Diviser pour Régner, I

I. Description de la méthode

Jusqu'à présent, nous avons vu des fonctions récursives qui, exécutées sur une entrée de taille n, font des appels récursifs sur des entrées de taille $n-1$ (un entier qui décroit, une liste dont on a enlevé l'élément en tête, etc.).

Ici, nous allons étudier une autre façon d'écrire des fonctions récursives, où pour un entrée de taille n, les appels récursifs sont faits avec des entrées de taille $n/2$.

On peut en général décomposer une telle fonction en plusieurs phases :

1. décomposition de l'entrée de taille n en sous-problèmes de taille $n/2$;
2. appels récursifs sur les sous-problèmes ;
3. recombinaison des résultats pour obtenir le résultat sur l'entrée de taille n.

Aux phases 1 et 2, on pourra avoir des traîtements supplémentaires.

Remarque Dans la suite, on découpe le problème initial en sous-problème de taille moitié. On pourra rencontrer des cas où les sous-problèmes font le tiers, le quart, le cinquième, etc. du problème initial. L'idée générale et les méthodes d'analyse s'adaptent facilement.

1. Exemple : Exponentiation rapide

On a vu que l'algorithme d'exponentiation « naïf » permet de calculer par récurrence a^n en utilisant la relation $a^{n+1} = a \times a^n$:

```
let rec expo_naive a n =
    if n = 0 then 1 else a * expo_naive a (n - 1) ;;
```

D'après l'étude précédente, la complexité c_n pour calculer a^n, on a $c_{n+1} = c_n + d_n$ avec $d_n = \Theta(1)$, d'où $c_n = \Theta(n)$.

Plutôt que de soustraire 1, il est plus intéressant de couper en deux. On obtient l'algorithme d'exponentiation rapide :

```
let rec expo a n =
  if n = 0 then 1
  else if n = 1 then a
  else
    let r = expo (a * a) (n / 2) in
    if n mod 2 = 0 then r else a * r
```

```
;;
```

On a maintenant $c_n = c_{\lfloor \frac{n}{2} \rfloor} + d_n$ avec $d_n = \Theta(1)$, d'où $c_n = \Theta(\log_2 n)$.

2. Exemple : Buy-and-Sell

On suppose que l'on a un tableau de nombres T de longueur n, et on cherche à déterminer

$$\max\Big(\{0\} \cup \{T(j) - T(i) \mid 0 \leq i < j < n\}\Big)$$

On peut voir le tableau T comme donnant la valeur au cours du temps, et on cherche à déterminer le gain maximum en effectuant un achat (au temps i) suivi d'une revente (au temps $j > i$).

Une approche naïve se fait à l'aide de deux boucles imbriquées :

```
let buy_and_sell t =
  let n = Array.length t
  and m = 0 in
  for i = 0 to n - 2 do
    for j = i + 1 to n - 1 do
      if t.(j) - t.(i) > !m then m := t.(j) - t.(i)
    done
  done;
  !m
;;
```

Une approche « diviser pour régner » peut être décrite ainsi : l'instant d'achat et l'instant de vente peuvent être

⋆ tous les deux dans la première moitié,
⋆ tous les deux dans la deuxième moitié,
⋆ un de chaque côté.

On en déduit la fonction suivant :

```
let buy_and_sell t =
  let n = Array.length t in
  let rec aux debut fin =
    if fin <= debut + 1 then
      0
    else
      let milieu = (debut + fin) / 2 in
      let g1 = aux debut milieu
      and g2 = aux milieu fin
      and g3 =
        max_tableau t milieu fin
        - min_tableau t debut milieu
```

```
      in
      max g1 (max g2 g3)
  in
  aux 0 n
;;
```

Est-ce plus efficace ? Ici, en fonction de la longueur n tu tableau, la complexité temporelle vérifie la relation :

$$c_n = c_{\lfloor \frac{n}{2} \rfloor} + c_{\lceil \frac{n}{2} \rceil} + \Theta(n)$$

Nous allons étudier ce genre de relation.

II. Méthode générale de calcul de coût

On est dans le cadre où l'on a des suites positives (c_n) et (d_n) telles que

$$c_1 = d_1 \quad \forall n \geq 2, c_n = a c_{\lfloor \frac{n}{2} \rfloor} + b c_{\lceil \frac{n}{2} \rceil} + d_n$$

pour $a, b \in \mathbf{N}$ tels que $a + b \geq 1$.

1. Quelques propriétés

Proposition 1 - Relations de comparaison

Si l'on a deux couples de suites $(c_n), (d_n)$ et $(c'_n), (d'_n)$ qui vérifient la relation précédente (pour les mêmes a et b), et si $d_n = \Theta(d'_n)$ (resp. O, Ω), alors $c_n = \Theta(c'_n)$ (resp. O, Ω).

Preuve
Cela découle directement de la linéarité de la relation. Ainsi, si $d_n \leq \alpha d'_n$ et, par hypothèse de récurrence, $c_{\lfloor \frac{n}{2} \rfloor} \leq \alpha c'_{\lfloor \frac{n}{2} \rfloor}$ et idem en $\lceil \frac{n}{2} \rceil$, alors

$$c_n = a c_{\lfloor \frac{n}{2} \rfloor} + b c_{\lfloor \frac{n}{2} \rfloor} + d_n \leq \alpha \left(a c'_{\lfloor \frac{n}{2} \rfloor} + b c'_{\lfloor \frac{n}{2} \rfloor} + d'_n \right) = \alpha c'_n$$

Cela montre que les raisonnements à base de Θ et autres restent corrects avec ce type de relation.

