LU2IN003 Partiel Algorithmique
Aucun document autorisé

Mercredi 29 mars 2023 - Durée 1 heure 30 minutes

Exercice 1 : Etude d'un algorithme itératif (7.5 points)
Soit un entier $n \in \mathbb{N}$. Un diviseur $d \in \mathbb{N}^\star$ de n est dit *propre* si $d \neq n$. Un nombre n est *parfait* si il est égal à la somme de ses diviseurs propres.
Par exemple, les diviseurs propres de $n = 6$ sont les entiers 1, 2 et 3 et on observe que $6 = 1 + 2 + 3$. Le nombre 6 est donc parfait. De même, $28 = 14 + 7 + 4 + 2 + 1$ est parfait.
Les nombres parfaits sont assez rares : le suivant est 496, puis 8128. De plus, on ne sait pas aujourd'hui si il existe des nombres parfaits impairs.
L'algorithme 1 retourne True si et seulement si un nombre est parfait. Le code `n//2+1` retourne $\lfloor \frac{n}{2} \rfloor + 1$. Pour la boucle `for`, la variable i prend les valeurs dans $\{1, \ldots, \lfloor \frac{n}{2} \rfloor\}$ en ordre croissant. Le code `n%i` retourne « n modulo i », autrement dit le reste de la division euclidienne de n par i.

```
def EstParfait(n):
    sum=0
    for i in range(1,n//2+1):
        if (n%i==0):
            sum=sum+i
    return sum==n
```

Algorithme 1 : Retourne True si et seulement si un nombre est parfait.

1. (1 point) Démontrer que la fonction `EstParfait` se termine.

> **Solution:** La boucle de l'Algorithme 1 est effectuée $\lfloor \frac{n}{2} \rfloor$ fois. Le corps de boucle est constitué de au maximum deux instructions. On en déduit que l'algorithme se termine.

2. On souhaite maintenant démontrer la validité de cette fonction. Pour cela, soit la suite $sum_{i^\star}$, pour $i^\star \in \{0, \ldots, \lfloor \frac{n}{2} \rfloor\}$ telle que $sum_0 = 0$ et pour $i^\star > 0$, $sum_{i^\star}$ a pour valeur celle de la variable sum à la fin du corps de boucle pour $i = i^\star$.
   (a) ($\frac{1}{2}$ point) Donner les valeurs successives de la suite $sum_{i^\star}$ pour $n = 12$ et $i^\star \in \{0, \ldots, 6\}$;
   (b) ($\frac{1}{2}$ point) Pour une valeur n fixée, exprimer un invariant de boucle $\mathcal{P}(i^\star)$ en utilisant $sum_{i^\star}$;
   (c) ($2\frac{1}{2}$ points) Démontrer l'invariant de boucle $\mathcal{P}(i^\star)$ par récurrence.

> **Solution:**
>
> 1.
>
> | $i^\star$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
> |---|---|---|---|---|---|---|---|
> | $sum_{i^\star}$ | 0 | 1 | 3 | 6 | 10 | 10 | 16 |

2. Soit un entier $n \in \mathbb{N}$. Pour tout $i^\star \in \{0, \dots, \lfloor \frac{n}{2} \rfloor\}$, $P(i^\star)$: $sum_{i^\star}$ est égal à la somme des diviseurs de n dans l'ensemble $\{0, \dots, i^\star\}$.

3. Cette propriété se démontre par récurrence faible sur $i^\star$.

   Base Pour $i^\star = 0$, au démarrage de la boucle, $sum_0 = 0$. $P(0)$ est donc vérifiée.

   Induction Soit $i^\star \in \{0, \dots, \lfloor \frac{n}{2} \rfloor - 1\}$, supposons que $P(i^\star)$, est vérifiée. On montre que $P(i^\star + 1)$ est vérifiée.

   On a deux cas à considérer :

   (a) Si $i^\star + 1$ divise n, on a alors $sum_{i^\star+1} = sum_{i^\star} + (i^\star + 1)$. Comme $sum_{i^\star}$ est la somme des diviseurs de n dans l'ensemble $\{1, 2, \dots, i^\star\}$, on en déduit que $P(i^\star+1)$ est vérifiée ;

   (b) Sinon, $sum_{i^\star+1} = sum_{i^\star}$ et comme $P(i^\star)$ est vérifié, on en déduit que $P(i^\star + 1)$ l'est également.

