IFT339 : STRUCTURES DE DONNÉES (T.P. 3)

DANIEL LEMIRE, PH.D.

1. Introduction et mise en contexte

Une des applications scientifiques les plus courantes est le tracé de courbes unidimensionnelles. En effet, beaucoup d'information vient sous la forme d'un ensemble de valeurs dans le temps : valeur d'un titre à la bourse, température prise à toutes les heures, etc. On dit qu'il s'agit d'une fonction discrète (quand on est matheux) ou d'un signal (quand on est ingénieur).

On n'est pas ici intéressé par le tracé des courbes (d'autres cours sont là pour ça !), mais par leur stockage. Imaginons que nous ayons, par exemple, la valeur du titre de Microsoft à tous les jours pour les 10 dernières années. Cette information est disponible sur un serveur et notre client désire récupérer cette information... mais pas toute cette information ! En effet, notre client a un écran qui fait 640 par 480 pixels. Dans le meilleur des cas, il pourra afficher, en supposant que l'axe du temps soit horizontal, 640 valeurs différentes. Inutile donc de stocker les quelques 3650 valeurs du titre de Microsoft ! On ne va donc échantillonner qu'une valeur sur 10 pour n'en récupérer que 365.

Le client a cependant la possibilité de récupérer progressivement l'information manquante. S'il veut, par exemple, obtenir un échantillonnage d'une valeur sur 5, on lui enverra les valeurs manquantes (environ 365 nouvelles valeurs). Il faut alors placer les nouvelles valeurs au bon endroit dans notre structure de données linéaire.

2. Théorie

À l'aide d'une structure de données linéaire, on veut pouvoir stocker des valeurs échantillonnées de manière à pouvoir raffiner par la suite l'échantillonnage si nécessaire. Nous connaissons au moins deux structures de données qui sont appropriées : le tableau et la liste chaînée. Dans ce qui suit, nous supposerons que nous avons au total $M = 2^k$ valeurs disponibles, mais que nous n'en transmettons que $N = 2^j$ ($j < k$) au départ, doublant à chaque fois l'échantillonnage par la suite.

2.1. **La liste chaînée.** La liste chaînée (list en STL, voir t.p. 2) peut être utilisée pour résoudre ce problème. Il suffit dans un premier temps de remplir la liste avec les N premières valeurs puis d'insérer les nouvelles valeurs avec un itérateur.

2.2. **Le tableau dynamique.** Le tableau peut être utilisé pour stocker les N premières valeurs, puis un nouveau tableau contenant $2N$ valeurs peut être utilisé quand d'autres valeurs sont ajoutées et ainsi de suite.

Date: 10 juillet 2001 (à remettre le 23 juillet 2001 à minuit).

2.3. **Le tableau statique.** L'algorithme avec tableau dynamique implique que l'on doive recopier à chaque fois les valeurs. Cette approche est clairement inefficace. Une meilleure approche serait d'utiliser tout de suite le plus grand tableau (de taille M) et de placer immédiatement les valeurs aux bons endroits et en laissant indéfinies les valeurs intermédiaires.

2.4. **Exemple.** Si nous avons les valeurs suivantes sur le serveur {1, 1, 2, 3, 20, 400, 8000, 2} (symbolisant la valeur du titre de Nortel ou de votre point-com favorite), on aura donc $M = 2^3$ ($k = 3$). On peut choisir $N = 2^1$ ($j = 1$) et les valeurs de départ $\{1, 20\}$, on enverra ensuite les valeurs $\{2, 8000\}$ puis les valeurs $\{1, 3, 400, 2\}$. Le client pourra donc afficher progressivement de plus en plus d'information en s'arrêtant au besoin. Dans l'approche avec liste, nous aurons d'abord un liste contenant $\{1, 20\}$, puis $\{1, 2, 20, 8000\}$ et finalement {1, 1, 2, 3, 20, 400, 8000, 2}. L'approche par tableau dynamique est similaire sauf que l'on doit faire une copie à chaque fois.

3. Énoncé

(60 points) Écrivez un programme capable de lire dans un fichier une suite de nombre (entiers). Les deux premiers nombres lus sont k et j respectivement et vous trouvez ensuite les valeurs. Les valeurs doivent être organisées selon le format de transmission au client. Par exemple, dans l'exemple de la section précédente, le fichier contiendrait les valeurs $3, 1, 1, 20, 2, 8000, 1, 3, 400, 2$ parce que $k = 3$ et $j = 1$.

Le programme doit ensuite écrire dans trois fichiers de destinations la suite réelle des valeurs (soit 1, 1, 2, 3, 20, 400, 8000, 2 dans notre exemple). Le premier fichier contiendra la solution obtenue par la liste, le deuxième fichier contiendra la solution obtenue avec les tableaux dynamiques et le dernier contiendra la solution avec le tableau statique. Vous devez aussi afficher à l'écran les temps de calcul respectif pour chaque approche (omettant l'accès disque).

La syntaxe de l'outil en console devra prendre la forme "PointCom fichierdedonnees.txt fichierliste.txt fichierdynamique.txt fichierstatique.txt".

Vous pouvez utiliser le code qui suit pour mesure le temps de calcul.

```
#include <time.h>
  //
  clock_t start = clock();
  //ma fonction
  clock_t finish = clock();
  double NombreDeSecondes = (double)(finish - start) / CLOCKS_PER_SEC;
```

Question 1. Vous devez comparer les trois approches de façon empirique. Pour ce faire, créez un fichier de données avec au moins 5 valeurs de k assez grandes pour que les résultats soient significatifs (à vous de juger !) et mesurez le temps nécessaire au traitement selon les trois approches. Vous aurez alors au moins 5×3 valeurs que vous devrez mettre dans un tableau. Vous devez commenter et expliquer vos résultats. Il est recommandé de faire un graphique. (30 points)

Question 2. Quel est l'inconvénient majeur de l'approche par tableau statique ? Expliquez. (10 points)

Question 3. (facultative) Pouvez-vous proposer une autre approche qui soit plus efficace ?

4. Correction

Le travail est sur 100 points. Il est permis de travailler en équipe de deux (et de ne remettre qu'un seul rapport par équipe). Il est recommandé, mais pas obligatoire, d'utiliser STL. Vous devez remettre un rapport, votre listing et une soumission électronique sera demandée pour la correction.