Mesure des algorithmes récursifs

D.E ZEGOUR

École Supérieure d'Informatique

ESI

Mesure des algorithmes récursifs

Méthode

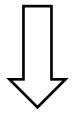
Pour mesurer un algorithme récursif

- trouver l'équation de récurrence
- la résoudre

Mesure des algorithmes récursifs

Équation de récurrence

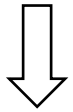
```
Fact(n)
Si n ≤ 1
   Fact := 1
Sinon
   Fact := n * Fact(n-1)
Fsi
```



$T(n) = a$ si $n \leq 1$

$= T(n-1) + b$ sinon

```
Tri(L, n) { Supposons n = 2^k }
Si n = 1
   Tri := L
Sinon
    L1 := L [1..n/2]
    L2 := L [n/2+1..n]
    Tri := Fusion( Tri(L1, n/2), tri(L2, n/2))
Fsi
```



$T(n) = a$ si $n=1$

$= 2\ T(n/2) + bn$ Sinon

Mesure des algorithmes récursifs

Résolution

Il existe en général 3 Méthodes

a) Par substitution (dilater la récurrence)

b) Deviner une solution f(n) et la démontrer par récurrence.

c) Utiliser les solutions de certaines équations de récurrence connues.

Mesure des algorithmes récursifs

Résolution par dilatation (Exemple 1)

```
Fact(n)
Si n ≤ 1
   Fact := 1
Sinon
   Fact := n * Fact(n-1)
Fsi
```

T(n) = a si n ≤ 1
T(n-1) + b sinon

Équation de récurrence

T(n) = b + T(n-1)
= b + (b + T(n-2))
= 2b + T(n-2)
= 2b + (b + T(n-3)) = 3b + T(n-3)
= ...
= (ib) + T(n-i)
= ...
= (n-1)b + T(1)
= nb - b + a

C'est $O(n)$

Mesure des algorithmes récursifs

Résolution par dilatation (Exemple 2)

```
Tri(L, n) { Supposons n = 2^k }
Si n = 1
   Tri := L
Sinon
    L1 := L [1..n/2]
    L2 := L [n/2+1..n]
    Tri := Fusion( Tri(L1, n/2), tri(L2, n/2))
Fsi
```

T(n) = a si n=1
= 2 T(n/2) + bn Sinon

Équation de récurrence

T(n) = 2 T(n/2) + bn
= 2 [2 T(n/4) + bn/2] + bn
= 4 T[n/4] + 2 bn
= 8 T[n/8] + 3 bn
= ...
= 2^i T[$n/2^i$] + ibn
= ...
= n T[1] + Log(n) bn
= an + Log(n) bn
= n(a + bLog(n))

C'est O(n Log(n))

Mesure des algorithmes récursifs

Résolution par devinette (Exemple)

```
Tri(L, n) { Supposons n = 2^k }
Si n = 1
   Tri := L
Sinon
    L1 := L [1..n/2]
    L2 := L [n/2+1..n]
    Tri := Fusion( Tri(L1, n/2), tri(L2, n/2))
Fsi
```

$T(n) = c_1$ si n=1
$= 2\,T(n/2) + c_2 n$ Sinon **(1)**

Équation de récurrence

On veut montrer que T(n) = O(nlog2(n))
c'est à dire $T(n) \leq anLog(n) + b$ pour a et b donnés à partir d'un rang n.

