Étude de la complexité du tri fusion

1 Principe du tri fusion

Le tri fusion est un algorithme récursif permettant de trier une liste. Le principe est le suivant : on partitionne la liste en deux parties de longueurs égales (à une unité près), puis on trie récursivement les deux parties avant de les fusionner en entrelaçant leurs éléments de manière à respecter l'ordre.

Voici la fonction de partition :

```
let rec partition (l : 'a list) : 'a list * 'a list =
  match l with
  | [] -> [], []
  | [t] -> [t], []
  | a :: b :: q ->
     let (l1, l2) = partition q in
     (a :: l1, b :: l2)
```

Voici la fonction de fusion :

```
let rec fusion (l1 : 'a list) (l2 : 'a list) : 'a list =
  match l1, l2 with
  | [], _ -> l2
  | _, [] -> l1
  | t1 :: q1, t2 :: _ when t1 <= t2 -> t1 :: fusion q1 l2
  | _, t2 :: q2 -> t2 :: fusion l1 q2
```

Voici enfin le code de la fonction de tri :

```
let rec tri_fusion (l : 'a list) : 'a list =
  match l with
  | [] | [_] -> l
  | _ ->
     let (l1, l2) = partition l in
     fusion (tri_fusion l1) (tri_fusion l2)
```

2 Étude de la complexité

La fonction `partition` est de complexité linéaire, puisqu'elle effectue un nombre constant d'opérations pour chaque élément de la liste en paramètre.

La fonction `fusion` est de complexité $\mathcal{O}(n_1 + n_2)$, où n_1 (resp. n_2) est la longueur de la liste `l1` (resp. `l2`) car, dans le pire cas, la fusion respecte une stricte alternance entre les valeurs des deux listes : on parcourt donc tous les éléments des deux listes.

Dans le cadre du tri fusion, si $n > 1$ est la longueur de la liste en paramètre, la fonction `partition` renvoie une liste `l1` de longueur $\left\lceil \frac{n}{2} \right\rceil$ et une liste `l2` de longueur $\left\lfloor \frac{n}{2} \right\rfloor$. La fusion des versions triées de ces deux listes se fera donc en temps $\mathcal{O}\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right) = \mathcal{O}(n)$.

La complexité $C(n)$ de la fonction `tri_fusion` vérifie donc la relation de récurrence :

$$\forall n > 1. C(n) \leqslant C\left(\left\lceil \frac{n}{2} \right\rceil\right) + C\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + an,$$

où a est une constante strictement positive couvrant le $\mathcal{O}(n)$ de la partition et de la fusion.

On veut montrer que $C(n) = \mathcal{O}(n \log_2(n))$, i.e. qu'il existe $c > 0$ et $N \in \mathbb{N}$ tels que pour tout $n \geqslant N$, $C(n) \leqslant cn \log_2(n)$. On procède par analyse puis synthèse, i.e. on suppose d'abord l'existence de c et de N et on cherche à déterminer des contraintes sur leur valeur afin de pouvoir dans un second temps démontrer que ces constantes existent en effet.

On note tout d'abord que $N > 1$, car la complexité est strictement positive (or $\log_2(1) = 0$). On note également les contraintes suivantes sur c :

$$\left\{ \begin{array}{ccc} c & \geqslant & \frac{C(2)}{2} \\ c & \geqslant & \frac{C(3)}{3 \log_2(3)} \end{array} \right. .$$

Enfin, supposons $n > 3$. Nous avons donc :

$$\begin{array}{rcl} C(n) & \leqslant & C\left(\left\lceil \frac{n}{2} \right\rceil\right) + C\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + an \\ & \leqslant & c\left\lceil \frac{n}{2} \right\rceil \log_2\left(\left\lceil \frac{n}{2} \right\rceil\right) + c\left\lfloor \frac{n}{2} \right\rfloor \log_2\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + an \\ & \leqslant & c\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right) \log_2\left(\left\lceil \frac{n}{2} \right\rceil\right) + an \\ & \leqslant & cn \log_2\left(\frac{n+1}{2}\right) + an \\ & \leqslant & cn \log_2(n+1) + (a-c)n \end{array}$$

Or la fonction $\log_2$ est concave, donc sa courbe représentative est située sous toutes ses tangentes, en particulier celle au point d'abscisse n qui est d'équation $y = \log_2(n) + \frac{x-n}{n}$. En $n+1$, cela donne :

$$\log_2(n+1) \leqslant \log_2(n) + \frac{1}{n}.$$

On a alors :

$$C(n) \leqslant cn \log_2(n) + c + (a-c)n,$$

ce qui impose $c + (a-c)n \leqslant 0$ pour tout $n > 3$. Or, dès que $c \geqslant 2a$, on a également $c \geqslant \frac{an}{n-1}$, i.e. $c + (a-c)n \leqslant 0$.

En conclusion, en choisissant $c = \max\left(\frac{C(2)}{2}, \frac{C(3)}{3 \log_2(3)}, 2a\right)$ et $N = 2$, on peut démontrer par récurrence forte que pour tout $n \geqslant N$, on a $C(n) \leqslant cn \log_2(n)$.