Algorithm II

Normal

2023-2024

Exercice 1 : Fonction récursive `sommeChiffres`

La fonction récursive `sommeChiffres` calcule la somme des chiffres d'un entier `n`. Le cas de base est un nombre à un seul chiffre. L'étape récursive consiste à additionner le dernier chiffre (`n mod 10`) avec la somme des chiffres du reste du nombre (`n div 10`).

```
Fonction sommeChiffres(n: entier): entier
Debut
    Si n < 10 alors
        Retourner n
    Sinon
        Retourner (n mod 10) + sommeChiffres(n div 10)
    FinSi
Fin
```

Exercice 2.1 : Fonction itérative `premier`

Cette fonction détermine si un entier positif `n` est un nombre premier. Un nombre est premier s'il est supérieur à 1 et n'a pas d'autres diviseurs que 1 et lui-même. On teste la divisibilité de 2 jusqu'à `n-1`.

```
Fonction premier(n: entier): booléen
Variables
    i: entier
Debut
    Si n <= 1 alors
        Retourner Faux
    FinSi

    Pour i de 2 a (n - 1) Faire
        Si (n mod i = 0) alors
            Retourner Faux
        FinSi
    FinPour

    Retourner Vrai
Fin
```

Exercice 2.2 : Fonction `premierTableau`

Cette fonction compte le nombre de nombres premiers dans un tableau d'entiers en utilisant la fonction `premier` précédemment définie.

```
Fonction premierTableau(T: tableau[] d'entiers, N: entier): entier
Variables
    i, compteur: entier
Debut
    compteur <-- 0
    Pour i de 1 a N Faire
        Si (premier(T[i]) = Vrai) alors
            compteur <-- compteur + 1
        FinSi
    FinPour
    Retourner compteur
Fin
```

Exercice 2 : Algorithme `FicheEleve`

Voici l'algorithme complété. Il saisit les informations d'un certain nombre d'élèves, puis trouve et affiche l'élève ayant la moyenne la plus élevée.

```
Algorithme FicheEleve

Type eleve = enregistrement
    nom: chaine de caracteres
    classe: chaine de caracteres
    moyenne: reel
Fin

Procedure saisirEleve(var E: eleve)
Debut
    Ecrire("Nom:")
    Lire(E.nom)
    Ecrire("Classe:")
    Lire(E.classe)
    Ecrire("Moyenne:")
    Lire(E.moyenne)
Fin

Variables
    T: Tableau [1..20] de eleve
    n, i, iMax: entier
    moyMax: reel

Debut
```

```
    Ecrire("Donner le nombre des eleves (entre 2 et 20):")
    Repeter
        Lire(n)
    Jusqu'a (n >= 2 ET n <= 20)

    // Remplir le tableau T
    Pour i de 1 a N faire
        saisirEleve(T[i])
    FinPour

    moyMax <-- T[1].moyenne
    iMax <-- 1
    Pour i de 2 a N faire
        Si (T[i].moyenne > moyMax) alors
            moyMax <-- T[i].moyenne
            iMax <-- i
        FinSi
    FinPour

    // Afficher les resultats
    Ecrire("L'eleve avec la moyenne maximale est :")
    Ecrire("Nom: ", T[iMax].nom)
    Ecrire("Classe: ", T[iMax].classe)
    Ecrire("Moyenne: ", moyMax)
Fin
```

Exercice 3 : Analyse Asymptotique et Complexité

1. Montrer que $f(n) = \Theta(g(n))$

On suppose que $\lim_{n\to\infty} \frac{f(n)}{g(n)} = l$ avec $l > 0$. Par définition de la limite, pour tout $\varepsilon > 0$, il existe un entier n_0 tel que pour tout $n \geq n_0$, on a : $|\frac{f(n)}{g(n)} - l| < \varepsilon$

Ceci est équivalent à : $l - \varepsilon < \frac{f(n)}{g(n)} < l + \varepsilon$

En utilisant l'indication et en choisissant $\varepsilon = \frac{l}{2}$ (possible car $l > 0$), on obtient : $l - \frac{l}{2} < \frac{f(n)}{g(n)} < l + \frac{l}{2}$ $\frac{l}{2} < \frac{f(n)}{g(n)} < \frac{3l}{2}$

En posant $c_1 = \frac{l}{2}$ et $c_2 = \frac{3l}{2}$, on a deux constantes positives. En supposant $g(n) > 0$, on obtient : $c_1 \cdot g(n) < f(n) < c_2 \cdot g(n)$

Ceci est la définition de $f(n) = \Theta(g(n))$.

2. Temps d'exécution $T(n)$ et complexité de `Algo`

L'opération élémentaire est `Ecrire(j)`. On compte le nombre de fois qu'elle est exécutée. La boucle externe s'exécute n fois (pour i de 1 à n). La boucle interne s'exécute i fois pour chaque valeur de i. Le nombre total d'opérations $T(n)$ est donc la somme :

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \sum_{i=1}^{n} i$$

Cette somme est une série arithmétique bien connue :

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

La complexité dans le pire des cas est déterminée par le terme de plus haut degré. La complexité est donc en $\boldsymbol{O(n^2)}$.

