INFO602, L3 Informatique, Algorithmique II
Lesson 2, analyse amortie

Jacques-Olivier Lachaud
LAMA, Université Savoie Mont blanc
https://www.lama.univ-savoie.fr/wiki
(suivre INFO602)

17 mars 2020

1 Analyse amortie des algorithmes

Dans une *analyse amortie*, le temps requis pour effectuer une suite d'opérations sur une structure de données est une moyenne sur l'ensemble des opérations effectuées. Chaque opération peut être coûteuse, mais l'analyse amortie permet de montrer que le coût moyen de chaque opération est faible (bien sûr si tel est vraiment le cas). Une analyse amortie est différente de l'analyse en moyenne car il n'est ici nul question de chance ou de probabilités. L'analyse amortie garantit les performances en moyenne de chaque opération dans le *cas le plus défavorable.* [1]

1.1 Méthode de l'agrégat

Dans la *méthode de l'agrégat* on montre que, pour tout m, une suite de m opérations prend le temps total $T_{total}(m)$ dans le pire cas. Dans ce pire cas, le coût moyen ou **coût amorti** par opération est $T_{total}(m)/m$. Ce coût amorti est donc le même quel que soit l'opération effectuée dans la séquence (car la séquence peut contenir plusieurs types d'opérations).

1.1.1 Exemple sur des opérations de Pile.

On se donne les opérations suivantes sur les Piles : Empiler(S, x), Dépiler(S), PileVide(S) et MultiDépiler(S, k) qui appelle k fois Dépile(S) ou s'arrête si la pile est vide. Pour simplifier, on supposera donc que les trois premières méthodes ont chacune un coût de 1, la dernière a en revanche un coût en $\mathcal{O}(k)$.

Il est clair qu'une suite de m opérations Empiler et Dépiler dans un ordre quelconque a un coût m (avec un temps d'exécution en temps réel en $\Theta(m)$). Le coût d'un appel à MultiDépiler est en revanche en $\min(s, k)$ où s est le nombre d'éléments dans la pile.

Prenons une pile initialement vide. Effectuons une suite de m opérations arbitraires Empiler, Dépiler, et MultiDépiler. La taille de la pile étant en pire cas de m, et une opération MultiDépiler pouvant avoir aussi un coût $\Theta(m)$, il vient vite que les m opérations ont un coût en $\mathcal{O}(m^2)$.

Cette borne est largement surestimée, tout simplement car on a regardé le pire cas d'une opération mais pas de toutes les opérations. Si on regarde les m opérations, on sait qu'il ne peut pas y avoir plus d'appels à Dépiler qu'à Empiler. La somme de tous les appels Dépiler faits directement ou par MultiDépiler ne peut excéder le nombre d'éléments empilés, qui est $\leq m \in \Theta(m)$. On obtient alors la complexité en pire cas de m opérations dans $\Theta(m)$. Plus précisement, soit $e \leq m$ le nombre

1. Une partie de ce chapitre est un résumé de Corben *et al.*

d'éléments empilés :

$$T_{total}(m) = \underbrace{\textsc{Empiler} + \cdots + \textsc{Empiler}}_{\leq e\Theta 1} + \underbrace{\textsc{Depiler} + \cdots + \textsc{MultiDepiler}}_{\leq e\Theta(1)} + \underbrace{\textsc{PileVide} + \cdots + \textsc{PileVide}}_{\leq (m-e)\Theta(1)}$$

Le **coût amorti** de MultiDépiler est donc bien de $\frac{(m+e)\Theta(1)}{m} = \Theta(1)$.

1.1.2 Exemple sur un compteur binaire.

On se donne maintenant un compteur binaire implementé comme un tableau $A[0..k-1]$ de bits, avec $long(A) = k$. Un tel tableau correspond au codage binaire d'un nombre $x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$. Au départ $x = 0$. Pour ajouter 1 modulo 2^k, on utilise l'action suivante :

Algorithme 1 : Fonction Incrémenter pour compteur binaire.

```
Action Incrémenter( ES A );
début
    i ← 0;
    tant que i < long(A) et A[i] = 1 faire
        A[i] ← 0;
        i ← i + 1;
    si i < long(A) alors A[i] ← 1;
```

Le coût de Incrémenter est donc proportionnel au nombre de bits basculés. Une séquence de m appels à Incrémenter prend donc $T_{total} = \mathcal{O}(mk)$ opérations dans le cas le plus défavorable, puisqu'en pire cas k bits peuvent potentiellement basculer. Mais ceci surestime la borne. En réalité, on peut borner le nombre de fois où chaque bit bascule plus précisément.

A[k-1]	...	A[3]	A[2]	A[1]	A[0]	Coût
0	...	0	0	0	0	
0	...	0	0	0	1	1
0	...	0	0	1	0	2
0	...	0	0	1	1	1
0	...	0	1	0	0	3
0	...	0	1	0	1	1
0	...	0	1	1	0	2
0	...	0	1	1	1	1
0	...	1	0	0	0	4
			...			