   Conclusion La proposition $P(i^\star)$ est donc vérifiée par récurrence faible pour toute valeur $i^\star \in \{0, \dots, \lfloor \frac{n}{2} \rfloor\}$.

3. (1 point) En déduire la validité de la fonction `EstParfait`.

Solution: En sortie de boucle, la variable de boucle i vaut $i^\star = \lfloor \frac{n}{2} \rfloor$. D'après la question précédente, $P(\lfloor \frac{n}{2} \rfloor)$ est vérifiée, et la variable *sum* vaut la somme des diviseurs de n dans l'ensemble $\{1, 2, \dots, \lfloor \frac{n}{2} \rfloor\}$. Comme le plus grand diviseur possible de n est $\lfloor \frac{n}{2} \rfloor$, on en déduit que *sum* vaut la somme des diviseurs de n.

D'après la définition, n est parfait si cette valeur vaut n. On en déduit la validité de la fonction `EstParfait`.

4. (1 point) Peut-on identifier un pire des cas et/ou un meilleur des cas pour `EstParfait` ? En déduire la complexité de cette fonction.

Solution: Une exécution du corps de la boucle contient 1 ou 2 instructions. Le corps de boucle est executé $\lfloor \frac{n}{2} \rfloor$ fois. Il n'y a donc pas de meilleur ou de pire des cas, et la complexité est en $\Theta(n)$.

On considère maintenant l'Algorithme 2 qui retourne le plus petit entier parfait pair.

```
def PlusPetitParfaitPair():
    p=2
    while True:
        if EstParfait(p):
            return p
        p=p+2
```

Algorithme 2 : Retourne le plus petit parfait pair.

5. (a) (½ point) Démontrer la terminaison de cet algorithme.

   (b) (½ point) On souhaite maintenant modifier cet algorithme pour calculer le plus petit nombre premier impair. Que pensez vous de la terminaison de ce nouvel algorithme ?

Solution:

1. On observe que toute exécution du corps de boucle se termine, car `EstParfait(p)` se termine, ainsi que le `return` et l'incrémentation de p. De plus, comme $p = 2$ et que toutes les valeurs de p sont paires, la fonction va boucler jusque $p = 6$. Donc, le corps de boucle est exécuté 3 fois (pour les valeurs $p \in \{2, 4, 6\}$). On en déduit que cet algorithme se termine.
2. L'énoncé affirme que l'on ne sait pas si il existe un nombre parfait impair. En posant $p = 1$, on ne peut donc rien dire sur le nombre de tours de boucles de l'Algorithme 2. On ne peut donc rien déduire sur sa terminaison.

Exercice 2 : Arbres binaires (7.5 points)

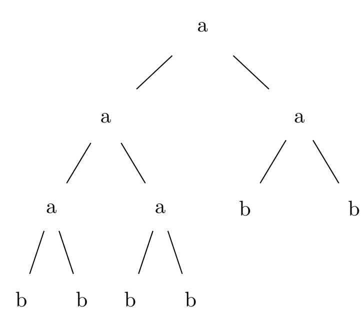
L'ensemble $ABba$ est défini inductivement de la manière suivante :

— $(b, \emptyset, \emptyset) \in ABba$

— si $G \in ABba$ et $D \in ABba$ alors $(a, G, D) \in ABba$.

1. (1 point) Dessiner un arbre de $ABba$ de taille 11 (c'est-à-dire ayant 11 nœuds au total).

Solution: Par exemple :



2. (1 point) Donner une définition inductive de la taille $n(A)$ d'un arbre $A \in ABba$.

Solution:
Base. Si $A = (b, \emptyset, \emptyset)$ alors $n(A) = 1$.
Induction. Si $A = (a, G, D)$ alors $n(A) = n(G) + n(D) + 1$.

