TP3: Algorithmes de tri

Préambule

L'objectif de cette séance est de se familiariser avec différents algorithmes de tri vus en cours. À la fin de ce TP, vous devriez être capable de mettre en œuvre des algorithmes de tri :

- de base (tri à bulles, tri par sélection, tri par insertion),
- en suivant une description d'une idée.

Quelques informations importantes :

- Le langage de programmation à utiliser est le C.
- Ce TP utilise les signaux (pour interrompre les tris lorsqu'ils dépassent une certaine limite de temps), il doit être fait sous Linux.
- Il est fortement recommandé de **compiler dans le terminal** (en utilisant make).
- Des outils pour tester les fonctions sont fournis dans le fichier zip `tp03_code.zip`.
- Pour chaque exercice, exécutez les tests dans les fichiers fournis pour vérifier le code écrit.

Programme de tests et exercices suggérés par Jean-Claude Georges et adaptés pour le cours.

1 Outils

Les fichiers suivants sont fournis (`tp03_code.zip`) :

- `main.c` – le programme principal
- `testtri.c` et `testtri.h` – les fonctions permettant de tester et chronométrer les fonctions
- `tri.c` et `tri.h` – les fonctions de tri (elles ne sont pas toutes écrites)
- `util.c` et `util.h` – des utilitaires (remplissage et affichage de tableaux, vérification de tri)
- `chrono.c` et `chrono.h` – des fonctions pour chronométrer et interrompre les exécutions
- `makefile` – un fichier permettant de créer l'exécutable en utilisant l'utilitaire `make`

Exercice 1: Prise en main du programme.

1. Copiez tous ces fichiers dans un répertoire.
2. Placez-vous dans ce répertoire.
3. Lancez l'utilitaire `make` : `make`
4. Lancez le programme : `./main`

Le programme affiche alors la configuration actuelle de test de tri, puis propose un menu :

```
--------------- Configuration actuelle --------------

Remplissage          : aleatoire
Taille               : 10000
Tri                  : Tri simple
```

```
Limite de temps      : 2.000 s

                --- Menu ---
        1. Lancer l'e(x)ecution du tri choisi
        2. L(a)ncer l'execution de tous les tris pour la configuration actuelle

        3. Changer de (r)emplissage
        4. Changer de (t)ri
        5. Changer le (n)ombre d'elements du tableau
        6. Changer la (l)imite de temps

        0. (Q)uitter le programme
```

La configuration de départ s'affiche et vous pouvez, sur un tableau de taille et rempli comme indiqué :

- lancer le tri sélectionné : `1` ou `x`
- lancer tous les tris : `2` ou `a`

Lors de l'exécution, si le programme peut trier le tableau dans le temps imparti, il affichera le temps effectivement mis. Sinon, il affichera un message indiquant qu'il n'a pas pu trier le tableau dans le temps limite. Si le tri ne fonctionne pas, un message vous l'indiquera.

Ainsi, si vous entrez `a` , vous obtenez (résultats sur ma machine, les temps peuvent différer) :

```
------------------------------------------------------
Tous les tris d'un tableau aleatoire de 10000 nombres :

        Tri simple                      0.49119 s
        Tri bulle de base               Tri non encore ecrit !
        Tri bulle ameliore              Tri non encore ecrit !
        Tri par selection               Tri non encore ecrit !
        Tri par insertion (seq.)        Tri non encore ecrit !
        Tri par insertion (dicho)       Tri non encore ecrit !
        Tri shell                       0.00271 s
        Tri rapide                      Tri non encore ecrit !
        Tri rapide (shell si n<=6)      Tri non encore ecrit !
        Tri par tas                     Tri non encore ecrit !

------------------------------------------------------
Tapez <Entr> pour continuer
```

Vous pouvez également modifier les paramètres de configuration :

- choisir le mode de remplissage : `3` ou `r`
- choisir un type de tri : `4` ou `t`
- modifier le nombre d'éléments du tableau : `5` ou `n`
- modifier la limite de temps au delà de laquelle le tri s'interrompra : `6` ou `l`

Remarquez que pour le tri simple, si le nombre d'éléments est doublé, le temps de tri est environ quadruplé : ce tri est en $\Theta(n^2)$.

Exercice 2: Lecture des programmes fournis (*). Éditez les différents fichiers fournis et analysez-les. Il n'est pas demandé de comprendre le fichier `chrono.c`.

2 Les tris

2.1 Tri à bulles

Le tri à bulles de base comprend $n-1$ balayages, chacun effectuant au maximum $n-1$ ordonnancements de couples. Le nombre de couples comparées est réduit de 1 après chaque balayage (en effet, le plus grand élément est forcément à sa place).

Exercice 3: Le tri à bulles (*). Écrivez et testez la fonction `tri_bulle_1`, qui exécute le tri à bulles de base.

Une seconde amélioration permettra de terminer le tri si, au cours d'un balayage, on constate qu'aucun couple n'a été réordonné.

Exercice 4: Le tri à bulles amélioré (*). Écrivez et testez la fonction `tri_bulle_2`, qui exécute le tri à bulles amélioré. Vérifiez si l'amélioration accélère le tri dans le cas de tableau trié et de tableau aléatoire.

2.2 Autres tris simples

Exercice 5: Le tri par sélection (*). Écrivez une fonction pour exécuter le tri par sélection (`tri_selection`). Comparez les résultats avec le tri à bulles.

Exercice 6: Le tri par sélection sur PLaTon (P). Modifiez la fonction que vous avez écrit pour résoudre l'exercice PLaTon. Vous devez afficher le tableaux après chaque itération.

Activité PLaTon : TP3

Exercice 7: Le tri par insertion séquentiel (*). Pour son écriture, ce tri nécessite des fonctions auxiliaires : `trouve_place` qui recherche la place d'insertion d'un élément dans un tableau trié et `decale_droite` , qui déplace d'une case vers la fin de tableau les éléments situés entre deux positions.

Écrivez la version séquentielle du tri par insertion.

Exercice 8: Le tri par insertion sur PLaTon (P). Modifiez la fonction que vous avez écrit pour résoudre l'exercice PLaTon. Vous devez afficher le tableaux après chaque itération.

Activité PLaTon : TP3

Exercice 9: Le tri par insertion dichotomique (**). Écrivez la version dichotomique du tri par insertion. Comparez les deux versions entre elles et avec les tris précédents.

3 Pour aller plus loin

Exercice 10: Le tri stupide (**). Mettre en œuvre le tri stupide, qui fonctionne de la manière suivante :

- commencer à la position $i = 0$ du tableau, et répéter tant que $i < n - 1$
- comparer les éléments en position i et $i + 1$, s'ils sont dans le bon ordre, avancer à la position $i + 1$, sinon les permuter et reculer à la position $i - 1$.

Modifiez également le fichier `testtri.c` pour ajouter cet algorithme au menu.