TD n° 7 - Analyse amortie

Dans la représentation des listes par des tableaux que l'on a vue en cours, il était nécessaire de déterminer à l'avance la taille maximale que pourrait avoir la liste, pour construire dès le début un tableau assez grand. Cette façon de faire pose plusieurs problèmes : si la taille choisie est bien trop grande, on utilise beaucoup trop de place en mémoire, mais si inversement elle est trop petite, on ne peut plus ajouter d'éléments à la liste.
Pour résoudre ces problèmes, on peut utiliser des tableaux dynamiques. L'idée est de partir d'un tableau de petite taille. À force d'ajouter des éléments dans la liste, on finit par remplir le tableau. Lorsque le tableau est plein, on créé un nouveau tableau de taille deux fois plus grande et l'on recopie les éléments de la liste dans ce nouveau tableau.

Exercice 1. *Première estimation*

1. Écrivez la fonction `inserer(x, t, i)` qui insère l'élément `x` dans le tableau `t` à l'indice `i` (en tenant compte du cas où le tableau est plein et où il faut alors créer un tableau deux fois plus grand et recopier les valeurs).

2. Quelle est la complexité de la fonction précédente dans le pire des cas en fonction de la taille de `t` (on compte comme une opération chaque modification d'une case d'un tableau).

3. En utilisant le résultat de la question précédente, donnez une première majoration de la complexité de n insertions d'éléments dans le tableau, partant d'un tableau de taille 1 ?

Pourquoi (intuitivement) cette majoration est-elle exagérée ?

On va maintenant améliorer la borne sur la complexité à l'aide d'une technique appelée *analyse amortie*. On suppose à partir de maintenant que la taille des tableaux que l'on manipule est toujours une puissance de 2.

Exercice 2. *Méthode par agrégat*

Une première méthode d'analyse amortie consiste à considérer les différentes opérations possibles par paquets et calculer le coût total de l'ensemble, puis diviser pour obtenir le coût « amorti » d'une opération. On exploite le fait que les opérations coûteuses ne peuvent pas avoir lieu s'il n'y a pas eu beaucoup d'opérations faciles à effectuer avant.
Attention : L'analyse amortie est différente de l'analyse « en moyenne ». On calcule bien la complexité d'un paquet d'opérations dans le pire des cas.

1. Quel est le coût réel d'une opération d'insertion ? (il y a deux expressions possibles, selon que le tableau est plein ou non)

2. Calculer le coût total de n opérations d'insertion successives dans un tableau initialement de taille 1.

3. Justifiez le fait qu'on puisse considérer qu'une insertion a un coût constant dans un tableau dynamique.

Exercice 3. *Méthode comptable*

Nous allons ici justifier que le coût d'une insertion dans un tableau dynamique est constant par une seconde méthode appelée *méthode comptable*. L'idée de la méthode comptable est de considérer que certaines opérations simples coûtent un peu plus que leur coût « réel » parce qu'on paie à l'avance une partie du coût des opérations complexes que l'on pourra éventuellement faire plus tard. Dans notre exemple, on va considérer que les opérations d'insertions qui n'agrandissent pas le tableau coûtent 2

opérations de plus, de telle sorte que lorsque l'on doit agrandir le tableau, tout le coût de la copie dans le tableau plus grand a déjà été « payé » à l'avance.

1. Quelle relation y a-t-il entre l'indice i du tableau où l'on va insérer un élément et la taille n du tableau au moment de l'insertion ? (on veut un encadrement de i en fonction de n)

On veut compter dans le coût amorti de toutes les insertions dans le tableau de taille 2^k le coût des déplacements de tous les éléments lors de l'extension de la taille 2^k à la taille 2^{k+1}.
Au moment où il est inséré dans le tableau de taille 2^{k+1} en position $(2^k + i)$, un élément paye à l'avance son coût d'insertion (immédiatement dépensé), son coût de déplacement lors de la prochaine extension et le coût de déplacement de l'élément en position i lors de la prochaine extension.

2. Justifiez que tous les coûts de déplacements nécessaires lors de l'extension suivante du tableau auront bien été payés.

3. Conclure.

Exercice 4. *Et si on supprime ?*

Supposons maintenant que l'on ait également une opération `supprime(t)` qui supprime le dernier élément inséré dans une liste (représentée par un tableau).

1. Si l'on veut réduire la taille du tableau lorsque celui-ci ne contient plus assez d'éléments (pour libérer de la mémoire lorsque la liste redevient petite), à quel moment devrait-on réduire la taille ? (on suppose que l'on garde des tailles en puissances de 2 et que l'on divise donc la taille du tableau par 2)

On suppose que l'on divise la taille du tableau par 2 lorsque le tableau est rempli à moins d'un quart.

2. Montrez par la méthode comptable que le coût amorti d'une opération de suppression est de 3.