Série 3 : Les équations de récurrence et problèmes Diviser pour Régner (DpR)

Corrigés

Exercice1 : Résoudre les équations suivantes :

a. $\begin{cases} T(n) = T(n\text{-}1) + 2n\text{-}1 & \text{si} \quad n>0 \\ T(0) = 0 \end{cases}$

b. $\begin{cases} T(n)=T(n\text{-}1)+2^n \\ T(0)=1 \end{cases}$

c. $\begin{cases} T(1) = 1 \\ T(n) = 2n + 1 + 2T(^n/_2) \end{cases}$

d. $T(n) = \begin{cases} 1 & si\ n = 1 \\ 16T\left(\frac{n}{4}\right) + n^2 & si\ n > 1 \end{cases}$

Corrigé

a. T(n)=T(n-1) + 2n-1 si n>0 et g(0)=0

=T(n-2)+2(n-1)-1 + 2n-1

=T(n-3)+2(n-2)-1 + 2(n-1)-1 + 2n-1

=T(n-4)+ 2(n-3)-1 + 2(n-2)-1 + 2(n-1)-1 + 2n-1

....

= T(0)+2($\sum_{i=1}^{n} i$)-n

= 0 + 2(n(n+1)/2) –n

= n^2

b. $T(n) = T(n-1) + 2^n$

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

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

…

$= T(0) + 2^1 + \cdots + 2^{n-2} + 2^{n-1} + 2^n$

Sachant que $\sum_0^n 2^k = 2^{k+1} - 1$

$T(n)=T(0) + \sum_{i=1}^{n} 2^i = 1 + 2^{n+1} - 1 - 1 \Rightarrow T(n) = \theta(2^n)$

c. $T(n) = 2n + 1 + 2T(n/2)$

$= 2n + 1 + 2(2(n/2) + 1 + 2T(n/2^2))$

$= 2n + 1 + 2n + 2 + 2^2T(n/2^2)$

$= 2n + 1 + 2n + 2 + 2^2(2(n/2^2) + 1 + 2T(n/2^3))$

$= 2n + 1 + 2n + 2 + 2n + 2^2 + 2^3T(n/2^3)$

$= \ldots$

$= 2n+2n+\ldots+2n+1+2+2^2+\ldots2^{k-1}+2^kT(n/2^k)$

$= 2n*k+\sum_{i=0}^{k-1} 2^i + 2^kT(n/2^k)$

On pose n=2^k $d'où\ \ k = logn$

T(n) = 2nlogn+$(2^k - 1)$+n*T(1)

T(n) = 2nlogn+n-1+n = 2nlogn+2n-1 d'où la complexité $\sim \Theta(nlogn)$

d. $T(n) = 16T\left(\frac{n}{4}\right) + n^2$

$= 16\left(16T\left(\frac{n}{4^2}\right) + \frac{n^2}{4^2}\right) + n^2$

$= 16^2T\left(\frac{n}{4^2}\right) + n^2 + n^2$

$= 16^2\left(16T\left(\frac{n}{4^3}\right) + \frac{n^2}{4^4}\right) + 2n^2$

$= 16^3\ T\left(\frac{n}{4^3}\right) + 3n^2$

…

$= 16^k\ T\left(\frac{n}{4^k}\right) + kn^2$

On pose n=4^k

$= n^2T(1) + n^2\log_4 n = n^2 + n^2\log_4 n = \Theta(n^2\log n)$

Exercice 2 : Soit l'équation de récurrence :

$$T(n) = \begin{cases} 1 & si\ n = 1 \\ \sum_{i=1}^{i=n-1} T(i) + 1 & si\ n \geq 2 \end{cases}$$

a. Calculez T(n) - T(n - 1); pour n ≥ 2.
b. Résoudre T(n)

Corrigé

$$T(n) - T(n-1) = \sum_{i=1}^{n-1} T(i) + 1 - \left(\sum_{i=1}^{n-2} T(i) + 1\right)$$

$$= \overbrace{\cancel{\sum_{i=1}^{n-2} T(i)} + T(n-1)} \cancel{+1} - \cancel{\sum_{i=1}^{n-2} T(i)} \cancel{-1}$$

$$= T(n-1)$$

⇨ $T(n) = 2 * T(n-1)$

Résolution :

$T(n)=2T(n-1)$

$=2(2T(n-2)) = 2^2T(n-2)$

$=2^2(2T(n-3))=2^3T(n-3)$

…

$= 2^{n-1}T(1)$

$=2^{n-1} \Rightarrow \Theta(2^n)$

Exercice 3 : Calculer en justifiant la complexité des deux fonctions suivantes :

Fonction Rec1(E/ n : entier) : entier {
Si (n=1) alors retourner 1 sinon retourner 2*Rec1(n-1) finsi ; }
Fonction Rec2(E/n : entier) : entier {
Si (n=1) alors retourner 1 sinon retourner Rec2(n-1) + Rec2(n-1) finsi ; }
Que constatez-vous ?

