Analyse amortie

Pierre Karpman

Lycée Champollion MP2I

https://membres-ljk.imag.fr/Pierre.Karpman/CPGE/2025/

Table des matières

Motivation & principe

Il est courant dans certains contextes (notamment les implémentations de structures de données) que :

- le coût pire cas d'une fonction soit élevé
- la moyenne des coûts de plusieurs appels portant sur une suite d'arguments reliés entre eux soit (asymptotiquement) plus faible. On appel coût amorti un tel coût (et *pas* coût moyen)

Dans un tel cas, l'analyse pire cas n'est *pas forcément* (la plus) pertinente, et une analyse amortie donne une information intéressante (mais elle-même pas toujours très pertinente, par exemple comme dans un contexte de système temps réel)

Exemple paradigmatique

Soit l'implémentation suivante de file fonctionnelle avec deux piles :

```
type 'a queue = { out: 'a list; inp: 'a list }
let push x { out ; inp } = { out; inp = x::inp }
let rec pop_opt
    = function
    | { out = []; inp = [] } -> None
    | { out = x::xs; inp }
        -> Some (x, { out = xs; inp })
    | { out = []; inp }
        -> pop_opt { out = List.rev inp; inp = [] }
```

- Le coût pire cas de `pop_opt` est linéaire en la longueur (du champ `inp`) de son argument
- Mais toute suite de N opérations `push`/`pop` **à partir d'une file vide** aura un coût **total** linéaire en N
- Le coût **amorti** de `pop_opt` est constant

Remarques sur la terminologie

- On parle de coût/analyse amortie et non pas « en moyenne » pour éviter une confusion avec une autre notion de coût/analyse en moyenne
- L'analyse en moyenne fait référence à une analyse où l'on s'intéresse au coût moyen d'**une** exécution (et pas une *suite* d'exécutions reliées entre elles comme dans l'analyse amortie) pour des entrées distribuées aléatoirement
- Formellement les entrées sont des variables aléatoires et l'on s'intéresse à l'espérance du coût
- C'est une technique d'analyse qui fait des hypothèses sur la loi de probabilité suivie par les entrées, et les résultats peuvent être difficiles à interpréter
- Vous verrez l'année prochaine les *algorithmes probabilistes de type Las Vegas*, où l'on s'intéresse à l'espérance du coût sur des variables aléatoires « contrôlées » par l'algorithme (ce qui a beaucoup plus de sens)

Table des matières

Méthodes d'analyse 

Aucune méthode d'analyse amortie n'est au programme, mais cela n'empêche pas d'en parler très rapidement

Trois grandes approches classiques :

- Méthode « de l'agrégat »
  - Essentiellement une analyse *ad hoc* ?
- Méthode « comptable »
- Méthode « du potentiel »
  - La plus formelle ?

Méthode comptable

Principe

- On affecte un coût amorti à chaque opération, qui représente une « facture » à payer au moment de la réaliser
- Lors d'une suite d'opération, on somme (« crédite sur le compte » de la structure) les coûts amortis et soustrait (« débite du compte ») les coûts réels
- Le choix des coûts amortis est correct si le compte de la structure est toujours positif pour n'importe quelle suite d'opération

Exemple sur la file

Métrique : nombre d'opérations de liste

- Coût amorti `push x { out; inp }` : on facture 3
  - 1 pour l'ajout sur `inp`, 1 pour l'ajout sur `out`, 1 pour l'extraction de `out`
- Coût amorti `pop_opt q` : on facture 0
  - Tout a été payé d'avance par `push`

Les deux opérations sont de coût amorti constant

Méthode du potentiel

Principe

- On définit une *fonction potentielle* sur l'état de la structure de donnée, typiquement réelle et valant zéro pour un état vide et étant positive pour tout autre état
- On montre que chaque opération a un coût amorti (idéalement facile à évaluer) qui majore le coût réel *plus le potentiel créé* (qui peut être négatif)
- Le coût d'une suite d'opérations depuis une structure vide se majore par une somme télescopique, facile à évaluer

L'idée sous-jacente est qu'une opération qui coûte cher doit consommer du potentiel (qui a dû être créé par des opérations qui ne coûtent pas cher), et limite le coût possible des opérations futures (car on a baissé le potentiel, toujours positif)

Plus formellement

On définit une fonction potentielle φ de façon à avoir :

- $\varphi(\varnothing) = 0$, $\varphi(s) \geq 0$ pour un état s quelconque
- $a_i \geq c_i + \varphi(s_i) - \varphi(s_{i-1})$ où c_i est le coût réel d'une opération quelconque produisant un état s_i à partir d'un état s_{i-1} ; *le coût amorti a_i majore le coût réel plus le potentiel* **créé** (éventuellement négatif)
- De façon équivalente, $c_i \leq a_i + \varphi(s_{i-1}) - \varphi(s_i)$; *le coût réel est majoré par le coût amorti plus le potentiel* **consommé** (*ditto*)
- Pour que ce soit pratique, a_i doit être facile à évaluer

Le coût d'une suite de n opérations produisant des états $s_1, \ldots, s_n$ à partir d'un état s_0 vide est alors :

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} a_i + \varphi(s_{i-1}) - \varphi(s_i) = \sum_{i=1}^{n} a_i + \varphi(s_0) - \varphi(s_n) \leq \sum_{i=1}^{n} a_i$$

Exemple sur la file

Définition du potentiel

On choisit `let pot { inp } = List.length inp`

- Nul pour une file vide, positif pour tout autre état

Coût des opérations

Métrique : nombre d'opérations de liste

- `push` : coût réel : 1, potentiel créé : 1
  - coût amorti 2
- `pop_opt` :
  - (zéro sur une file vide)
  - si `out` non vide : coût réel 1, potentiel créé : 0
  - si `out` vide, `inp` de longueur n : coût réel $1 + n$, potentiel créé : $-n$
  - coût amorti 1

Les coûts amortis des opérations sont constants