A chaque appel, un seul bit bascule à un. Or un bit bascule à zéro seulement s'il avait été basculé à un avant. Donc le nombre de bits basculés à zéro est plus petit que le nombre de bits basculés à un, qui est m.

$$T_{total}(m) = \Theta(\underbrace{\#(\text{bits } 0 \to 1)}_{m} + \underbrace{\#(\text{bits } 1 \to 0)}_{\leq m}).$$

On voit que le nombre *total* de bits basculés est au maximum de l'ordre de $2m$. Le coût amorti de Incrémenter est donc de $\Theta(1)$.
Exercices : Vous pouvez faire déjà TD 2 : exo 1.

1.2 Méthode comptable

Dans la **méthode comptable** on affecte des coûts différents à chaque opération, certaines recevant un coût supérieur et d'autres un coût inférieur à leur coût réel. Chacun de ces coûts définit le *coût amorti* de l'opération. Lorsque pour une opération donnée, le coût amorti excède le coût réel, la différence est affectée à un objet de la structure comme crédit pour une opération future. On voit donc que dans cette méthode les coût amortis des opérations sont potentiellement différents, contrairement à la méthode de l'agrégat où le coût amorti est réparti équitablement.

La difficulté est donc de bien choisir les coûts amortis. Si on note c_i le coût réel de la i-ème opération et $\hat{c}_i$ son coût amorti, il faut faire en sorte que pour toutes les séquences de m opérations, on ait

$$\sum_{i=1}^{m} \hat{c}_i \geq \sum_{i=1}^{m} c_i. \tag{1}$$

D'après (1), il faut faire attention à ce que le crédit total stocké ne devienne jamais négatif. Ensuite le temps total sera borné par la somme des coûts amortis.

$$T_{total} = \underbrace{\sum_{i=1}^{m} c_i}_{\text{coûts réels}} \leq \underbrace{\sum_{i=1}^{m} \hat{c}_i}_{\text{coûts amortis}}$$

Par ailleurs, le coût amorti par opération est alors simplement $\hat{c}_i$.

1.2.1 Exemple sur des opérations de Pile.

Pour illustrer la méthode, voilà les coûts réels et coûts amortis choisis pour les opérations sur la pile :

i-ème Opération	coût réel c_i	coût amorti $\hat{c}_i$
Empiler(S, x)	1	2
Dépiler(S)	1	0
MultiDépiler(S, k)	$\min(s, k)$	0

Pour simplifier, on considère que PileVide ne coûte rien. Il est clair qu'à chaque fois qu'on empile une valeur, on dispose d'un crédit de 1 associé à cette valeur empilée. Donc, lorsqu'on dépile avec Dépiler, la valeur qu'on dépile dispose toujours d'un crédit de 1. Ce crédit est donc un acompte pour payer le coût de son dépilement. Lorsqu'on dépile, le crédit permet de payer exactement la différence coût réel moins coût amorti.

De même, lors d'un appel à MultiDépiler, il y aura toujours un crédit de 1 par valeur dépilée, et donc le coût réel $\min(s, k)$ sera toujours compensé par ces crédits. Donc, pour une séquence quelconque de m opérations Empiler, Dépiler et MultiDépiler, le coût amorti total est un majorant du coût total d'après (1). Une fois qu'on a bien montré que (1) est satisfaite ($\sum_{i=1}^{m} \hat{c}_i \geq \sum_{i=1}^{m} c_i$), on calcule le coût amorti total. Soit $e < m$ le nombre de Empiler, alors

$$\begin{aligned} T_{total} = \underbrace{\sum_{i=1}^{m} c_i}_{\text{coûts réels}} &\leq \underbrace{\sum_{i=1}^{m} \hat{c}_i}_{\text{coûts amortis}} \\ &\leq e\hat{c}(\text{Empiler}) + (m-e)\max(\hat{c}(\text{Depiler}), \hat{c}(\text{MultiDepiler})) \\ &\leq 2 \times e + (m-e) \times 0 \leq 2m. \end{aligned}$$

Le temps total de toutes les m opérations est bien un $\Theta(m)$, et on en conclurait que le temps amorti de chaque opération est $\Theta(m)/m = \Theta(1)$.

On peut aussi conclure avant que le coût amorti de chaque Empiler est 2, tandis que le coût amorti de chaque Depiler ou MultiDepiler est 0 !

Exercices : Vous pouvez faire déjà TD 2 : exo 2 et exo 3.

1.2.2 Exemple sur un compteur binaire.

Pour le compteur binaire, le principe est similaire. A chaque fois que l'on bascule un bit à 1, on met un coût amorti de 2 alors que le coût réel est 1. A chaque fois que l'on rebasculera ce bit à 0, on aura donc un crédit de 1, qui correspond à son coût réel. Comme le nombre de bits à 1 n'est jamais négatif, le crédit n'est jamais négatif. Le coût total d'un séquence de n INCRÉMENTER est donc de $2n = \Theta(n)$.
Exercices : Vous pouvez faire déjà TD 2 : exo 4.