Proposition 2 - Croissance

Si (d_n) est croissante, alors (c_n) est croissante.

Preuve

On procède par récurrence forte. Pour $n = 2$,

$$c_2 = (a+b)c_1 + d_2 \geq c_1$$

Pour $n \geq 3$, on a :

$$c_{n+1} = a\,c_{\lfloor \frac{n+1}{2} \rfloor} + b\,c_{\lceil \frac{n+1}{2} \rceil} + d_{n+1} \geq a\,c_{\lfloor \frac{n}{2} \rfloor} + b\,c_{\lceil \frac{n}{2} \rceil} + d_n = c_n$$

Notons que si la suite (d_n) n'est pas croissante, il suffit de raisonner avec une suite croissante (d'_n) telle que $d_n = \Theta(d'_n)$.

Dans ce cas, pour tout $n \geq 2$, on a $2^{\lfloor \log_2 n \rfloor} \leq n \leq 2^{\lfloor \log_2 n \rfloor} + 1$ d'où

$$c_{2^{\lfloor \log_2 n \rfloor}} \leq c_n \leq c_{2^{\lfloor \log_2 n \rfloor}+1}$$

On peut donc se restreindre aux puissances de 2. On a :

$$\begin{aligned}
c_1 &= d_1 \\
c_2 &= (a+b)c_1 + d_2 \\
&= (a+b)d_1 + d_2 \\
c_4 &= (a+b)c_2 + d_4 \\
&= (a+b)^2 d_1 + (a+b)d_2 + d_4
\end{aligned}$$

et, plus généralement :

$$c_{2^r} = \sum_{k=0}^{r} (a+b)^k d_{2^{r-k}} = (a+b)^r \sum_{k=0}^{r} \frac{d_{2^k}}{(a+b)^k}$$

Remarques

- ⋆ On le voit, seule compte la valeur $a+b$, et on n'a pas besoin de s'embêter avec la distinction $\lfloor \frac{n}{2} \rfloor$ et $\lceil \frac{n}{2} \rceil$, puisque l'on ramène notre étude aux puissances de 2.
- ⋆ Parfois, les valeurs de a et de b peuvent varier selon la situation, mais la valeur de $a+b$ reste constante.

2. Résolution dans quelques cas typiques

On rappelle que l'on peut raisonner à Θ près.

Si pour tout n, $d_n = n^\alpha$, alors

$$\sum_{k=0}^{r} \frac{d_{2^k}}{(a+b)^k} = \sum_{k=0}^{r} \frac{\left(2^k\right)^\alpha}{(a+b)^k} = \sum_{k=0}^{r} \left(\frac{2^\alpha}{a+b}\right)^k$$

Ainsi,

- ⋆ si $\frac{2^\alpha}{a+b} < 1$, autrement dit si $\log_2(a+b) > \alpha$, alors la somme reste bornée et on a alors on a $c_{2^r} = \Theta\big((a+b)^r\big)$;
- ⋆ si $\frac{2^\alpha}{a+b} = 1$, autrement dit si $\log_2(a+b) = \alpha$ alors la somme est égale à $n+1$ et on a $c_{2^r} = \Theta\big(r(a+b)^r\big)$;
- ⋆ si $\frac{2^\alpha}{a+b} > 1$, autrement dit si $\log_2(a+b) < \alpha$, alors la somme est de l'ordre de $\left(\frac{2^\alpha}{a+b}\right)^r$ et donc $c_{2^r} = \Theta\big(2^{\alpha r}\big)$.

Pour le cas général, on utilise $r = \log_2 n$ et on remets les Θ, d'où :

Proposition 3

Si $c_n = \Theta\big(n^\alpha\big)$, alors :

- ⋆ si $\log_2(a+b) < \alpha$, alors $u_n = \Theta\big(n^\alpha\big)$,
- ⋆ si $\log_2(a+b) = \alpha$, alors $u_n = \Theta\big(n^\alpha \log n\big)$,
- ⋆ si $\log_2(a+b) > \alpha$, alors $u_n = \Theta\big(n^{\log_2(a+b)}\big)$,

3. Reprise des exemples précédents

3.1. Exponentiation rapide

On avait la relation

$$\forall\, n \geq 2,\ c_n = c_{\lfloor \frac{n}{2} \rfloor} + \Theta(1)$$

Avec les notations précédentes, on a donc $a = 1$, $b = 0$ (et donc $a+b = 1$), et $d_n = \Theta(n^\alpha)$ pour $\alpha = 0$.

On a alors $\log_2(a+b) = 0 = \alpha$ d'où $c_n = \Theta(n^\alpha \log_2 n) = \Theta(\log_2 n)$.

3.2. Buy and Sell

On a cette fois

$$\forall\, n \geq 2,\ c_n = c_{\lfloor \frac{n}{2} \rfloor} + c_{\lceil \frac{n}{2} \rceil} + \Theta(n)$$

soit $a+b = 2$, $d_n = \Theta(n^\alpha)$ pour $\alpha = 1$.

On a, cette fois encore, $\log_2(a+b) = 1 = \alpha$ d'où $c_n = \Theta(n \log_2 n)$. Notons que c'est bien mieux que l'algorithme naïf qui était en $\Theta\big(n^2\big)$.