Si n = 1 , $T(1) \leq b$ (On peut prendre $b= c_1$)

On suppose $T(k) \leq akLog(k) + b$ pour tout k < n
et on essaie d 'établir que $t(n) \leq anLogn + b$

Supposons $n \geq 2$,
de **(1)** on obtient
$T(n) \leq 2T(n/2) + c_2 n$
$\leq 2(a\,(n/2)Log(n/2) + b) + c_2 n$
$\leq anLog(n) - anLog(2) + 2b + c_2 n$
$\leq anLog(n) - an + 2b + c_2 n$
$\leq anLog(n) + b + (b + c_2 n - an)$

Mesure des algorithmes récursifs

Résolution par devinette (Exemple)

```
Tri(L, n)
Si n = 1
   Tri := L
Sinon
    L1 := L [1..n/2]
    L2 := L [n/2+1..n]
    Tri := Fusion( Tri(L1, n/2), tri(L2, n/2))
Fsi
```

$T(n) = c_1$ si n=1
$= 2\ T(n/2) + c_2 n$ Sinon

(1)

Équation de récurrence

Pour que $T(n) \leq an\text{Log}(n) + b$ il faut que
$b + c_2 n - an \leq 0$
$an \geq b + c_2 n$
$a \geq (b + c_2 n)/n$

Pour tout $n \geq 1$ $\quad a \geq b + c_2$

Donc $T(n) \leq an\text{Log}(n) + b$ que si les deux conditions suivantes sont satisfaites :
$b \geq c_1$
$a \geq b + c_2$

En choisissant $b = c_1$ et $a = c_1 + c_2$, on conclut que pour tout n >1
$T(n) \leq (c_1 + c_2)\ n\text{Log}(n) + c_1$

En d'autres termes, T(n) est $O(n\ \text{Log}(n))$

Mesure des algorithmes récursifs

Résolution par équation de récurrence

1. Équations homogènes

$a_0\, t_n + a_1\, t_{n-1} + \ldots a_k\, t_{n-k} = 0$ **(1)**

admet comme équation caractéristiques

$a_0\, x^k + a_1\, x^{k-1} + \ldots + a_k = 0$

Solutions dans R :

$r_1, r_2, \ldots . r_k$.

Si toutes les solutions ri sont distinctes, solution de (1) est $t_n = c_1(r_1)^n + c_2(r_2)^n + \ldots . . c_k(r_k)^n$

Si r_j est une solution multiple (de multiplicité m), solution de **(1)** est

$t_n = c_1(r_1)^n + c_2(r_2)^n + \ldots + \mathbf{(c_{j1}(r_j)^n + c_{j2} n\,(r_j)^n + \ldots c_{jm} n^{m-1}(r_j)^n\ )} + \ldots . . + c_k(r_k)^n$

Remarque : les constantes c_j sont déterminées par les conditions initiales

Mesure des algorithmes récursifs

Rappel solution

$t_n = c_1(r_1)^n + c_2(r_2)^n + ...c_k(r_k)^n$

Equation homogène (Exemple)

Equation homogène

$t_n - 3t_{n-1} - 4t_{n-2} = 0$ pour $n \geq 2$

$t_0 = 0, t_1 = 1$

Equation caractéristique : $x^2 - 3x - 4 = 0$

Solutions : -1 et 4.

$t_n = c_1(-1)^n + c_2 4^n$

Conditions initiales

$0 = c_1 + c_2$

$1 = - c_1 + 4c_2$

$c_1 = -1/5$ et $c_2 = +1/5$

$t_n = -(1/5)(-1)^n + (1/5)4^n$

c'est $O(4^n)$.

Mesure des algorithmes récursifs

Résolution par équation de récurrence

2. Équations non homogènes

$a_0t^n + a_1t^{n-1} + ...a_kt^{n-k} = b_1^nP1(n) + b_2^nP2(n)+....$

b_i : constantes

P_i : des polynômes de degré d_i

Equation caractéristique

$(a_0x^k + a_1x^{k-1}+...a_k)\ (x-b_1)^{d1+1}\ (x-b_2)^{d2+1}\ .. = 0$

Solutions dans R :

$r_1, r_2, ...r_k$.

