Annexe A

Complexité amortie

Hugo Salou MPI*

Dernière mise à jour le 7 février 2023

Table des matières

Avec une fonction $\mathrm{PA}(n)$ ayant une complexité en $\mathcal{O}(f(n))$, on considère le problème ci-dessous.

```
for i = 0 to n - 1 do
  PA(i)
done
```

Cet algorithme a une complexité en $\mathcal{O}(nf(n))$. Mais, parfois, cette complexité est trop approximative : parfois, des sommes mathématiques se compensent :

$$\sum_{i=0}^{n-1} 2^i = 2^n \neq \cancel{\mathcal{O}}(n2^n).$$

On considère le problème ci-dessous.

```
for (int i = 0; i < n; i = i + 1) (
  // calcul qui ne coute pas cher
}
```

Le calcul `i = i + 1` est parfois plus coûteux que le calcul dans la boucle. Par exemple, un algorithme permettant de faire ce calcul est celui ci-dessous.

Algorithme 1 Calcul de $n+1$ avec un tableau de *bit*s

Entrée un entier n, représenté sous la forme d'un tableau de *bit* T (où les *bit*s de poids forts sont à droite

```
I ← len(T) − 1
tant que T[I] = 1 faire
    T[I] ← 0
    I ← I − 1
T[I] ← 1
```

Avec un tel algorithme, on a une complexité, dans le pire des cas, en $\Theta(\log_2 n)$. Ainsi, en modifiant le code, on peut avoir une complexité importante.

```
// n est une valeur donnee par l'utilisateur
for (int i = 0; i < 2^n; i = i + 1) (
  // calcul qui ne coute pas cher
}
```

La complexité de cet algorithme, que l'on nommera $\mathcal{A}$ dans la suite, est en $\mathcal{O}(\mathtt{n}2^{\mathtt{n}})$, car le `i = i + 1` coûte, au pire des cas, `n`. En réalisant des mesures, et en graphant le temps de cet algorithme divisé par $2^{\mathtt{n}}$, on remarque que ce le ratio n'est pas une droite de coefficient directeur `n`, mais une constante (à partir d'un certain rang). On doit donc faire une étude plus précise de la complexité, et faire le calcul de la somme plus proprement. Une étude plus fine nous montre que l'algorithme est beaucoup plus long pour les entiers en plaisances de deux, mais les autres nombres, on n'a pas besoin d'autant de calcul. Faisons cette étude plus fine.

On nomme $\mathcal{T}$ l'ensemble des tableaux de taille n contenant des valeurs dans $\{0,1\}$. On a,

$$\mathrm{Coût}_{\mathcal{A}}(n) = \sum_{t\in\mathcal{T}} \mathrm{Coût}_{\mathrm{Incr}}(t).$$

Partitionnons $\mathcal{T}$:

$$\mathcal{T}_i = \left\{ \begin{array}{|c|c|c|c|c|c|} \hline \ldots & i+1 & i & i-1 & \ldots & 0 \\ \hline \ldots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \ldots & \mathbf{1} \\ \hline \end{array} \in \mathcal{T} \right\}.$$

Si $t \in \mathcal{T}_i$, on sait que $\text{Coût}_{\text{Incr}}(t) = i + 1$. Ainsi,

$$\begin{aligned}
\sum_{t \in \mathcal{T}} \text{Coût}_{\text{Incr}}(t) &= \sum_{i=0}^{n-1} \sum_{t \in \mathcal{T}_i} \text{Coût}_{\text{Incr}}(t) \\
&= \sum_{i=0}^{n-1} |\mathcal{T}_i| \cdot (i+1) \\
&= \sum_{i=0}^{n-1} 2^{n-1-i} \, (i+1) \\
&= 2^n \sum_{i=1}^{n} \frac{i}{2^i} \\
&= 2^n \times \mathcal{O}(1) \\
&= \mathcal{O}(2^n)
\end{aligned}$$

ce qui explique les résultats trouvés précédemment. L'incrementation `i = i + 1` est donc en $\mathcal{O}(1)$, et non en $\mathcal{O}(\log_2 \texttt{i})$.