3. ($1\frac{1}{2}$ points) Démontrer, par induction structurelle, que la taille d'un arbre de $ABba$ est toujours impaire.

Solution:
Base. Si $A = (b, \emptyset, \emptyset)$ alors $n(A) = 1$ donc $n(A)$ est impaire.
Induction. Soit $G \in ABba$ et $D \in ABba$, supposons que $n(G)$ et $n(D)$ sont impaires et soit $A = (a, G, D)$.
Alors $n(A) = n(G) + n(D) + 1$ est la somme de 3 nombres impairs donc $n(A)$ est impaire.
Conclusion. On a montré, par induction structurelle, que la taille d'un arbre de $ABba$ est toujours impaire.

4. (3 points) Démontrer, par induction structurelle, que si $A \in ABba$ est de taille $n = 2k + 1$ alors le parcours infixe de A est égal à $[1] + [0, 1] * k$ (c'est-à-dire à la liste `[b,a,b,a,b,...,a,b]`, de longueur $2k + 1$).

Solution: Pour $A \in ABba$, on note $infixe(A)$ le parcours infixe de A. Montrons que, si $n(A) = 2k+1$ alors $infixe(A) =$ `[b]+[a,b]*k`.

Base. Si $A = (b, \emptyset, \emptyset)$ alors $k = 0$, $infixe(A) = [b] = [b] + [a,b] * 0$.

Induction. Soit $G \in ABba$ et $D \in ABba$, supposons que la propriété est vraie pour G et D et soit $A = (a, G, D)$.

On a $n(G) = 2k_1 + 1$, $n(D) = 2k_2 + 1$ et $n(A) = 2k + 1$ avec $k = k_1 + k_2 + 1$.

Par hypothèse d'induction, $infixe(G) =$ `[b]+[a,b]`$*k_1$ et $infixe(D) =$ `[b]+[a,b]`$*k_2$.

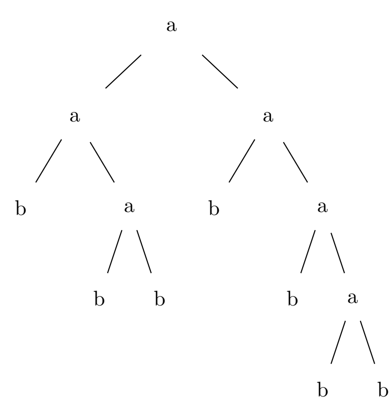
Donc $infixe(A) = infixe(G) +$ `[a]` $+ infixe(D) =$ `[b]+[a,b]`$*k_1 +$ `[a]` $+$ `[b]+[a,b]`$*k_2$.

D'où $infixe(A) =$ `[b]+[a,b]`$*(k_1 + 1 + k_2) =$ `[b]+[a,b]*k`.

Conclusion. On a montré, par induction structurelle, que si A est un arbre de $ABba$ de taille $n = 2k+1$ alors le parcours infixe de A est égal à `[b]+[a,b]*k`.

5. (1 point) Dessiner un arbre de $ABba$ dont le parcours préfixe est $[a, a, b, a, b, b, a, b, a, b, a, b, b]$. Y-a-t-il plusieurs solutions ?

Solution:
Un arbre de $ABba$ dont le parcours préfixe est $[a, a, b, a, b, b, a, b, a, b, a, b, b]$:



La solution est unique.

Exercice 3 : QCM (5 points) Un seul choix est possible. Une bonne réponse = 0.5 points. Une mauvaise réponse = -0.25 points. Si la note globale du QCM est négative, on la considère égale à 0.

1. (½ point) Pour démontrer $a \Rightarrow b$ par l'absurde, on doit :
   - ○ démontrer que si a et b sont faux, on obtient une contradiction ;
   - ○ démontrer que si b est faux, alors a est faux ;
   - √ **démontrer que l'on si a et vrai et b faux, on obtient une contradiction ;**
   - ○ Aucun des choix précédents.
2. (½ point) A quel ensemble apartient $u = \frac{n^2}{\sqrt{n}}$?
   - √ $u \in \Omega(n) \cap \mathcal{O}(n^2)$ **;**
   - ○ $u \in \Theta(n^2)$;
   - ○ $u \in \Omega(n^2) \cap \mathcal{O}(n^2 \log n)$;
   - ○ Aucun des choix précédents.