Si toutes les solutions ri sont distinctes, solution de (1) est $t_n = c_1(r_1)^n + c_2(r_2)^n + ...c_k(r_k)^n$

Si r_j est une solution multiple (de multiplicité m), solution de **(1)** est

$t_n = c_1(r_1)^n + c_2(r_2)^n + ... + \mathbf{(c_{j1}(r_j)^n + c_{j2}n\ (r_j)^n + ...\ c_{jm}n^{m-1}(r_j)^n\ )} + ...+ c_k(r_k)^n$

Remarque : les constantes c_j sont déterminées par les conditions initiales

Rappel : $a_0t^n + a_1t^{n-1} + ...a_kt^{n-k} = b_1{}^nP1(n) + b_2{}^nP2(n)+..$

Rappel $t_n = c_1(r_1)^n + c_2(r_2)^n + ...c_k(r_k)^n$

Mesure des algorithmes récursifs

Equation non homogène (Exemple 1)

Suite de Fibonacci
Fib(n) := Fib(n-1) + Fib(n-2) si n > 1
Fib(0) = 0, Fib(1) = 1

Equation de récurrence
T(n) = T(n-1) + T(n-2) + b si n > 1
T(n) = a sinon

Equation non homogène
$t_n - t_{n-1} - t_{n-2} = b = 1^n\ b$
$t_0 = 0, t_1 = 1,$

$b_1 = 1, P_1 = b$, b étant un polynôme de degré 0

Équation caractéristique : $(x^2 - x - 1)(x-1) = 0$

$(x-r1)\ (x-r2)(x-1) = 0$

$r1 = (1 + \sqrt{5})/2$
$r2 = (1 - \sqrt{5})/2$
$r3 = 1$

Donc $t_n = c_1\ ((1 + \sqrt{5})/2)^n + c_2\ ((1 - \sqrt{5})/2)^n + c_3\ 1^n$

Mesure des algorithmes récursifs

Equation non homogène (Exemple 1)

$t_n = c_1 ((1 + \sqrt{5})/2)^n + c_2 ((1 - \sqrt{5})/2)^n + c_3 1^n$

Conditions initiales

$0 = c_1 + c_2 + c_3$

$1 = c_1(1 + \sqrt{5})/2 + c_2(1 - \sqrt{5})/2 + c_3$

$0 = c_1 + c_2$

$1 = c_1(1 + \sqrt{5})/2 + c_2(1 - \sqrt{5})/2$

$c_2 = - c_1$

$c_1(1 + \sqrt{5})/2 - c_1(1 - \sqrt{5})/2 = 1$

$c_1\sqrt{5} = 1$

Les conditions initiales donnent

$c_1 = 1/\sqrt{5}$, $c_2 = -1/\sqrt{5}$ et $c3 = 0$

$t_n = 1/\sqrt{5} ((1 + \sqrt{5})/2)^n - 1/\sqrt{5} ((1 - \sqrt{5})/2)^n$

c'est $\mathbf{O((1 + \sqrt{5})/2)^n)}$

c'est $O(1.6180339^n)$

Rappel : $a_0t^n + a_1t^{n-1} + ...a_kt^{n-k} = b_1^nP1(n) + b_2^nP2(n)+....$

Mesure des algorithmes récursifs

Equation non homogène (Exemple 2)

Rappel : r_j solution multiple (m)

$t_n = c_1(r_1)^n + c_2(r_2)^n + ... + \mathbf{(c_{j1}(r_j)^n + c_{j2}n\,(r_j)^n + ... c_{jm}n^{m-1}(r_j)^n\ )} + ...+ c_k(r_k)^n$

$t_n - 2t_{n-1} = n + 2^n$

$t_0 = 0$

$b_1 = 1, P_1 = n; b_2 = 2, P_2 = 1$

Équation caractéristique : $(x-2)\ (x-1)^2\ (x-2) = 0$

Donc $t_n = c_1\ 1^n + c_2\ n\ 1^n + c_3\ 2^n + c_4\ n\ 2^n$

Les conditions initiales donnent $c_1=-c_3$, c_2 et c_4 quelconques

c'est $O(n\ 2^n)$.