**TP7**

Novembre 2004

# Tp7, les tris et découverte de la lib graphique

Laure Gonnord
`http://www-verimag.imag.fr/~gonnord/`

## Généralités, Objectifs

Le but de ce TP est d'étudier les algorithmes classiques de tri, et aussi de commencer à utiliser la librairie graphique "graphdsu.h", qui est une librairie faite pour vous !

## 1 Préliminaires

**Déclaration de la taille des tableaux**

▶ Déclarer une constante `MAX` qui sera la taille des tableaux à trier.

**Echange de 2 cases** Dans les tris, on échange constamment deux cases d'un tableau.

▶ Ecrire une procedure `void swap(int t[MAX], int i, int j)` qui réalise l'échange des cases $i$ et $j$ du tableau $t$. Si $i$ et $j$ dépassent le tableau, on imprimera un message d'erreur.

**Visualisation d'un tableau**

▶ Écrire rapidement une procédure `void imprime(int t[MAX])` qui permet de visualiser le tableau `t`.

**Initialisation au hasard**

▶ Après avoir copié d'un TP précédent la fonction `hasard`, écrire rapidement une procédure `void initialise(int t[MAX])` qui remplit le tableau `t` avec des nombres tirés au hasard entre 1 et 400.

## 2 Tri-bulle

### 2.1 Tribulle non graphique

On veut faire en sorte qu'après un premier passage, la dernière case du tableau contienne le maximum. Pour ce faire, on fait "remonter ce maximum" à partir de la première case du tableau jusqu'à la dernière case, en échangeant de gauche à droite deux cases adjacentes si elles ne sont pas dans le bon ordre.

Au deuxième tour de boucle, on veut que le deuxième élément plus grand soit rangé dans la $MAX-2$-ième case (l'avant-dernière). Pour ce faire, on réalise la même opération que précédemment, c'est à dire que l'on fait remonter à partir de $T[0]$ le deuxième élément plus grand en échangeant éventuellement deux cases adjacentes. Cette fois, on s'arrête ce processus à la $(n-2)$-ième case. Et on continue ...

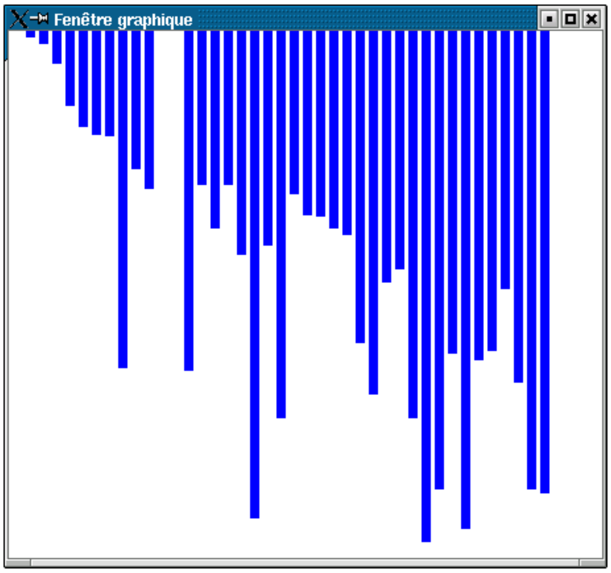
Au $j$-ième tour de boucle on veut placer le ??-ième plus rang élément à la ?? case. Pour cela on le fait remonter de la gauche vers la droite en arrêtant les échanges à la ??ème case.

▶ Coder la procédure `void tri_bulle(int t[MAX])` qui réalise le tri-bulle sur le tableau $t$. Tester cette procédure sur des exemples pertinents.

Remarque 1 Pour une autre explication, se reporter au cahier de TP page 59. Attention, le tri est fait dans l'autre sens.

## 2.2 Tribulle graphique

Pour faire une visualisation graphique du tribulle comme ci-dessous, on se reportera à l'explication pages 60 et 61 du cahier de TP.




# 3 Autres tris

## 3.1 Tri-insertion

On suppose que le sous-tableau $T[0], \ldots T[k-1]$ est trié avec le tri insertion. On désire *insérer* l'élément $T[k]$. De 2 choses l'une :

- ou bien $T[k]$ est plus grand que tous les éléments $T[0], \ldots T[k-1]$, auquel cas il reste à sa place,
- ou bien il doit être inséré dans le sous tableau $T[0] \ldots T[k-1]$.

**Une** solution consiste à décaler l'élément $T[k]$ vers la gauche tant que possible en effectuant des échanges entre voisins.

▶ Coder la procédure `void tri_insertion(int t[MAX])` qui réalise le tri par insertion du tableau $t$. Tester cette procédure.

## 3.2 Tri-sélection

Comme pour le tri-bulle, on désire qu'après un tour, le maximum du tableau soit à sa place. Cette fois par contre, on ne fait qu'un seul échange par boucle, la première fois on échange la case contenant le maximum et la case $MAX-1$ du tableau. Au deuxième tour, on cherche le deuxième élément plus grand (dans les $MAX-1$ premières cases du tableau), et on l'échange à la fin avec la case $MAX-2$. Et ainsi de suite…

▶ Coder la procédure `void tri_selection(int t[MAX])` qui réalise le tri par sélection du tableau $t$. Tester cette procédure.