3.1. Donner la formule récurrente de la suite $f_{n,m}$ En observant le code, on peut déduire la relation de récurrence pour $f_{n,m} = \text{mystere}(n, m)$.

- **Cas de base** $(n = 0)$ **:** $f_{0,m} = m$
- **Cas récursif** $(n > 0)$ **:** $f_{n,m} = f_{n-1,m} + m^n$

La formule récurrente est donc :

$$f_{n,m} = \begin{cases} m & \text{si } n = 0 \\ f_{n-1,m} + m^n & \text{si } n > 0 \end{cases}$$

3.2. Donner l'expression explicite de $f_{n,m}$ En développant la récurrence : $f_{n,m} = m^n + f_{n-1,m}$ $f_{n,m} = m^n + m^{n-1} + f_{n-2,m}$ $f_{n,m} = m^n + m^{n-1} + \cdots + m^1 + f_{0,m}$ $f_{n,m} = m^n + m^{n-1} + \cdots + m^1 + m$

Ceci est la somme $\sum_{k=1}^{n} m^k + m$. (Attention, le terme m vient de $f_{0,m}$, et m^1 est le premier terme de la somme déroulée). Vérifions : $f_{1,m} = f_{0,m} + m^1 = m + m = 2m$. Notre formule : $m + \sum_{k=1}^{1} m^k = m + m = 2m$. C'est correct. L'expression est $f_{n,m} = m + \sum_{k=1}^{n} m^k$.

On peut exprimer la somme géométrique :

- Si $\boldsymbol{m = 1}$: $f_{n,1} = 1 + \sum_{k=1}^{n} 1^k = 1 + n$.
- Si $\boldsymbol{m \neq 1}$: La somme $\sum_{k=1}^{n} m^k$ est une série géométrique de premier terme m, de raison m et de n termes. Sa valeur est $m\frac{m^n-1}{m-1}$. Donc, $f_{n,m} = m + m\frac{m^n-1}{m-1}$.

L'expression explicite est :

$$f_{n,m} = \begin{cases} n + 1 & \text{si } m = 1 \\ m\left(1 + \frac{m^n-1}{m-1}\right) & \text{si } m \neq 1 \end{cases}$$

Exercice 4 : Analyse de fonctions Python

1. Fonction F

- **1.1. Que retourne la fonction F pour a=80 ?** La boucle `while` s'exécute tant que $2^n \leq 80$.
  - n=0 : $2^0 = 1 \leq 80 \rightarrow n = 1$
  - n=1 : $2^1 = 2 \leq 80 \rightarrow n = 2$
  - n=2 : $2^2 = 4 \leq 80 \rightarrow n = 3$
  - n=3 : $2^3 = 8 \leq 80 \rightarrow n = 4$
  - n=4 : $2^4 = 16 \leq 80 \rightarrow n = 5$
  - n=5 : $2^5 = 32 \leq 80 \rightarrow n = 6$
  - n=6 : $2^6 = 64 \leq 80 \rightarrow n = 7$
  - n=7 : $2^7 = 128 > 80$. La boucle s'arrête.

  La fonction retourne la dernière valeur de n, soit **7**.

— **1.2. Que fait la fonction F ? Justifier.** La fonction F calcule le plus petit entier n tel que $2^n > a$. *Justification :* La boucle `while` incrémente n tant que la condition $2^n \leq a$ est vraie. Elle s'arrête dès que n atteint la première valeur pour laquelle 2^n devient strictement supérieur à a. La fonction retourne cette valeur de n. Mathématiquement, cela correspond à $\lfloor \log_2(a) \rfloor + 1$.

2. Fonction G

— **2.1. Que retourne la fonction G pour n=30 ?** La boucle `while` parcourt i de 1 à $n//2 = 15$. La variable p accumule les diviseurs de 30. Les diviseurs de 30 jusqu'à 15 sont : 1, 2, 3, 5, 6, 10, 15. $p = 1+2+3+5+6+10+15 = 42$. Après la boucle, l'instruction `p=p+n` est exécutée : $p = 42 + 30 = 72$. La fonction retourne **72**.

— **2.2. Que fait la fonction G ? Justifier.** La fonction G calcule la somme de tous les diviseurs positifs d'un entier n. *Justification :* La variable p est initialisée à 0. La boucle `while` itère de $i = 1$ jusqu'à la moitié de n. La condition `if (n%i==0)` vérifie si i est un diviseur de n. Si c'est le cas, i est ajouté à p. La boucle additionne donc tous les diviseurs de n jusqu'à $n/2$. Finalement, n lui-même (qui est le plus grand diviseur) est ajouté à p. Le résultat est la somme de tous les diviseurs de n.