Cours 2 : Complexité des algorithmes récursifs

Jean-Stéphane Varré

Université Lille 1

jean-stephane.varre@lifl.fr

Rappels

Algorithme récursif :

- un algorithme qui se rappelle lui-même
- récursivité simple ou linéaire (par exemple factorielle, tours de Hanoï) / récursivité croisée ou mutuelle (par exemple test de parité)

Conception :

- définir les bonnes conditions d'arrêt
  (par exemple lorsque le tableau a 0 éléments)
- s'assurer que les appels récursifs font bien aller vers les conditions d'arrêt
  (par exemple en s'assurant que le tableau dans l'appel récursif est plus petit)

Permet souvent d'exprimer de manière simple la résolution d'un problème

Rappel 1/2

Présence d'un élément dans une liste

```
function estPresent (l : LISTE; e : ELEMENT) : BOOLEAN;
begin
   estPresent :=
      (not estListeVide(l)) and
      ((tete(l) = e) or (estPresent(reste(l),e)));
end {estPresent};
```

On compte : les comparaisons, et celles faites dans les appels récursifs

Pour une liste de longueur n, on a dans le pire des cas :

$$\begin{cases} c(0) & = & 0 \\ c(n) & = & 1 + c(n-1) \end{cases}$$

Rappel 2/2

Calcul des nombres de Catalan

```
function catalan(n : CARDINAL) : CARDINAL;
var
   k, r : CARDINAL;
begin
   r := 0;
   if (n = 0) or (n = 1) then
      r := 1
   else
      for k := 0 to n-1 do
         r := r + catalan(n-k-1) * catalan(k);
   catalan := r;
end {catalan};
```

On compte : les multiplications, et celles faites dans les appels récursifs

Pour calculer la valeur du nombre de Catalan de n, on a :

$$\begin{cases} c(0) & = & 0 \\ c(1) & = & 0 \\ c(n) & = & \sum_{k=0}^{n-1} (1 + c(n-k-1) + c(k)) \end{cases}$$

Etablissement des équations de récurrence

On va compter :

- soit les opérations

Sur l'exemple des nombres de Catalan

$$\begin{cases} c(0) & = & 0 \\ c(1) & = & 0 \\ c(n) & = & \sum_{k=0}^{n-1} 1 + c(n-k-1) + c(k) \end{cases}$$

- soit les appels récursifs

Sur l'exemple des nombres de Catalan

$$\begin{cases} c(0) & = & 1 \\ c(1) & = & 1 \\ c(n) & = & 1 + \sum_{k=0}^{n-1} c(n-k-1) + c(k) \end{cases}$$

Recherche dichotomique

Principe

Problème : recherche d'un élément e dans un tableau trié de taille n.
Solution : tester la valeur recherchée par rapport à l'élément en position $m = n/2$ puis recommencer récursivement sur le bon demi tableau.

1 . . . 5 . . . . 10 15	g	d	m	t[m] = 11 ?	
3 4 7 9 9 10 12 17 20 21 22 23 27 29 30	1	15	8	$<$	faux
3 4 7 9 9 10 12	1	7	4	$>$	faux
9 10 12	5	7	6	$>$	faux
12	7	7	7	résultat	

Recherche dichotomique

Complexité

```
function estPresentDico (t : TABLEAU; e : ELEMENT; g,d : CARDINAL) :
var
   m : CARDINAL;
begin
   m := (g+d) div 2;
   write(g); write('-'); write(d); write('-'); write(m); writeln;
   if g > d then
      estPresentDico := false
   else if g = d then
      estPresentDico := (e = t[m])
   else if e < t[m] then
      estPresentDico := estPresentDico(t,e,g,m-1)
   else if t[m] < e then
      estPresentDico := estPresentDico(t,e,m+1,d)
   else
      estPresentDico := true;
end {estPresentDico};
```

Pire des cas : l'élément n'est pas trouvé ou bien trouvé dans la dernière case explorée.

$$\begin{cases} c(0) & = & 1 \\ c(n) & = & 1 + c(\lfloor n/2 \rfloor) \end{cases}$$

Types d'équation de récurrence

3 types d'équations :

- équations de récurrence linéaires

$$c(n) = \alpha_1 c(n-1) + \alpha_2 c(n-2) + \cdots + \alpha_k c(n-k) + f(n)$$

- équations de récurrence linéaires sans second membre

$$c(n) = \alpha_1 c(n-1) + \alpha_2 c(n-2) + \cdots + \alpha_k c(n-k)$$

- équations de partitions

$$c(n) = a \times c\left(\frac{n}{b}\right) + f(n)$$

Problème : comment résudre ces équations pour en déduire le comportement asymtotique ?