Corrigé :
Pour Rec1 un seul appel récursif donc l'équation de récurrence est la suivante :
$T(1)=1$
$T(n)=T(n-1)+1$

Après résolution on a $T(n) = T(1) + (n-1) * 1 = n$ d'où la complexité est de l'ordre de $O(n)$.
Pour Rec2 on a deux appels récursifs donc l'équation de récurrence est la suivante :
T(1)=1
T(n)=2*T(n-1)+1

Après résolution on a : $T(n) = 2^{n-1}T(n-(n-1)) + \sum_{i=0}^{n-2} 2^i$

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

$T(n) = 2^n + 1$

D'où la complexité est de l'ordre de $O(2^n)$.
Rec1, qui ne contient qu'un seul appel récursif est linéaire par contre Rec2 avec deux appels récursifs est exponentielle. On va préférer bien sûr Rec1.

Exercice 4 : On considère la fonction F suivante :

Fonction F(E/ n : entier) : entier
Début
si (n=0) alors retourner 0
sinon si (n=1) alors retourner 2
*sinon retourner 4*F(n-1) – 4*F(n-2) ;*
finsi ;
finsi ;
Fin ;

a. Que calcule cette fonction ? Le prouver
b. Déterminer la complexité de F. Justifier votre réponse.
c. Est-il possible d'améliorer la complexité de F ? Si oui donner la nouvelle fonction ainsi que sa complexité.

Corrigé : F(2)=8, F(3)=24, F(4)=64, F(5)=160,…

a. Cette fonction calcul : $F(n) = n2^n$
Preuve par récurrence
n=0 $0 * 2^0 = 0$ cette fonction est vérifiée pour n=0
n=1 $1 * 2^1 = 2$ cette fonction est vérifiée pour n=1
On suppose que cette fonction est vérifiée pour n F(n)= $n2^n$ et on montre qu'elle est vérifiée pour n+1.
On montre que F(n+1)= $(n+1)2^{n+1}$

$$F(n+1) = 4\big(F(n) - F(n-1)\big)$$

$$F(n+1) = 4n2^n - 4(n-1)2^{n-1}$$
$$= 4n2^n - (4n-4)2^{n-1}$$

$= 4n2^n - 4n2^{n-1} + 4 * 2^{n-1}$

$= 4n2^n - 4n * \frac{2^n}{2} + 4 * \frac{2^n}{2}$

$= 4n2^n - 2n2^n + 2 * 2^n$

$= 2n2^{n+1} - n2^{n+1} + 2^{n+1}$

$= n2^{n+1} + 2^{n+1}$

$= (n+1)2^{n+1} c.q.f.d.$

b. Complexité de l'ordre de $O(2^n)$
Pour n=4 on a $2^0 + 2^1 + 2$ appels récursifs
Pour n=5 on a $2^0 + 2^1 + 2^2 + 2$ appels récursifs
Pour n=6 on a $2^0 + 2^1 + 2^2 + 2^3 + 2$ appels récursifs
Ce qui donne la formule $2+\sum_{i=0}^{n-3} 2^i = 2 + 2^{n-2} + 1 = 3 + 2^{n-2} \sim 0(2^n)$

c. Oui, il possible d'améliorer en éliminant la récursivité (fonction itérative)

Fonction F_Ité(E/ n : entier) : entier
Début
si (n=0) alors retourner 0
sinon si (n=1) alors retourner 2
sinon x←0 ; y←2 ;
pour i←2 à n faire
z←4(y-x) ; x←y ; y←z ;
fait ;
retourner z ;
finsi ;
finsi ;
Fin ;

Complexité linéaire $\sim O(n)$

<u>Exercice 5 :</u> Soit la fonction mystère suivante :

Fonction mystère (E/ n, m : entier) ;
Début
Si (m=0) alors retourner 0
Sinon retourner mystère (2*n, m div 2) +n*(m mod 2) ;
Fin ;

a. Déroulez cette fonction avec n=8 et m=3 puis n=5 et m=7
b. Que calcule cette fonction ? Donner et justifier la complexité de mystère.
c. Donner la fonction itérative équivalente à mystère. Donner sa complexité.
d. Qu'allez-vous préférer la solution itérative ou récursive ?

