Montrer que le nombre de boucles est fini (Terminaison)
2) Montrer qu'un invariant de boucle est conservé tout au long du calcul
3) Montrer que la valeur de l'invariant quand on sort de la boucle est le résultat attendu (Validité)