1.3 Méthode du potentiel

La méthode du potentiel considère un crédit global (ou potentiel) attaché à toute la structure plutôt que de répartir le crédit sur chacun des éléments de la structure. Ce potentiel peut donc servir à payer des opérations futures. On construira ce potentiel de manière à ce qu'il soit toujours positif ou nul.

Soit D_0 notre structure de données initiale sur laquelle les n opérations sont effectuées. Pour chaque i-ème opération sur la structure D_{i-1}, soit c_i son coût réel et D_i la structure résultante. La fonction *potentiel* Φ associe un nombre réel à chaque structure D_i. On définit le **coût amorti** $\hat{c}_i$ de la i-ème opération par :

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}). \tag{2}$$

Il est facile de voir que le **coût amorti total** est :

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0). \tag{3}$$

Si Φ est construite de manière à ce que $\Phi(D_i) \geq \Phi(D_0)$ alors le coût amorti total est un majorant du coût réel total. Le résultat dépend donc de la fonction Φ choisie. Il faudra donc "sentir" quelle est la bonne fonction Φ pour obtenir la complexité voulue.

On calculera ainsi le coût amorti de chaque opération possible, on sommera ces coûts amortis, ce qui nous donnera un majorant du coût réel total en pire cas.

1.3.1 Exemple sur des opérations de Pile.

On définit simplement le potentiel Φ d'une pile comme étant le nombre d'éléments de la pile. On a donc évidemment $\Phi(D_0) = 0$ si la pile est vide au début et $\forall i, \Phi(D_i) - \Phi(D_0) = \Phi(D_i) \geq 0$. On est donc bien dans les conditions où le coût amorti est à tout moment un majorant du coût réel.

Une fois Φ bien choisi, le calcul des coûts amortis est direct en sommant coût réel plus $\Phi(D_i) - \Phi(D_{i-1})$ (cf. (2)).

i-ème Opération	coût réel c_i	$\Phi(D_i) - \Phi(D_{i-1})$	coût amorti $\hat{c}_i$
EMPILER(S, x)	1	$(s+1) - s$	2
DÉPILER(S)	1	$(s-1) - s$	0
MULTIDÉPILER(S, k)	$\min(s, k)$	$(s - \min(s, k)) - s$	0
PILEVIDE(S)	1	$s - s$	1

On vérifie les coûts amortis :
— EMPILER : $\hat{c}_i = 1 + 1$ (car la pile augmente de 1)
— DÉPILER : $\hat{c}_i = 1 - 1$ (car la pile diminue de 1)
— MULTIDÉPILER : $\hat{c}_i = k' - k'$ (si $k' = \min(k, s)$, car la pile diminue de k')
— PILEVIDE : $\hat{c}_i = 1 - 0$

Comme le coût amorti de chaque opération est un $O(1)$, il est linéaire pour n'importe quelle séquence d'opération.

1.3.2 Exemple sur un compteur binaire.

On prend cette fois-ci pour $\Phi(D_i)$ le nombre de bits à 1 b_i dans la structure D_i qui modélise le compteur.

Si l'appel de INCRÉMENTER a réinitialisé t_i bits, le coût réel de l'opération est au plus de $1 + t_i$ (car il met t_i bits à 0 et 1 bit à 1). Si $b_i = 0$ alors $b_{i-1} = t_i = k$. Si $b_i > 0$ alors $b_i = b_{i-1} - t_i + 1$. On vérifie que $b_i \leq b_{i-1} - t_i + 1$ dans tous les cas. La différence de potentiel est :

$$\begin{aligned} \Phi(D_i) - \Phi(D_{i-1}) &= b_i - b_{i-1} \\ &\leq b_{i-1} - t_i + 1 - b_{i-1} \\ &= 1 - t_i. \end{aligned}$$

Le coût amorti vaut alors :

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &\leq (t_i + 1) + (1 - t_i) \\ &= 2. \end{aligned}$$

On peut même analyser le coût amorti lorsqu'on part d'un compteur à une valeur arbitraire. On trouve que le coût global :

$$\sum_{i=1}^{n} c_i \leq 2n - b_n + b_0.$$

Ce qui permet de conclure quand même sur la linéarité de m incrémentations si $k = \Theta(m)$.

En résumé : Analyse amortie des algorithmes

- L'analyse amortie permet de déterminer de façon plus fine la complexité en pire cas d'une séquence d'opérations, notamment dans le cas où les opérations n'ont pas un coût forcément constant.
- On dispose de trois méthodes standards pour l'analyse amortie : (i) la méthode de l'agrégat détermine un majorant pour le temps d'exécution de toute la séquence en pire cas, (ii) la méthode comptable affecte des coûts différents à chaque opération considérée et permet de mettre de côté du temps pour des opérations ultérieures plus coûteuses, (iii) la méthode du potentiel attribue un crédit global à toute la structure, ce crédit pouvant être dépensé ultérieurement dans d'autres opérations.