TD 06 – Analyse amortie

Exercice 1. *Pile*

On considère une pile munie des opérations suivantes :
- $PUSH(S, x)$: empile un objet x sur la pile S
- $POP(S)$: dépile le sommet de la pile S et retourne l'objet dépilé
- $MULTIPOP(S, k)$: dépile au plus k objets de la pile S

Algorithm 1: $MULTIPOP(S, k)$

```
début
    tant que S ≠ ∅ et k ≠ 0 faire
        POP(S);
        k ← k − 1;
```

1. Quelle est la complexité de chacune des 3 opérations ? En déduire avec la méthode globale (méthode de l'agrégat) le coût amorti pour une suite de n opérations $PUSH$, POP et $MULTIPOP$ sur une pile initialement vide.
2. Même question avec la méthode des acomptes.
3. Même question avec la méthode des potentiels.
4. On souhaite implémenter une file à l'aide de deux piles, de telle façon que le coût amorti des opérations $ENQUEUE$ et $DEQUEUE$ soit $O(1)$. Comment peut-on faire ?

Exercice 2. *Remise à zéro d'un compteur binaire*

On analyse des opérations sur un compteur binaire sur k bits qui commence à zéro. Ce compteur est représenté par un tableau $A[0..k-1]$ de bits, et un nombre x représenté par le compteur est tel que $x = \sum_{i=0}^{k-1} A[i].2^i$.

1. Rappeler le principe d'une opération Incrémenter sur ce compteur, qui rajoute 1 modulo 2^k à la valeur actuelle du compteur. Montrer que si l'on disposait également d'une opération Décrémenter sur le compteur, alors une suite de n opérations pourrait coûter jusqu'à un temps en $O(nk)$.
2. On garde uniquement l'opération Incrémenter, et on rajoute au compteur une opération RÀZ qui remet le compteur à zéro. Montrer comment implémenter ce compteur sous la forme d'un tableau de bits pour qu'une séquence quelconque de n opérations Incrémenter et RÀZ prenne un temps $O(n)$ sur un compteur initialement à zéro.
   Conseil : Gérer un pointeur vers le 1 de poids fort.

Exercice 3. *Recherche d'éléments*

On veut une structure de données qui permette la recherche (d'un élément), et l'insertion (d'un nouvel élément) de manière efficace.

1. Quels sont les coûts de la recherche et de l'insertion si on utilise un tableau trié de taille n ?

On propose la solution suivante : soit $k = \lceil \log(n+1) \rceil$ et soit $(n_{k-1}, n_{k-2}, \ldots, n_0)$ la représentation binaire de n. On a k tableaux triés $A_0, A_1, \ldots, A_{k-1}$, et la taille de A_i est 2^i pour tout i. Le tableau A_i est plein si $n_i = 1$ et vide si $n_i = 0$. Ainsi le nombre total d'éléments stockés dans les k tableaux est $\sum_{i=0}^{k-1} n_i 2^i = n$. Noter que chaque tableau est trié mais qu'il n'y a aucune relation entre les éléments de deux tableaux.

2. Expliquer comment faire une recherche dans cette structure, et donner le coût au pire cas.

3. Expliquer comment faire une insertion dans cette structure, et donner le coût au pire cas et en analyse amortie.
4. Expliquer comment supprimer un élément.

Exercice 4. *Structure de données*

On souhaite avoir une structure de données S contenant des réels quelconques (pouvant être égaux entre eux). Cette structure doit supporter les deux opérations suivantes :

— $Insertion(S, x)$: insère x dans S

— $SuppressionMoitieSuperieure(S)$: supprime les $\lceil |S|/2 \rceil$ données les plus grandes de S

Expliquer comment implémenter ces deux opérations afin qu'elles s'exécutent en $O(1)$ en temps amorti.