Fiche de TP4 Tas K-aire et 2 multiplications de 2 entiers à n chiffres

Contenus et objectifs

Comprendre la structure arborescente de Tas K-aire et les 2 multiplications récursives de 2 entiers à N chiffres.
ATTENTION : Pensez à commenter vos programmes Java. Pour le rendu, archivez dans un fichier `Gx_TP4.tgz` ou `Gx_TP4.zip` tous vos fichiers `.java` et téléverser le fichier archive compressé sur Chamilo dans le groupe x dédié.

A. Implémenter la classe `TasK`

En cours, nous avons vu les algorithmes pour construire un Tas binaire (avec K=2) à partir d'un tableau donné d'entiers et d'utiliser ce Tas binaire pour trier. Maintenant, nous souhaitons avoir une généralisation du Tas binaire au Tas K-aire. Le paramètre K est ici un entier supérieur ou égal à 2. Nous considérons toujours un Max-Tas ou Maximier K-aire pour fixer les idées.

Remarque importante : dans cet exercice, il faudra soigner les calculs d'indice dans vos méthodes.

1.1. Ecrivez la classe JAVA `TasK` (avec tous les attributs et méthodes nécessaires) qui permet de manipuler un Tas K-aire à partir d'un tableau donné d'entiers. Par exemple, l'utilisateur peut écrire dans un programme principal :

```
int[] myTable = new int[]{16,4,10,14,7,9,3,2,8,1} ;
int K=3 ;
TasK t=new TasK(K, myTable) ; // t référence un Tas 3-aire
```

1.2. Ajouter dans cette classe `TasK` une méthode qui permet de supprimer le plus grand éléments dans le Tas K-aire courant. Par exemple, dans le programme principal, on peut écrire l'instruction:

```
Int max=t.SuppMax () ;  // l'entier le plus grand est supprimé du Tas 3-aire courant t et sa valeur est retournée. ATTENTION, t doit rester un Tas K-aire après la suppression de l'entier le plus grand.
```

1.3. Ajouter dans cette classe `TasK` une méthode qui permet d'ajouter un nouvel entier dans le Tas K-aire courant. Par exemple, dans le programme principal, on peut écrire l'instruction:

```
t.Ajouter(23) ;  // l'entier 23 est ajouté dans le Tas 3-aire courant t. ATTENTION, t doit rester un Tas K-aire après l'ajout du nouvel entier.
```

1.4. Ajouter dans cette classe `TasK` une méthode qui permet de trier le Tas K-aire courant. Par exemple, dans le programme principal, on peut écrire l'instruction :

```
int[] mySortedTable=t.Trier() ; // myTableTrié référence un
tableau trié dans l'ordre croissant des entiers du Tas K-aire
courant t. ATTENTION, t doit rester un Tas K-aire inchangé
après le tri.
```

B. Implémenter la classe `Mult2n`

Nous souhaitons ici implémenter une classe `Mult2n` qui permet de calculer la multiplication de 2 entiers de n chiffres vu en cours. Nous rappelons ici les 2 algorithmes de calcul.

Soient X et Y deux entiers à n chiffres. Pour simplifier sans perdre de généralité, nous supposons que n est une puissance de 2. On peut alors décomposer X et Y comme suit :

$X = a*10^{n/2} + b$ et $Y = c*10^{n/2} + d$.

Algorithme 1 M2n1(X, Y)

Si X et Y sont des nombres à un chiffre alors retourner X*Y ;
Sinon

Partitionner X en $a*10^{n/2} + b$; Partitionner Y en $c*10^{n/2} + d$;

Retourner $(M2n1(a,c)*10^{n} + (M2n1(a,d)+M2n1(b,c))*10^{n/2} + M2n1(b,d))$;

Fsi.

Remarque : Les opérations de multiplication *, d'addition + et de soustraction - sont celles des opérations arithmétiques classiques.

Algorithme 2 M2n2(X, Y)

Si X et Y sont des nombres à un chiffre alors retourner X*Y ;
Sinon

Partitionner X en $a*10^{n/2} + b$; Partitionner Y en $c*10^{n/2} + d$;

Retourner $(M2n2(a,c)*10^{n}+(M2n2(a+b,c+d)-M2n2(a,c)-M2n2(b,d)*10^{n/2}+M2n2(b,d))$;

Fsi.

Remarque : pensez à optimiser un peu l'algorithme 2

Vous avez le choix des structures de données simples pour gérer les 2 entiers. Testez vos 2 algorithmes sur les 2 entiers 1234 et 2139 et afficher les étapes de calcul les résultats de ces multiplications. Par exemple, l'utilisateur peut écrire dans un programme principal :

```
Mult2n XY= new Mult2n(1234,2139);
XY.M2n1R(); XY.Afficher();
XY.M2n2R(); XY.Afficher();
```

Conseil : N'hésitez pas à afficher toutes les étapes intermédiaires, étape par étape des algorithmes récursifs pendant le calcul pour bien comprendre leur fonctionnement.