Résolution : "à la main", en appliquant un théorème

Résolution “à la main” - méthode de substitution

$$\begin{cases} c(0) & = & 1 \\ c(n) & = & 2 \times c(n-1) + 1 \end{cases}$$

Résultat :

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

Se souvenir que :

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1}, x \neq 1$$

Résolution “à la main” - méthode de l’arbre

$$\begin{cases} c(1) & = & 1 \\ c(n) & = & 2 \times c(n/2) + 1 \end{cases}$$

Résultat (en supposant que n est une puissance de 2) :

$$c(n) = 2^n + 2^{\log n - 1} = \Theta(2^n)$$

Se souvenir que : le nombre de noeuds d’un arbre binaire complet à une profondeur p est 2^p

Les divisions entières par 2

Il arrive fréquemment que les équations de récurrences soient du type :

$$c(n) = 2c\left(\frac{n}{2}\right) + 1$$

Théorème

Soit $c(n)$ une fonction croissante de IN vers IR. Si, pour tout n assez grand de la forme $n = 2^p$, $c(n) = \mathcal{O}(n^{\alpha}(\log n)^{\beta})$ avec $\alpha \geq 0$ et $\beta \geq 0$, alors l’égalité reste vraie pour tout n assez grand.

Cela signifie que si on résout l’équation pour des n de la forme 2^p alors la solution sera valable pour n’importe quel n, pourvu qu’il soit grand.

Récurrences linéaires

Definition

Une équation de récurrence est dite linéaire d’ordre k si chaque terme s’exprime comme la combinaison linéaire des k termes qui le précèdent plus une certaine fonction de n :

$$c(n) = \alpha_1 c(n-1) + \alpha_2 c(n-2) + \cdots + \alpha_k c(n-k) + f(n)$$

Definition

Une équation de récurrence est dite linéaire d’ordre k est dite sans second membre si f(n) = 0.

Résolution des équations linéaires sans second membre

Soit une équation de récurrence linéaire d'ordre k sans second membre :

$$c(n) = \alpha_1 c(n-1) + \alpha_2 c(n-2) + \cdots + \alpha_k c(n-k)$$

On y associe son polynôme caractéristique :

$$P(x) = x^k - \alpha_1 x^{k-1} - \alpha_2 x^{k-2} + \cdots - \alpha_k$$

La résolution de ce polynôme donne $k' \leq k$ racines r_i de multiplicité w_i.

La solution de l'équation est alors :

$$c(n) = Q_1(n) r_1^n + Q_2(n) r_2^n + \cdots + Q_{k'}(n) r_{k'}^n$$

où les $Q_i(n)$ sont des polynômes de degré $w_i - 1$.
Les coefficients de chaque Q_i sont déterminés grâce aux k premiers termes.

Exemple

$$\begin{cases} c(0) & = & 2 \\ c(1) & = & 2 \\ c(n) & = & c(n-1) + c(n-2) \end{cases}$$

$$P(x) = x^2 - x - 1$$

$$r_1 = \frac{1+\sqrt{5}}{2}, r_2 = \frac{1-\sqrt{5}}{2}$$

$$c(n) = a r_1^n + b r_2^n$$

$$a = \frac{\sqrt{5}+1}{\sqrt{5}}, b = \frac{\sqrt{5}-1}{\sqrt{5}}$$

$$c(n) = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

Résolution des équations linéaires d'ordre 1

La solution d'une équation linéaire d'ordre 1 de la forme :

$$c(n) = a \times c(n-1) + f(n)$$

est :

$$c(n) = a^n \left(c(0) + \sum_{i=1}^{n} \frac{f(i)}{a^i} \right)$$

Exemple

$$\begin{cases} c(0) & = & 1 \\ c(n) & = & 2 \times c(n-1) + 1 \end{cases}$$

$$c(n) = 2^n (1 + \sum_{i=1}^{n} \frac{1}{2^i}) = 2^{n+1} - 1$$

Utiliser l'équation ci-dessous avec $x = \frac{1}{2}$:

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}, x \neq 1$$

Résolution des équations linéaires d'ordre 2

A une équation linéaire d'ordre 2 de la forme :

$$c(n) = a \times c(n-1) + b \times c(n-2) + f(n) \qquad (1)$$

on associe l'équation homogène :

$$c'(n) = a \times c'(n-1) + b \times c'(n-2)$$

puis on résout $c'(n)$ qui est une équation linéaire sans second membre.

