Tri polyphasé

Mathias Millet

2014

Introduction

L'algorithme du tri polyphasé est un algorithme de tri adapté de l'algorithme de tri fusion. Il permet d'obtenir de bonnes performances dans le cas d'utilisation de bandes magnétiques en tant que mémoires externes.

Nous utiliserons ici la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$, définie par $F_0 = 0$, $F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$ pour $n \geq 0$.

Modèle de calcul

On se place dans le modèle suivant :

- Un processeur avec une mémoire centrale de taille M
- Trois bandes magnétiques pour stoquer les données, chacune équipée d'une tête de lecture qui peut lire ou écrire une case mémoire à la fois. On supposera les bandes de longueur non bornée.

On dispose donc des opérations élémentaires supplémentaires suivantes :

- `var` $\leftarrow$ `lire`$_{\texttt{i}}$: donne à la variable `var` la valeur lue sur la bande i ;
- `écrire`$_{\texttt{i}}$(`var`) : écris la valeur de `var` sur la bande i ;
- `avancer`$_{\texttt{i}}$: avance d'une case mémoire sur la bande i ;
- `reculer`$_{\texttt{i}}$: recule d'une case mémoire sur la bande i ;

Complexité Entrées/Sorties. Dans ce modèle de calcul, on peut définir la complexité entrées/sorties d'un algorithme comme le nombre d'opérations `avancer` et `reculer` effectuées pendant une exécution.

Tri polyphasé

Définition (Monotonie) Une monotonie est un tableau trié par ordre croissant, placé sur une bande magnétique.

Définition (Fusion) La fusion de deux monotonies M_1 et M_2, de taille m_1 et m_2 respectivement, est la création d'une nouvelle monotonie, contenant exactement la réunion des éléments de M_1 et M_2. L'algorithme de la fusion est directement adapté de celui du tri fusion. Pour être efficace en terme d'entrées/sorties, chaque monotonie doit se trouver sur une bande différente, et les têtes de lecture doivent être placées initialement au début des monotonies. La complexité de l'opération de fusion est alors en $O(m_1 + m_2)$.

On suppose en général que les données lues sont effacées au fur et à mesure.

Algorithme du tri polyphasé

On suppose que le tableau à trier se trouve sur une seule bande. La taille de ce tableau est N. On effectue les étapes suivantes :

- On trouve n tel que $\frac{N}{F_n} > M \geq \frac{N}{F_{n+1}}$.
- On découpe les données en paquets de taille $t = \frac{N}{F_{n+1}}$ (on obtient donc F_{n+1} paquets), qu'on trie en mémoire, puis qu'on replace sur les autres bandes, obtenant ainsi des monotonies. On en place F_{n-1} sur la première bande, F_n sur la deuxième.
- On fusionne alors les paquets comme expliqué ci-dessous.

Fusion des monotonies

Première étape La première étape se déroule de la manière suivante :

1. La première bande contient F_n monotonies de taille t ;
2. La deuxième bande contien F_{n-1} monotonies de taille t ;
3. La troisième bande est vide.

On fusionne alors 2 à 2 les monotonies des deux premières bandes, en plaçant les nouvelles monotonies obtenues sur la troisième bande. On effectue ainsi F_{n-1} fusions, de complexité $2t$ chacune.

On obtient à la fin des fusions :

1. La première bande contient F_{n-2} monotonies de taille $t \cdot F_{i+1}$;
2. La deuxième bande est vide.
3. La troisième bande contient F_{n-1} monotonies de taille $t \cdot F_{i+2}$.

$i^{\text{ème}}$ étape Au début de la $i^{\text{ème}}$ étape, on a la situation suivante :

1. Une bande avec F_{n-i+1} monotonies de taille $t \cdot F_i$;
2. Une bande avec F_{n-i} monotonies de taille $t \cdot F_{i+1}$;
3. Une bande vide.

On fusionne alors 2 à 2 les monotonies des deux premières bandes, en plaçant les nouvelles monotonies obtenues sur la troisième bande. On effectue ainsi F_{n-i} fusions, de complexité $t \cdot F_i + t \cdot F_{i+1}$ chacune.

On obtient à la fin des fusions :

1. Une bande avec F_{n-i-1} monotonies de taille $t \cdot F_{i+1}$;
2. Une bande vide ;
3. Une bande avec F_{n-i} monotonies de taille $t \cdot F_{i+2}$.

Arrêt de l'algorithme On obtient une unique monotonie à la fin de la $(n-1)^{\text{ème}}$ étape. Les données sont alors triées !

Calcul de complexité

Théorème : La complexité E/S de l'algorithme du tri polyphasé est en $O\left(N \log\left(\frac{N}{M}\right)\right)$.

Preuve : Le tri et la formation des monotonies initiales sont faits en complexité linéaire. Calculons maintenant la complexité des fusions de monotonies. La fusion s'exécute en $n-1$ étapes, la $i^{\text{ème}}$ étape impliquant F_{n-i} fusions de complexité $O(t \cdot F_i + t \cdot F_{i+1})$ chacune, d'où une complexité totale en

$$O\left(t \cdot \sum_{i=1}^{n-1} F_{n-i}F_{i+2}\right)$$

. On dispose aussi des deux lemmes suivants :

Lemme 1 : $\displaystyle\sum_{k=0}^{n} F_{n-k}F_k = \frac{n-1}{5}F_n + \frac{2n}{5}F_{n-1}.$

Lemme 2 : $F_n = \dfrac{1}{\sqrt{5}}(\phi^n + \bar{\phi}^n)$ avec $|\phi| > 1$, $|\bar{\phi}| < 1$.