Définition : Étant donnée une structure (des éléments d'un type de données abstrait, TDA), $\mathbb{F}$, munie d'opérations $\mathbb{O}$ opérant sur $\mathbb{F}$ munis de fonctions de coût

$$\forall o \in \mathbb{O}, \quad C_o : \mathbb{F} \to \mathbb{R}^+.$$

Étant donné un élément initial $f_0 \in \mathbb{F}$ et une suite d'opérations $(o_1, \ldots, o_n) \in \mathbb{O}^n$, cela conduit donc à une suite d'éléments

$$f_0 \overset{o_1}{\rightsquigarrow} f_1 \overset{o_2}{\rightsquigarrow} f_2 \rightsquigarrow \cdots \rightsquigarrow f_n.$$

On appelle *complexité* de cette séquence, notée $\tilde{c}$,

$$\tilde{C}\big((o_1, \ldots, o_n), f_0\big) = \sum_{i=0}^{n} C_{o_i}(f_{i-1}).$$

On appelle alors *complexité amortie* depuis $f_0 \in \mathbb{F}$ la suite

$$C_{\text{A}}(f_0, n) = \frac{1}{n} \sup_{(o_1, \ldots, o_n) \in \mathbb{O}^n} \tilde{C}\big((o_1, \ldots, o_n), f_0\big).$$

Exemple (Tableaux dynamiques) :
On s'intéresse aux tableaux à longueur variable : on alloue un tableau de petite taille, et on alloue plus de mémoire au besoin. On a une structure de tableau dynamique :

Algorithme 2 Agrandit(T), fonction agrandissant le tableau t

1: Soit taille$' = f\big(\text{len}(T)\big)$ ▷ *f reste à déterminer*
2: On alloue T' de taille taille$'$
3: On recopie T dans T'
4: $T \leftarrow T'$

Cet algorithme a une complexité $\text{Coût}_{\text{Agrandit}}(n) = n + f(n)$. On suppose que le tableau T est rempli jusqu'à la r-ième case.

Algorithme 3 AJOUT(T, x), ajout d'un élément dans le tableau

```
si len(T) = r alors
    AGRANDIT(T)
T[r] ← x
r ← r + 1
```

On choisit la fonction f.

Cas 1 On choisit $f(n) = n + 1$. Soit une suite de n opérations AJOUT depuis un tableau de taille 1, où $r = 0$. Ainsi,

$$f_0 \overset{\text{AJOUT}}{\rightsquigarrow} f_1 \overset{\text{AJOUT}}{\rightsquigarrow} \cdots \rightsquigarrow f_i \rightsquigarrow \cdots \rightsquigarrow f_{n-1} \overset{\text{AJOUT}}{\rightsquigarrow} f_n.$$

La complexité de cette suite d'opérations est

$$\tilde{C}((o_1, \ldots, o_n), f_0) = n + 2 \cdot \frac{n(n+1)}{2},$$

d'où la complexité amortie est de $C_{\mathrm{A}}(f_0, n) = \Theta(n)$.

Cas 2 On choisit $f(n) = 2n$. On somme les complexités : $2n$ (clairement par dessin). Ainsi, $C_{\mathrm{A}}(f_0, n) = \Theta(1)$.

Méthode (du potentiel) **:** Considérons une fonction $h : \mathbb{F} \to \mathbb{R}^+$ dite *de potentiel* telle que $h(f_0) = 0$. Intéressons nous alors à $\underline{C}_o(f) = C_o(f) + h(\bar{f}) - h(f)$, où $f \overset{o}{\rightsquigarrow} \bar{f}$. Soit alors

$$f_0 \overset{o_1}{\rightsquigarrow} f_1 \overset{o_1}{\rightsquigarrow} f_2 \rightsquigarrow \cdots \rightsquigarrow f_n$$

une suite d'opérations. Alors,

$$\begin{aligned}\sum_{i=1}^{n} \underline{C}_{o_i}(f_{i-1}) &= \sum_{i=1}^{n} \Big(C_{o_i}(f_{i-1}) + h(f_i) - h(f_{i-1}) \Big) \\ &= \Big(\sum_{i=1}^{n} C_{o_i}(f_{i-1}) \Big) + \underbrace{h(f_n) - h(f_0)}_{\geqslant 0}\end{aligned}$$

par télescopage. Ainsi,

$$\sum_{i=1}^{n} C_{o_i}(f_{i-1}) \leqslant \sum_{i=1}^{n} \underline{C}_{o_i}(f_{i-1}).$$

EXEMPLE :
On applique la méthode du potentiel au cas 2 de l'exemple ci-avant. On rappelle que $\mathbb{F}$ est l'ensemble des tableaux. On pose la fonction

$$\begin{aligned} h : \mathbb{F} &\longrightarrow \mathbb{R}^+ \\ (T, r) &\longmapsto 6 \left(r - \frac{\mathrm{len}(T)}{2} \right) \end{aligned}$$