Corrigé :

a. Déroulement n=8 et m=3

mystère(8,3) mystère(16,1) mystère(32,0)

mystère(8,3) retourner mystère(16,1)+8*1 retourner mystère(32,0)+16*1 retourner 0

=24 = 16+8 = 0+16

n=5 et m=7

mystère(5,7) mystère(10,3) mystère(20,1) mystère(40,0)

retourner 0

mystère(5,7) retourner mystère(10,3)+5*1 retourner mystère(20,1)+10*1 retourner mystère(40,0)+20*1

=35 = 30+5 = 20+10 = 0+20

b. Elle calcule n*m

c. T(n, 0)=1

T(n, m)=T(n*2,m/2)+1

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

$=T(n*2^3, \frac{m}{2^3})+1+1+1= T(n*2^3, \frac{m}{2^3})+3$

…

$= T(n*2^k, \frac{m}{2^k})+k$

On pose $m= 2^{k-1} \Rightarrow logm = log2^{k-1}$

$logm = (k-1)log2 \Rightarrow k = logm + 1$

$T(n,m)= T(n*2^k, \frac{1}{2})+k$

$= T(n*2^k, 0)+k$

$= 1 + logm+1 \Rightarrow$ *la complexité de l'ordre de* $\boldsymbol{\Theta(logm)}$

Exercice 5 : Recherche *du maximum et du minimum d'un tableau*

Soit l'algorithme naïf suivant qui retourne respectivement le maximum et le minimum des éléments d'un tableau d'entiers A[1..n].

```
Fonction MinMaxNaif(A : Tableau d'Entier, n : Entier) : Couple d'Entier
Variables i, min, max : Entier;
Début   min := A[1]; max := A[1];
    Pour i := 2 à n  faire
        Si (A[i] < min) Alors min := A[i]
                        Sinon Si (A[i] > max) Alors max := A[i]; FinSi
        FinSi
    FinPour
```

retourner(min, max);

Fin ;

Taille de l'entrée : n (le nombre des éléments dans le tableau)

Opérations fondamentales : comparaisons entre les éléments du tableau.

a) Quel est le meilleur cas et donnez la complexité.
b) Quel est le pire des cas et donnez la complexité.
c) Donnez la solution récursive DpR de ce problème. Quelle est sa complexité ?
d) Qu'allez-vous préférer la solution itérative ou récursive ?

Corrigé

a. Le meilleur cas c'est quand le tableau est trié dans l'ordre décroissant donc on a n-1 comparaisons
b. Le pire cas c'est quand le tableau est trié dans l'ordre croissant donc on a 2n-2 comparaisons
c. On propose de diviser le problème en deux sous-problèmes

```
Fonction MinMax(A : Tableau d'Entier, d, f : Entier) : Couple d'Entier
Variables milieu, min1, max1, min2, max2 : Entier;
Début
   Si (f=d=1) alors retourner(A[d], A[f]) // cas où le tableau contient une seule valeur
   Sinon Si ((f-d) ≤ 1) alors retourner(Min(A[d], A[f]), Max(A[d], A[f]))
         Sinon  milieu=⌊d + f/2⌋ ;
                (min1, max1)=MinMax(A, d,m) ;
                (min2, max2)=MinMax(A,m+1,f) ;
                Retourner (Min(min1, min2), Max(max1, max2)) ;
         fsi ;
   fsi ;
Fin ;
```

L'équation de récurrence de cette fonction :

T(n)=2*T(n/2) + 2 (2 appels récursifs et 2 comparaisons 1 pour Min+ 1 pour Max)

T(1)=0 (une seule valeur dans le tableau donc aucune comparaison)

T(2)=2 (deux valeurs donc 2 comparaisons)

Après substitution on obtient :

$T(n)=2^{k-1}*T(n/2^{k-1})+2^{k-1} + \ldots + 2^1$

On pose $n=2^k$

$T(n)=2^k * 2^{-1} * T\left(\frac{2^k}{2^{k-1}}\right) + \sum_{i=0}^{k} 2^i - 2^k - 2^0$

$T(n)=2^k * 2^{-1} * T\left(\frac{2^k}{2^{k-1}}\right) + 2^{k+1} - 1 - 2^k - 1$

$T(n)=\frac{n}{2} * T(2) + 2n - n - 2$

T(2)=2 et on obtient $2n - 2$ opérations

Réponse : Cette solution n'améliore pas la solution naïve.