Mettons notre expression sous une forme compatible avec celle du lemme 1 :

$$C = t \cdot \sum_{i=1}^{n-1} F_{n-i}F_{i+2} = t \cdot \sum_{i=1}^{n-1} F_{n+2-(i+2)}F_{i+2} = t \cdot \sum_{k=3}^{n+1} F_{n+2-k}F_k$$

$$= t \cdot \left(\sum_{k=0}^{n+2} F_{n+2-k}F_k \quad - \quad (2 \cdot F_0F_{n+2} + F_1F_{n+1} + F_2F_n)\right)$$

On peut maintenant réécrire

$$C = t \cdot \left(\frac{n+1}{5}F_{n+2} + \frac{2(n+2)}{5}F_{n+1} - F_{n+2}\right) = t \cdot \left(\frac{n-4}{5}F_{n+2} + \frac{2(n+2)}{5}F_{n+1}\right)$$

Or on sait que $t = \dfrac{N}{F_{n+1}}$, on a donc $C = N \cdot \left(\dfrac{n-4}{5}\dfrac{F_{n+2}}{F_{n+1}} + \dfrac{2(n+2)}{5}\right)$

Par définition de $(F_n)_n$, on a $\dfrac{F_{n+2}}{F_{n+1}} = 1 + \dfrac{F_n}{F_{n+1}} = O(1)$, et

$$\log(F_n) = -\frac{1}{2}\log(5) + n\log(\phi) + \log(1 + o(1)) \Longrightarrow n = O(\log(F_n)).$$

On obtient donc que $C = O(N\log(F_n))$.

Finalement, comme $\dfrac{N}{F_n} > M$, $F_n = O(\frac{N}{M})$, et la complexité de l'algorithme est

$$\boxed{O\left(N\log(\frac{N}{M})\right)}$$

□

Preuve du lemme 1 : Montrons par récurrence, pour $n \geq 1$, la propriété

$$\mathcal{P}(n) : \quad \sum_{k=0}^{n} F_{n-k}F_k = \frac{n-1}{5}F_n + \frac{2n}{5}F_{n-1}$$

- $\mathcal{P}(1) : 2F_1F_0 = 0F_1 + \frac{2}{5}F_0 = 0$ est vraie.
- $\mathcal{P}(2) : 2F_2F_0 + F_1^2 = \frac{1}{5}F_2 + \frac{4}{5}F_1 = 1$ est vraie.

- Soit $n \geq 2$, supposons $\mathcal{P}(n)$ et $\mathcal{P}(n-1)$ vraies. On a

$$\begin{aligned}
\sum_{k=0}^{n+1} F_{n+1-k}F_k &= \sum_{k=0}^{n-1} F_{n+1-k}F_k \quad + \quad F_nF_1 + F_{n+1}F_0 = \sum_{k=0}^{n-1}(F_{n-k} + F_{n-1-k})F_k \quad + \quad F_n \\
&= \sum_{k=0}^{n-1} F_{n-k}F_k + \sum_{k=0}^{n-1} F_{n-1-k}F_k \quad + \quad F_n = \sum_{k=0}^{n} F_{n-k}F_k + \sum_{k=0}^{n-1} F_{n-1-k}F_k \quad + \quad F_n \\
&= \frac{n-1}{5}F_n + \frac{2n}{5}F_{n-1} \quad + \quad \frac{n-2}{5}F_{n-1} + \frac{2(n-1)}{5}F_{n-2} \quad + \quad F_n \\
&= \frac{4}{5}F_n + \frac{n}{5}(F_n + F_{n-1}) + \frac{2(n-1)}{5}(F_{n-1} + F_{n-2}) \\
&= \frac{n}{5}F_{n+1} + \frac{2(n-1)}{5}F_n = \frac{4}{5}F_n + \frac{n}{5}F_{n+1} + \frac{2(n+1)}{5}F_n
\end{aligned}$$

  Donc $\mathcal{P}(n+1)$ est vraie.

On a ainsi prouvé par récurrence que, pour tout $n \geq 1$, et la propriété $\mathcal{P}(n)$ est vraie. □

Preuve du lemme 2 : D'après le théorème sur les suites récurrentes définies par une équation linéaires, les suites $(u_n)_n$ qui vérifient $u_{n+2} = u_{n+1} + u_n$ pour $n \geq 0$ sont de la forme $u_n = \lambda\phi^n + \mu\bar{\phi}^n$, où $\phi = \frac{1+\sqrt{5}}{2}$ et $\bar{\phi} = \frac{1-\sqrt{5}}{2}$ sont les deux solutions de l'équation $X^2 - X - 1 = 0$.

Les conditions initiales nous permettent de poser : $F_0 = \lambda + \mu = 0$ et $F_1 = \lambda\phi + \mu\bar{\phi} = \frac{1}{2}((\lambda+\mu) + \sqrt{5}(\lambda-\mu) = 1$, d'où $\lambda = -\mu = \frac{1}{\sqrt{5}}$, et $F_n = \frac{1}{\sqrt{5}}(\phi^n + \bar{\phi}^n)$. □

Références

- Christine Froidevaux, Marie-Claude Gaudel, Michèle Soria, *Types de données et algorithmes.*

  Directement tiré de ce livre

- Donald Knuth, *The Art of Computer Programming, Volume 3, Sorting and Searching*

  Traite le tri polyphasé, je n'ai pas regardé en détails.