Inspectons alors

$$\underline{C}_{\text{AJOUT}}(T, r) = C_{\text{AJOUT}}(\underline{T}, \underline{r}) + 6\bar{r} - 3\,\mathrm{len}(\bar{T}) - 3\underline{r} + 3\,\mathrm{len}(\underline{T}).$$

Si $\text{len}(\underline{T}) = \underline{r}$, alors $C_{\text{Ajout}}(\underline{T}, \underline{r}) = 3\,\text{len}(\underline{T})$ et $\text{len}(\bar{T}) = 2\,\text{len}(\underline{T})$ et $\bar{r} = \underline{r} + 1$. D'où,

$$\underline{C}_{\text{Ajout}}(\underline{T}, \underline{r}) = 3\,\text{len}(\underline{T}) + 6\underline{r} + 6 - 6\,\text{len}(\underline{T}) - 6\underline{r} + 3\,\text{len}(\underline{T}) = 6.$$

Sinon, $\text{len}(\underline{T}) > \underline{r}$, alors $\text{len}(\bar{T}) = \text{len}(\underline{T})$ et $\bar{r} = \underline{r} + 1$. Ainsi,

$$\underline{C}_{\text{Ajout}}(\underline{T}, \underline{r}) = 1 + 6(\underline{r} + 1) - 6\,\text{len}(\bar{T}) - 6\underline{r} + 6\,\text{len}(\underline{T}) = 7$$

D'où

$$\sum_{i=1}^{n} C_{o_i}(f_{i-1}) \leqslant \sum_{i=1}^{n} \underline{C}_{o_i}(f_{i-1}) \leqslant 7n.$$

Le coût amorti est en $\mathcal{O}(1)$.

Exemple (Méthode du Banquier) :
On encode une file avec deux piles. Au moment de défiler, on doit potentiellement transvaser une pile dans une autre. Avec la méthode du Banquier, on a l'*intuition* que le coût amorti est constant.

On pose $\mathbb{F}$ l'ensemble des couples de piles (p_1, p_2). On a

$$C_{\text{défiler}}\big((p_1, p_2)\big) = \begin{cases} \text{taille } p_1 + 1 & \text{si } p_2 \text{ est vide} \\ 1 & \text{sinon} \end{cases}, \text{ et } C_{\text{enfiler}}\big((p_1, p_2)\big) = 1.$$

Soit h la fonction de potentiel définie comme

$$\begin{aligned} h : \mathbb{F} &\longrightarrow \mathbb{R}^+ \\ (p_1, p_2) &\longmapsto \text{taille } p_1 \end{aligned}$$

Étudions alors $\underline{C}_{\text{défiler}}\big((p_1, p_2)\big)$.

— Si p_2 est vide, alors

$$\begin{aligned} \underline{C}_{\text{défiler}}\big((p_1, p_2)\big) &= C_{\text{défiler}}\big((p_1, p_2)\big) + h\big((\bar{p}_1, \bar{p}_2)\big) - h\big((p_1, p_2)\big) \\ &= \text{taille } p_1 + 1 + \overbrace{\text{taille } \bar{p}_1}^{=0} - \text{taille } p_1 \\ &= 1. \end{aligned}$$

— Si p_2 n'est pas vide, alors

$$\underline{C}_{\text{défiler}}\big((p_1, p_2)\big) = 1 + \underbrace{\text{taille } \bar{p}_1}_{=\text{taille } p_1} - \text{taille } p_1.$$

D'où, pour $(p_1, p_2) \in \mathbb{F}$, $\underline{C}_{\text{défiler}}\big((p_1, p_2)\big) \leqslant 1$. De plus,

$$\begin{aligned} \underline{C}_{\text{enfiler}}\big((p_1, p_2)\big) &= C_{\text{enfiler}}\big((p_1, p_2)\big) + h\big((\bar{p}_1, \bar{p}_2)\big) - h\big((p_1, p_2)\big) \\ &= 1 + \text{taille } \bar{p}_1 - \text{taille } p_1 \\ &= 2 \end{aligned}$$

Finalement, pour toute séquence d'opérations $o_1, \ldots, o_n$ initialisée à la file vide f_0, on a

$$\begin{aligned}\frac{1}{n}\tilde{C}\big((o_1, \ldots, o_n), f_0\big) &= \frac{1}{n}\sum_{i=0}^{n} C_{o_i}(f_{i-1}) \\ &\leqslant \frac{1}{n}\sum_{i=1}^{n} C_{o_i}(f_{i-1}) \\ &\leqslant 2\end{aligned}$$

D'où, un coût amorti constant.