Analyse du tri rapide

Manon Ruffini

On considère le problème de tri suivant :
Entrée : Un tableau T contenant n nombres
Sortie : Le tableau T trié

Étape 1

Présentation de l'algorithme de tri rapide

On va utiliser l'algorithme du tri rapide, qui est basé sur le paradigme diviser pour régner : Pour trier un sous-tableau $T[p, r]$, on utilise :

Diviser On choisit un élément de $T[p, r]$ qu'on notera *pivot*. On partitionne $T[p, r]$ en deux sous-tableaux $T[p, q]$ et $T[q + 1, r]$ tels que :
- $T[q] = pivot$
- Les éléments de $T[p, q]$ sont $\leqslant pivot$
- Les éléments de $T[q + 1, r]$ sont $> pivot$.

Régner On trie récursivement $T[p, q]$ et $T[q + 1, r]$

Combiner Par construction, il n'y a rien à faire.

Remarque : Ce tri est en place mais n'est pas stable.

Par exemple, on peut choisir $pivot = T[r]$, ce qui conduit aux algorithmes suivants :

Algorithme 1 : `Tri-rapide`(T, p, r)

Entrées : Un tableau T à n éléments ; deux indices p, r de T
si $p < r$ **alors**
    $q \leftarrow$ `partition`(T, p, r);
    `Tri-rapide`$(T, p, q - 1)$;
    `Tri-rapide`$(T, q + 1, r)$
fin

Algorithme 2 : `Partition`(T, p, r)

$pivot \leftarrow T[r]$;
$i \leftarrow p - 1$;
pour $p \leqslant j \leqslant r - 1$ **faire**
    si $T[j] \leqslant pivot$ **alors**
        $i \leftarrow i + 1$;
        $T[i] \leftrightarrow T[j]$;
    fin
    $T[i + 1] \leftrightarrow T[r]$
fin
retourner $pivot$

Étape 2

Correction de l'algorithme

La correction de `Tri-rapide` repose sur la correction de `partition`. Pour cela, on peut montrer l'invariant de boucle suivant [1] :

— Si $p \leqslant k \leqslant i$, alors $T[k] \leqslant pivot$
— Si $i < k < j$, alors $T[k] > pivot$
— Si $k = r$, alors $T[k] = pivot$

On peut aussi remarquer que ce tri est en place mais n'est pas stable.

Étape 3

Analyse de la complexité dans le cas le plus défavorable.

On note $C(n)$ la complexité dans le cas le plus défavorable, pour une entrée de taille n. (Rappel : on compte le nombre de comparaisons). On a :

$$C(n) = \max_{0 \leqslant q \leqslant n-1} (C(q) + C(n-q-1)) + \Theta(n)$$

Par récurrence forte sur n, montrons qu'il existe une constante k telle que $\forall n \in \mathbb{N}, C(n) \leqslant kn^2$

$n = 1$: $C(1) = C(0) + C(0) + \Theta(1) \leqslant k \cdot 1$, où k est une constante réelle

Soit $n \in \mathbb{N}^*$. Supposons l'hypothèse vraie pour tout $q < n$

$$\begin{aligned} C(n) &= \max_{0 \leqslant q \leqslant n-1} (C(q) + C(n-q-1)) + \Theta(n) \\ &\leqslant \max_{0 \leqslant q \leqslant n-1} (kq^2 + k(n-q-1)^2) + \Theta(n) \\ &= k \max_{0 \leqslant q \leqslant n-1} (q^2 + (n-q-1)^2) + \Theta(n) \end{aligned}$$

Or, si on étudie les variations de la fonction $f : q \mapsto q^2 + (n-q-1)^2$, on s'aperçoit qu'elle est toujours inférieure à $(n-1)^2$. Donc :

$$C(n) \leqslant k(n-1)^2 + \Theta(n) = kn^2 - k(2n-1) + \Theta(n)$$

Quitte à prendre k plus grande, on peut supposer que $\forall n \in \mathbb{N}^*, k(2n+1) \geqslant \Theta(n)$.

On peut montrer que ce cas est atteint : Supposons qu'à chaque appel récursif, `partition` crée un tableau à 0 élément. Alors :

$$\begin{aligned} C(n) &= C(n-1) + C(0) + \Theta(n) \\ &= C(n-1) + \Theta(n) \\ &= C(n-2) + \Theta(n-1) + \Theta(n) \\ &= \dots \\ &= \Theta(n^2) \end{aligned}$$

1. Pour gagner du temps, on peut se contenter de faire le dessin :

p	i	$i+1$	j		r
$\leqslant pivot$		$> pivot$			$pivot$

Étape 4

Complexité dans le cas le plus favorable (Soit C_f)

C'est le cas d'un partitionnement équilibré ; les tableaux sont de taille $\lfloor n/2 \rfloor$ et $\lceil n/2 \rceil - 1$, ie de taille $\leqslant n/2$. D'où :

$$C_f(n) \leqslant 2C_f\left(\frac{n}{2}\right) + \Theta(n)$$

Donc

$$C_f(n) = O(n \log n)$$

Étape 5

Complexité en moyenne (soit c)

On suppose que les n clés sont distinctes. Soit Q la variable aléatoire qui correspond à la position de *pivot* à la fin de `Partition`. Le tableau est dans un ordre a priori aléatoire, donc Q correspond à une loi uniforme (toutes les positions sont équiprobables). On remarque que `Partition` nécessite $n-1$ comparaisons dans le cas d'une entrée de taille n. Ainsi, pour `Tri-rapide` :

$$\begin{aligned}
c(n) &= 1 + n - 1 + \mathbb{E}\left(c(Q-1) + c(n-Q-1)\right) \\
&= n + \sum_{q=1}^{n} \mathbb{P}(Q=q)\left(c(q-1) + c(n-q-1)\right) \\
&= n + \frac{1}{n}\sum_{q=1}^{n}\left(c(q-1) + c(n-q-1)\right) \\
&= n + \frac{2}{n}\sum_{q=1}^{n}\left(c(q-1)\right) \qquad \left(\text{car } \sum_{q=1}^{n} c(n-q-1) = \sum_{q'=1}^{n} c(q'-1)\right)
\end{aligned}$$

Ainsi,

$$\forall n \geqslant 3, \begin{cases} nc(n) = n^2 + 2\sum_{q=1}^{n} c(q-1) \\ (n-1)c(n-1) = (n-1)^2 \sum_{q=1}^{n-1} c(q-1) \end{cases}$$

En soustrayant ces deux équations, on obtient :

$$nc(n) - (n-1)c(n-1) = n^2 - (n-1)^2 + 2c(n-1)$$

Et donc :

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

En divisant par $n(n+1)$:

$$\begin{aligned}
\frac{c(n)}{n+1} &= \frac{c(n-1)}{n} + \frac{2n-1}{n(n+1)} \\
&= \sum_{k=1}^{n} \frac{2k-1}{k(k+1)} + c(0) \\
&\sim \sum_{k=1}^{n} \frac{2}{k+1} + c(0) \\
&\text{(par équivalence des sommes partielles pour les séries à termes positifs divergentes)} \\
&= 2\sum_{k=1}^{n+1} \frac{1}{k}
\end{aligned}$$

Finalement, si H_n désigne la série harmonique,

$$c(n) \sim 2(n+1)H_n \sim 2(n+1)\log n$$

Ainsi,

$$c(n) = O(n \log n)$$

Références

[1] Thomas H. Cormen, *Algorithmique*. Dunod, 3è édition, 2010.

[2] C. Froidevaux, M-C. Gaudel, M. Soria, *Types de données et algorithmes*. McGraw-Hill, 1990.