Algorithmique
TD n° 11 : Analyse amortie

Exercice 1 : des piles et des files

1. Une pile (“stack") est une structure de données abstraites telle que toute suppression porte sur l'élément inséré le plus récemment, “last in first out" (LIFO). On accède à cette structure par des opérations de création, d'insertion d'un élément (“push"), ou, pour une pile non vide, de suppression de l'élément le plus récemment inséré (“pop").

   Considérer la suite d'opérations $Push(x_1)Push(x_2)\cdots Push(x_8)Pop\ Pop\ Pop\ Pop$ consistant en l'insertion successive des éléments x_1, puis x_2, et ainsi de suite jusqu'à x_8, suivi par 4 suppressions. Quel est l'ensemble des éléments contenus dans la pile à la fin ?

2. Une file (“queue") est une structure de données abstraite telle que tout suppression porte sur l'élément inséré le plus anciennement, “first in first out" (FIFO). On accède à cette structure par des opérations de création, d'insertion d'un élément (“enqueue"), ou, pour une file non vide, de suppression de l'élément le plus anciennement inséré (“dequeue").

   Considérer la suite d'opérations $I(x_1)I(x_2)\cdots I(x_8)SSSS$ consistant en l'insertion successive des éléments x_1, puis x_2, et ainsi de suite jusqu'à x_8, suivi par 4 suppressions, quel est l'ensemble des éléments contenus dans la file à la fin ?

3. On suppose qu'on vit dans un monde étrange où les files n'existent pas encore, mais où il existe des piles dont chaque opération prend temps constant $O(1)$. Proposer une implémentation de la structure de données abstraite file à l'aide de *deux* piles, de telle façon que
   - chaque création ou insertion prend temps constant,
   - Pour la suite d'opérations $I(x_1)I(x_2)\cdots I(x_n)SS...S$ consistant en l'insertion successive des éléments x_1, puis x_2, et ainsi de suite jusqu'à x_n, suivi par $n/2$ suppressions, le temps total d'exécution de cette suite d'opérations est $O(n)$.
   - chaque suppression prend temps au plus $O(n_t)$, où n_t dénote la taille de la file à l'instant t de la suppression.

4. Démontrer que pour toute suite de T opérations, votre implémentation a complexité totale $O(T)$.

5. En déduire que pour votre implémentation, les opérations création, insertion, suppression, de la structure abstraite file ont complexité amortie $O(1)$.

Exercice 2 : la vache et la barrière

On a une vache qui cherche à sortir d' un pré avec une clôture comportant une ouverture, mais la vache ne sait pas où est l'ouverture. Le pré est un demi-plan, la vache se trouve au point origine $A = (0, 0)$, la clôture est la droite d'équation $y = 1$, et l'ouverture est un point sur cette droite. La vache est myope : elle ne peut voir l'ouverture que si elle est juste dessus. Elle va donc commencer par aller jusqu'à la clôture, au point $(0, 1)$, mais alors se pose une question : de quel côté est l'ouverture ? Faut-il aller à droite ou à gauche ? La vache souhaite passer le moins de temps possible à chercher l'ouverture. Elle marche à vitesse constante de 1 mètre par seconde.
Donner un algorithme pour la vache lui garantissant que quelle que soit la position B de l'ouverture, elle la trouvera en temps $O(dist(A, B))$.

Exercice 3 :

On suppose qu'on a une suite d'opérations. La i-ième opération a coût $c(i)$. Dans chacun des cas suivants, est-ce que le coût amorti par opération est constant ?

1. $c(i) = i$ si i est une puissance de 2, et $c(i) = 1$ sinon.
2. $c(i)$ est la plus grande puissance de 2 qui divise i.
3. $c(i)$ est le plus grand k tel que 2^k divise i.

Exercice 4 :

On suppose qu'on peut insérer un élément dans ou le supprimer d'une table de hachage de capacité c en temps $O(1)$ si cette table est remplie à moins de $3/4 = 75\%$. Pour avoir un hachage efficace sans gaspiller de l'espace mémoire, on essaie d'ajuster la taille de la table au nombre d'éléments présents un utilisant une stratégies de recopie : étant donné une table dont la capacité est c et qui est trop ou trop peu remplie, créer une nouvelle table de capacité c' bien choisie, insérer tous les éléments dans la nouvelle table, et détruire la table précédente. Soit n_t le nombre d'éléments dans la table après t opérations (insertions ou suppressions).

1. On s'arrange pour que la table soit toujours à peu près à moitié remplie, plus précisément, qu'elle ait une capacité entre $2n_t - 2$ et $2n_t + 2$, quitte à créer une nouvelle table si nécessaire selon le processus décrit ci-dessus. Quel est le coût d'une opération dans le pire cas ? Quel est le coût amorti d'une opération ? Le démontrer.
2. On s'arrange pour que la table ait capacité entre $(1/4)n_t$ et $(3/4)n_t$, quitte à créer une nouvelle table si nécessaire selon le processus décrit ci-dessus. Quel est le coût d'une opération dans le pire cas ? Quel est le coût amorti d'une opération ? Le démontrer.