NE PAS IMPRIMER CE DOCUMENT

il va être modifié et complété

Document de travail pour le TP2

Vous devez en priorité terminer le TP1

FIBONACCI

A partir de votre classe de manipulation de matrice (*2 lignes-2 colonnes*) et de vecteur (*1 ligne-2 colonnes*) d'entiers, écrivez et testez les divers calculs liés à la suite de Fibonacci vus en TD :

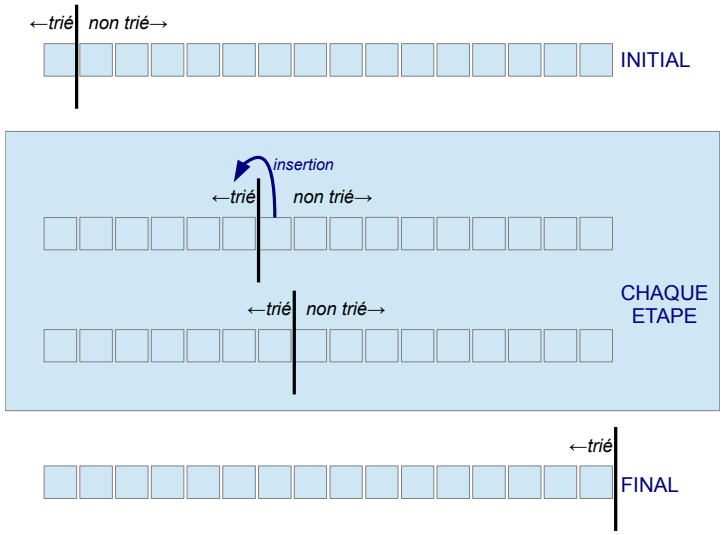
- calcul récursif simple,
- calcul matriciel en utilisant un calcul simple de puissance de matrices (par multiplications successives),
- calcul matriciel en utilisant un calcul de puissance rapide de matrices.

Evaluez chacun de ces calculs et comparez les résultats obtenus. *Veillez à testez vos programmes sur de petites valeurs avant de passer à des valeurs élevées.*

RECHERCHE binôme ayant terminé cet exercice pendant le TP
pour faire un exposé rapide (5 mn maxi) en début du TP suivant sur les évaluations obtenues

TRI NAIF

A partir de votre classe de manipulation de tableaux d'entiers, écrivez le tri naïf d'un tableau.



L'indice de début de la partie non triée du tableau doit être un des paramètres de votre procédure récursive de tri naïf. C'est cet indice qui permet de décider de l'arrêt des appels récursifs. Par exemple, sur un tableau `T` contenant `nb` éléments et avec `deb` comme indice de début de la partie triée :

```
triNaif(T,deb, nb) :
                  incrémentation du compteur d'appels récursifs
                  si deb <= nb
                  alors insérer T[deb] dans le début de T
                          en incrémentant le compteur de permutations
                        triNaif(T,deb+1)
                  fsi

                     appel avec triNaif(T,2,nb)
```

Avec comme algorithme d'insertion :

```
insertion (t, der, nb)
      si der < nb-1
      alors sauv ← t[der+1]
            i ← der
            tant que i >= 0 ET t[i] > sauv
            faire t[i+1] ← t[i]
                  décrémenter i
            ftq
            // ICI  t[i+1] <= sauv ou i<0
            si i < der
            alors // il y a eu des déplacements, il faut remettre sauv en place
                  t[i+1] ← sauv
            fsi
       fsi
```

Vous devez pouvoir évaluer ce tri (nombres de comparaisons et de permutations d'éléments du tableau, nombre d'appels récursifs, évaluation éventuelle du temps d'exécution). Comparez les évaluations du tri sur les trois types de tableaux que vous pouvez créer de façon aléatoire (aucun ordre, croissant, décroissant). *Veillez à testez vos programmes sur des tableaux de petite taille avant de passer à de grands tableaux.*

RECHERCHE binôme ayant terminé cet exercice pendant le TP
pour faire un exposé rapide (5 mn maxi) en début du TP suivant
sur les évaluations obtenues

Pour les exercices suivants, vous trouverez des documentations sur les méthodes graphiques aux adresses :

- *pour les bases (objet graphique, fenêtre, visibilité du dessin, couleur d'écriture, drawline) :*
  http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html
- *pour l'épaisseur de trait (setStroke) :*
  http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html

VON KOCH version 2

Modifiez le programme de Von Koch que vous avez écrit sur la base du schéma fourni : vous devez éliminer la profondeur des appels récursifs qui figure dans la liste de paramètres et choisir entre un tracé simple du segment ou un tracé récursif avec quatre segments en fonction de la distance entre les deux points sur lesquels est fait l'appel.

Pour information :

- vous pouvez garder la structure du programme déjà testé : le paramètre niveau devient la longueur maximim d'appel récursif. Attention cette longueur doit être de type *double*. A l'appel vous devez faire figurer le point décimal (par exemple 100. pour une longueur de *100*) ;
- la longueur du segment d'appel initial dans la première version de Von Koch est d'environ *450* ;
- pour passer en paramètre à votre programme une valeur de type double :

```
public static void main (String[] args)
  { double lg = Double.parseDouble(args[0]);
```

TOURS DE HANOI version graphique

Après avoir programmé la courbe de Von Koch, reprenez votre programme des tours de Hanoï et testez un affichage graphique des déplacements pour un nombre réduit de galets. Utilisez un segment ou un rectangle pour représenter les galets :

- dans un premier temps, utilisez une méthode pour afficher les déplacements de façon textuelle (noms des tours de départ et d'arrivée et flèche), par exemple `D → A` ;
- ajoutez à cette méthode un paramètre qui indique aussi la taille du galet déplacé (pour *n* galets le plus grand a une taille de *n* et le plus petit une taille de *1*), par exemple `3 : D → A` ;
- ajouter un paramètre Graphics à la méthode récursive et à votre méthode qui affiche les déplacements. Vous pouvez représenter les galets par des segments horizontaux qui commencent sur la tour sur laquelle ils sont posés ;
- pour votre affichage, vous devez rendre les traits visibles au fur et à mesure en plaçant l'instruction *setVisible(true)* dans la méthode *init()* ;
- vous pouvez changer la couleur de trait pour rendre plus visible les déplacements (par exemple afficher en rouge le galet qui va être déplacé) et effacer les galets (en les retraçant en blanc) : instruction *setColor()* avec les couleurs : *Color.black*, *Color.white*, *Color.red*, *Color.blue*, etc.
- utilisez des attentes pour ralentir l'affichage (la variable entière *duree* correspond au temps d'attente en millisecondes) :

```
try { Thread.sleep(duree/2);
    } catch (InterruptedException e){};
```