On sait ensuite que $c(n) = c'(n) + g(n)$ où $g(n)$ est une solution particulière de l'équation 1.

C-à-d: $g(n) = a \times g(n-1) + b \times g(n-2) + f(n)$

Exemple

$$\begin{cases} c(0) &= 1 \\ c(1) &= 1 \\ c(n) &= c(n-1) + c(n-2) + 1 \end{cases}$$

Equation homogène associée :

$$c'(n) = c'(n-1) + c'(n-2)$$

On retrouve (les conditions initiales ont changé, a' et b' sont différents de a et b) :

$$c'(n) = a' r_1^n + b' r_2^n$$

On sait que :

$$c(n) = c'(n) + g(n)$$

Et que $g(n)$ est solution particulière, cad : $g(n) = g(n-1) + c(n-2) + 1$, on trouve que $g(n) = \alpha$ fonctionne :

$$\alpha = \alpha + \alpha + 1$$

Et donc $\alpha = -1$.

Théorème (Théorème général)

Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $c(n)$ définie pour les entiers non négatifs par la récurrence

$$c(n) = a \times c(n/b) + f(n),$$

où l'on interprète n/b comme étant $\lfloor n, b \rfloor$ ou $\lceil n/b \rceil$. $c(n)$ peut alors être bornée asymtotiquement de la façon suivante.

1. *si $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ pour une certaine constante $\varepsilon > 0$, alors*

$$c(n) = \Theta(n^{\log_b a}).$$

2. *si $f(n) = \Theta(n^{\log_b a})$, alors*

$$c(n) = \Theta(n^{\log_b a} \log_2 n).$$

3. *si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ pour une certaine constante $\varepsilon > 0$, et si $a \times f(n/b) \leq k \times f(n)$ pour une certaine constante $k < 1$ et pour n suffisamment grand, alors*

$$c(n) = \Theta(f(n)).$$

Exemples

1. $$\begin{cases} c(0) &= 1 \\ c(n) &= c(\lfloor \frac{n}{2} \rfloor) + 1 \end{cases}$$
   $a = 1, b = 2, f(n) = 1$, $\log_b a = \log_2 1 = 0$, $f(n) = 1 = \Theta(n^0)$, cas 2, $c(n) = \Theta(\log n)$
2. $$\begin{cases} c(0) &= 1 \\ c(n) &= 9 \times c\left(\lceil \frac{n}{3} \rceil\right) + n \end{cases}$$
   $a = 9, b = 3, f(n) = n$, $\log_b a = \log_3 9 = 2$, $f(n) = n = \mathcal{O}(n^{2-\varepsilon})$, cas 1, $c(n) = \Theta(n^2)$
3. $$\begin{cases} c(0) &= 1 \\ c(n) &= 3 \times c\left(\lfloor \frac{n}{4} \rfloor\right) + n \log n \end{cases}$$
   $a = 3, b = 4, f(n) = n \log n$, $\log_b a = \log_4 3 \approx 0.79$, $f(n) = \Omega(n^{0.79+\varepsilon})$, cas 3, $af(\frac{n}{b}) = 3\frac{n}{4} \log \frac{n}{4} \leq \frac{3}{4} n \log n$, on est bien dans le cas 3, $c(n) = \Theta(n \log n)$

Conclusion

- Il faut savoir reconnaître le type d'équation :
  - de partiition
    - bien identifier et *prouver* le cas
  - linéaire
    - avec / sans second membre
    - attention aux erreurs de calcul
- Utiliser la méthode de substition/arbre pour les cas "simples", mais attention aux erreurs de calcul dans la hauteur de l'arbre, sur les termes additionnels, etc.