3. ($1/2$ point) Soit la suite $u_n = 2u_{\frac{n}{2}} + 1$ définie pour $n = 2^k$, $k \in \mathbb{N}$ et $u_1 = 1$. Quel est le terme général de u_n ?
   - ◯ $u_n = 2^n - 1$;
   - √ **$u_n = 2n - 1$;**
   - ◯ $u_n = \log_2 n - \log_2(1) + 1$;
   - ◯ Aucun des choix précédents.
4. ($1/2$ point) Dans les deux questions suivantes, on considère la fonction `mystere` dont le code suit.

```
def mystere(n):
    if (n==1):
        return 5
    return mystere(n-1)+(n+1)
```

   Une seule des affirmation est vérifiée, laquelle ?
   - ◯ `mystere(6)` retourne 30 et `mystere(9)` retourne 47 ;
   - √ **`mystere(6)` retourne 30 et `mystere(9)` retourne 57 ;**
   - ◯ `mystere(6)` retourne 23 et `mystere(9)` retourne 47 ;
   - ◯ Aucun des choix précédents.
5. ($1/2$ point) On considère la fonction `mystere` exprimée dans la question précédente. Une seule des affirmation est vérifiée, laquelle ?
   - √ **Pour $n \geq 1$, `mystere(n)` retourne $2 + \frac{(n+1).(n+2)}{2}$;**
   - ◯ Pour $n \geq 1$, `mystere(n)` retourne $5 + \frac{(n+1).(n+2)}{2}$;
   - ◯ Pour $n \geq 1$, `mystere(n)` retourne $5 + \frac{(n+1).(n)}{2}$;
   - ◯ Aucun des choix précédents.
6. ($1/2$ point) Dans les deux questions suivantes, on considère la fonction `mystereListe` dont le code suit. $L[1:]$ désigne la liste L sans son premier élement $L[0]$.

```
def mystereListe(L,x):
    res=0
    if (len(L)!=0):
        res=L[0]+x*mystere(L[1:],x)
    return res
```

   Une seule des affirmation est vérifiée, laquelle ?
   - ◯ La fonction ne se termine pas pour certaines listes ;
   - √ **Pour toute liste de n entiers, l'appel `mystereListe(L,x)` se termine et retourne $\sum_{i=0}^{n-1} x^i.L[i]$;**
   - ◯ Pour toute liste de n entiers, l'appel `mystereListe(L,x)` se termine et retourne $\sum_{i=1}^{n} x^i.L[i]$;
   - ◯ Aucun des choix précédents.
7. ($1/2$ point) On suppose dans cette question que la liste L est représentée sous la forme d'une liste simplement chaînée. Quelle est la complexité de la fonction `mystereListe(L,x)` ?
   - ◯ $\Theta(n^2)$;
   - √ **$\Theta(n)$;**
   - ◯ $\Omega(1)$ et $\mathcal{O}(\log n)$;
   - ◯ Aucun des choix précédents.
8. ($1/2$ point) Une seule assertion ne peut pas être vraie, laquelle ?
   - ◯ Ma fonction de tri par comparaison a une complexité en $\Omega(n)$;
   - ◯ Ma fonction de tri par comparaison a une complexité en $\Theta(n \log n)$;

√ **Ma fonction de tri par comparaison a une complexité en $\Theta(n)$.**

9. (½ point) La complexité du Quicksort est en :
   - ◯ $\Theta(n \log n)$;
   - √ $\Omega(n \log n)$ **;**
   - ◯ $\mathcal{O}(n \log n)$;
   - ◯ Aucun des choix précédents.
10. (½ point) La complexité du tri à bulles est en :
    - ◯ $\mathcal{O}(n \log n)$ et $\Omega(n)$;
    - ◯ $\Omega(n^2)$ et $\mathcal{O}(n \log n)$;
    - √ $\Theta(n^2)$ **;**
    - ◯ Aucun des choix précédents.