Chapître IX.
Analyse amortie

1. Analyse du temps

Nous connaissons déjà deux types d'analyse du temps d'un algorithme:

1) analyse au pire des cas

2) analyse du cas moyen

En plus, nous avons considéré de nombreux exemples pour l'analyse du temps d'un algorithme ou d'une opération d'une structure de données.

Exemple:

1) algorithmes de tri, algorithme de PRIM, arbres binaires de recherche, tas binomial etc.

2) tri rapide, hachage

2. Analyse amortie

L'**analyse amortie** est un autre outil pour analyser le temps des opérations des structures de données.

Dans une **analyse amortie**, on considère le temps pour effectuer une **séquence d'opérations** sur une structure de données.

L'**analyse amortie** peut servir à montrer que le coût moyen d'une opération est faible, si l'on établit sa moyenne sur une séquence d'opérations, même si une seule opération peut se révéler coûteuse.

Très importants:

-) L'analyse amortie n'est pas un outil pour l'analyse (directe) du temps d'un algorithme.

-) **L'analyse amortie n'est pas une analyse du cas moyen.**

-) **L'analyse amortie est une amélioration de l'analyse au pire des cas pour les besoins d'analyse de performance des structures de données.**

3. Un exemple

Nous considérerons en cours des tas de Fibonacci qui sont une implémentation d'une file de priorité et aussi une application très importante d'analyse amortie.

Dans un tas de Fibonacci l'opération $EXTRAIRE - MIN$ s'exécute en temps $O(n)$ dans le pire des cas.

Heureusement le coût amorti de l'opération $EXTRAIRE - MIN$ dans une séquence d'opérations d'un tas Fibonacci est $O(\log n)$.

C'est-à-dire, dans une séquence quelconque d'opérations admissibles d'un tas Fibonacci le coût moyen de toutes les opérations $EXTRAIRE - MIN$ est $O(\log n)$ au pire des cas.

C'est-à-dire, s'il y a k opérations $EXTRAIRE - MIN$ dans une séquence, alors le temps d'exécution de ces k opérations $EXTRAIRE - MIN$ est au pire des cas $k \cdot O(\log n)$.

Une analyse amortie garantit la performance moyenne de chaque opération dans le pire des cas.

Dans une analyse amortie on considère le temps requis pour effectuer une séquence d'opérations sur une structure de données.

Il faut être capable de comprendre et d'interpréter des bornes amorties comme par exemple pour

-) tas de Fibonacci

-) fôrets d'ensembles disjoints

4. Techniques utilisées en analyse amortie

-) la méthode par agrégat

-) la méthode comptable

-) la méthode du potentiel

Faire une analyse amortie est une tache qui est vraiment difficile. On peut le laisser faire par des algorithmiciens.

5. La necessité de l'analyse amortie

Nous regardons la méthode par agrégat pour un petit exemple.

Pile modifiée:

Soit S une pile. C'est-à-dire que S supporte les opérations

$EMPILER(S, x)$ empile l'objet x sur la pile S.

$DEPILER(S)$ dépile le sommet de la pile S et retourne l'objet dépilé.

Les opérations s'exécutent chaque en temps $O(1)$ au pire des cas.

On ajoute une autre opération utilisant l'opération $PILE\!-\!VIDE(S)$ qui retourne $VRAI$ s'il n'y a plus aucun objet sur la pile, sinon $FAUX$.

$MULTIPOP(S, k)$

tant que pas $PILE - VIDE(S)$ et $k \neq 0$
  faire $DEPILER(S)$
    $k \leftarrow k - 1$

Alors l'opération $MULTIPOP(S, k)$ s'exécute en temps $O(\min(s, k))$, si le nombre d'objets de S est s.

Analysons une séquence de n opérations $EMPILER$, $DEPILER$ et $MULTIPOP$.

au pire des cas:

$MULTIPOP(S, k)$ s'exécute en temps $O(n)$ au pire des cas, par exemple si $s = n$ et $k = n$.

$\Rightarrow$ Une séquence de n opérations coûte $O(n^2)$, puisqu'on a $O(n)$ $MULTIPOP$ opérations coûtant chacune $O(n)$.

$O(n^2)$ **est correct mais trop pessimiste,** parce qu'il est impossible d'avoir beaucoup d'opérations $MULTIPOP$ coûteuses dans une séquence de n opérations.

Grâce à la méthode d'analyse amortie par agrégat, on peut obtenir une meilleure borne supérieure, prenant en compte globalement la séquence des n opérations.

analyse amortie:

Une séquence quelconque de n opérations $EMPILER$, $DEPILER$ et $MULTIPOP$ **sur une pile initialement vide** peut coûter au plus $O(n)$.

Pourquoi? Chaque objet peut être dépilé (par $DEPILER$ ou $MULTIPOP$) au plus une fois pour chaque empilement de ce même objet.

Puisque le nombre d'opérations $EMPILER$ est au plus n, le nombre d'opérations $DEPILER$ à l'intérieur de $MULTIPOP$ est au plus n.

$\Rightarrow$ **Pour une valeur quelconque de** n**, n'importe quelle séquence de** n **opérations** $EMPILER$**,** $DEPILER$ **et** $MULTIPOP$ **sera donc au total de** $O(n)$**.**

$\Rightarrow$ **Le coût amorti d'une opération est la moyenne:** $O(n)/n = O(1)$**.**

6. Conclusion

1) L'analyse amortie a permis de démontrer qu'une séquence d'opérations s'exécutent au pire des cas en $O(n)$. C'était impossible de le démontrer avec une analyse classique au pire des cas.

2) C'est l'analyse amortie qui facilitie aussi la construction des structures de données ayant des opérations coûteuses au pire des cas qui sont trés rares
(comme $EXTRAIRE - MIN$ pour un tas de Fibonacci).