IFT339 : STRUCTURES DE DONNÉES (T.P. 4)

DANIEL LEMIRE, PH.D.

1. Introduction et mise en contexte

Nous avons vu dans le cours qu'il existe plusieurs façon de coder une pile ou une file ou n'importe quelle autre structure de données. En particulier, nous avons vu que le STL nous permet de créer diverses sortes de piles ou de files par le biais d'un adapteur sur un contenant (list, vector...).

On peut se demander à quoi tout cela peut bien servir ! Certes, nous avons vu au t.p. 3 que pour un même problème, diverses structures de données et codes pouvaient avoir des performances différentes... mais qu'est-ce qui se passe si la vitesse n'est pas importante ? On peut penser à de nombreuses applications en réseautique où la vitesse de transmission est le facteur limitant.

Il reste alors l'utilisation de la mémoire. Dans ce t.p., nous allons coder une structure de données sur disque.

2. Théorie

On considère l'implémentation d'une pile dans un ordinateur assez limité en mémoire principale (vive), et bien pourvu en mémoire secondaire (disque rigide). Les opérations empiler et depiler doivent être supportées, mais la pile devra pouvoir croître jusqu'à dépasser largement la mémoire principale (vive) disponible, ce qui impose d'en stocker la plus grande partie sur disque.

Conserver la pile entière sur disque est simple quoique peu efficace. On devra donc recourrir à de la mémoire tampon. Pour ce faire, on stocker le N dernières valeurs empilées de telle manière à ce que le code "empiler(1), empiler(2),...,empiler(N), depiler(),...depiler()" ne requiert aucun accès disque (si le nombre de depiler est exactement N).

3. Énoncé

Votre pile (appelée ici PileSurDisque) devra être un gabarit permettant de stocker des éléments arbitraires. Cependant, on peut supposer, pour les fins de ce problème, que votre type est soit "int", soit "double", soit "char" (ce qui vous permettra de tester rapidement votre code et de faire certaines suppositions). La pile devra stocker les N dernières valeurs empilées de manière à réduire les accès au disque (le paramètre N sera donnée au constructeur), mais vous pouvez ignorer cette condition lors de vos premiers essais. Votre pile utilisera un fichier (dont le nom sera donnée au constructeur) pour stocker les éléments reçus. Votre pile devra effectuer les opérations depiler et empiler dans un temps constant (en supposant que le positionnement dans un fichier se fasse dans un temps constant).

Vous devez créer une classe sous la forme...

Date: 22 juillet 2001 (à remettre le 6 août 2001 à minuit).

```
template <class TYPE>
  class PileSurDisque {
  public :
  PileSurDisque(char * fichierbinaire, int N = 0) ; // créer une pile
sur un fichier
  TYPE depiler() ;
  void empiler(TYPE e) ;
  } ;
```

et votre paramètre N devrait être tel que dans le code

```
int main() {
  PileSurDisque<int> psd("test.bin",10) ;
  for(int k = 0 ; k < 10 ; ++k) psd.empiler(k) ;
  for(int k = 0 ; k < 10 ; ++k) psd.depiler() ;
  }
```

la fonction "depiler" ne fera aucun accès un disque.

Votre pile doit être telle que sa consommation de mémoire est constante, peu importe la taille de la pile. Quelques astuces pour vous aider...

(1) On peut déterminer la taille d'un fichier avec les fonctions "mFichier.seekg (0, ios : :end) ;mTaille = mFichier.tellg() ;" si mFichier est un fstream.
(2) On peut se positionner dans un fichier en lecture avec seekg(pos) et en écriture avec seekp(pos). On se place en écriture ou en lecture à la fin du fichier en faisant respectivement seekg(0,ios : :end) et seekp(0,ios : :end).
(3) On peut écrire dans un fichier binaire avec read et write (voir la syntaxe dans les notes de cours). On ouvrir un fichier binaire avec ios : :binary (voir les notes de cours).

Vous devez créer un outil en console nommé PSD qui lit le contenu d'un fichier texte (sous la forme d'une séquence de nombres), le place dans une pile puis dépile le contenu dans un second fichier texte (sous la forme d'une séquence de nombres séparés par des espaces). Vous pouvez supposer qu'il s'agit d'entiers. La syntaxe doit être "PSD entree.txt sortie.txt".

Indice : votre fichier entree.txt pourrait être gigantesque et votre code devrait tout de même fonctionner en autant que vous ayez suffisamment d'espace disque! On supposera toujours que l'on a suffisamment d'espace disque.

Question 1. Donner une application possible au paramètre N (un exemple de programme où cela pourrait être utile). (10 points)

Question 2. Expliquez sommairement comment vous pourriez faire la même chose avec une file (créer une file qui nécessite une quantité constante de mémoire). Est-ce que c'est aussi simple qu'avec la pile? Sinon, pourquoi? (10 points)

Question 3. On a dit que la pile pouvait manquer d'espace disque... que devez-vous faire dans votre code pour traiter ce cas (inutile de le faire dans le code)? (10 points)

4. Correction

Les mêmes conditions que les t.p. précédents s'appliquent. Le t.p. sera corrigé sur 100 points. 70 points seront accordés pour le code.