Files d'attente et suite de Hamming

Christian Rinderknecht

10 juillet 2003

1 Piles et files d'attentes

Une pile est une liste (c'est-à-dire une suite non-bornée d'éléments) dont on limite l'usage aux opérations suivantes :

— la pile est-elle vide ?
— ajouter un élément en tête de la pile ;
— extraire la tête de la pile.

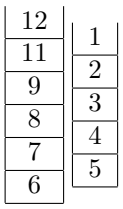
Une file d'attente est une liste dont on limite l'usage aux opérations suivantes :

— la liste est-elle vide ?
— ajouter un élément en queue de la liste ;
— extraire la tête de la liste.

Une file d'attente permet de stocker des informations qui ne peuvent être traitées tout de suite, mais qui devront l'être en respectant l'ordre de leur arrivée.

On souhaite réaliser une file d'attente à l'aide de deux piles : la tête de la première (dite *arrière*) contiendra la queue de la file, et la tête de la seconde (dite *avant*) contiendra la tête de la file. Plus précisément, les nouveaux arrivants seront placés en tête de la liste (pile) arrière, les premiers arrivés seront extraits de la liste (pile) avant. Lorsque la pile avant est épuisée, on retourne la pile arrière, on la place à l'avant et on vide la pile arrière. Cette opération de retournement a une complexité linéaire en la taille de la liste arrière, donc le temps d'extraction du premier élément d'une file ainsi implantée n'est pas constant, mais comme un élément de la file n'est retourné qu'une seule fois, le temps cumulé de n extractions en tête de la file est linéaire en n.

Dans l'exemple suivant, 12 est la queue de la file (donc la tête de la pile arrière) et 1 est la tête de la file (donc la tête de la pile avant) :



La première tâche consiste à concevoir en Java une interface et une classe pour la pile [1] (on souhaite pouvoir y placer des éléments de type quelconque),

1. Vous pouvez aussi chercher une pile dans la blibliothèque standard Java.

puis pour la file. Il faut donc cacher au client de la classe l'implantation à l'aide des deux piles. En revanche, il faut offrir les méthodes qui effectuent les opérations suivantes :

— création d'une file vide ;
— dit si une file est vide ;
— renvoie la longueur de la file ;
— ajoute un élément en queue de la file ;
— retire le premier élément de la file et le renvoie ;
— renvoie le premier élément de la file sans le retirer.

Les deux dernières méthodes devront utiliser des exceptions pour gérer le cas où la file est vide.

2 La suite de Hamming

Un entier de Hamming est un entier naturel non nul dont les facteurs premiers éventuels sont 2, 3 et 5. Le *problème de Hamming* consiste à énumérer les n premiers entiers de Hamming par ordre croissant. Pour cela, on remarque que le premier entier de Hamming est 1 et que tout autre entier de Hamming est le double, le triple ou le quintuple d'un entier de Hamming plus petit (ces cas n'étant pas exclusifs). Il suffit donc d'utiliser trois files d'attente, h_2, h_3 et h_5 contenant initialement le seul nombre 1, puis d'appliquer l'algorithme suivant :

> Déterminer x, le plus petit des trois nombres en tête des files d'attente. Imprimer x, le retirer de chacune des files le contenant et insérer en queue de h_2, h_3 et h_5 respectivement $2x$, $3x$ et $5x$.

1. Programmez cet algorithme en Java et faites afficher les n premiers entiers de Hamming. Observez l'évolution des files d'attente à chaque étape.
2. L'algorithme précédent est très dispendieux car il place la plupart des entiers de Hamming dans les trois files alors qu'une seule suffirait. En effet, si x est un entier de Hamming divisible à la fois par 2, 3 et 5, alors x a été placé dans h_2 au moment où l'on extrayait $x/2$, dans h_3 au moment où l'on extrayait $x/3$ et dans h_5 au moment où l'on extrayait $x/5$. Modifiez votre programme de sorte qu'un même entier de Hamming ne soit inséré que dans une seule des files d'attente.
3. Les entiers Java sont limités à un milliard environ, ce qui ne permet pas d'aller très loin dans la suite de Hamming. Pour pouvoir traiter des grands nombres, on convient de représenter un entier de Hamming x par le triplet (a, b, c) tel que $x = 2^a 3^b 5^c$. Reprendre votre programme avec cette convention et calculer le millionième entier de Hamming. Pour comparer deux entiers de Hamming x et y connus par leurs exposants, il suffit de comparer les réels $\log(x)$ et $\log(y)$. On admettra que la précision des calculs sur les flottants est suffisante pour ne pas induire de comparaison erronée.
4. Déterminer expérimentalement la complexité mémoire de l'algorithme de recherche du n-ième nombre de Hamming. Cette complexité sera mesurée par la somme l des longueurs des trois files h_2, h_3 et h_5. Pour cela on relèvera les valeurs de l pour $n = 1000$, $n = 2000$, $n = 4000$ etc., et vous conjecturerez une relation simple entre l et n.