TP : Évaluation expérimentale de trois algorithmes de tri

Objectifs : Ce TP a pour but de compter les nombres d'affectations (a_n) et de comparaisons (c_n) d'éléments d'un tableau pour trier un tableau et de voir comment ces nombres évoluent avec la taille n du tableau et avec l'algorithme utilisé.

Les algorithmes de tri étudiés seront
- le tri par sélection ;
- le tri par insertion ;
- le tri par fusion.

Matériel requis : Pour faire ce TP, il est nécessaire
- d'avoir programmé les trois tris. Il sera supposé dans la suite que vous avez déjà écrit, en suivant les algorithmes décrits en cours, les trois procédures

```
// trie le tableau t
// algo = tri par sélection
procedure tri_select(var t : TABLEAU);

// trie le tableau t
// algo = tri par insertion
procedure tri_insert(var t : TABLEAU);

// trie le tableau t
// algo = tri par fusion
procedure tri_fusion(var t : TABLEAU);
```

  Le type TABLEAU est défini dans l'unité `U_TABLEAU`[1].
- de disposer du logiciel (libre) GNUPLOT (ce qui est le cas dans les salles de TP).

1 Présentation de l'unité `U_TABLEAU`

1.1 L'unité `U_TABLEAU`

Cette unité définit

1. un type nommé TABLEAU dont les éléments sont des entiers

   ```
   type
      ELEMENT = CARDINAL;
   ```

   et dont les indices sont des entiers compris entre 0 et une borne qui dépend de la taille du tableau, taille définie par les fonctions de génération présentées ci-dessous ;
2. une procédure permettant l'affichage d'un tableau

   ```
   // Une procédure pour afficher un tableau
   procedure afficherTableau(const t : TABLEAU) ;
   ```

3. une procédure de lecture des éléments d'un tableau. Le premier nombre à fournir lors de la lecture est la taille du tableau. Doivent suivre ensuite les valeurs succéssives des éléments du tableau

   ```
   // Une procédure pour lire un tableau
   procedure lireTableau(out t : TABLEAU) ;
   ```

4. trois fonctions de génération de tableaux

[1] Cette unité est installée dans les salles de TP et vous n'avez pas à vous en préoccuper. En revanche, si vous voulez l'utiliser en dehors, il vous faut le fichier `U_TABLEAU.pas`

```
// génère un tableau de taille éléments
// avec des entiers choisis aléatoirement entre 0 et max
function genereTableau(const taille,max : CARDINAL) : TABLEAU;

// génère un tableau contenant les entiers de 0 à n-1 dans cet ordre
// (∀k ∈ [[0,n-1]] t[k]=k)
function genereTableauTrie(const n : CARDINAL) : TABLEAU;

// génère un tableau contenant les entiers de n-1 à 0 dans cet ordre
// (∀k ∈ [[0,n-1]] t[k]=n-1-k)
function genereTableauAntiTrie(const n : CARDINAL) : TABLEAU;
```

5. une fonction de copie de tableau

```
// retourne une copie de t
function copieTableau(const t : TABLEAU) : TABLEAU;
```

1.2 Un programme illustrant l'usage de ces unités

Le programme **essaiTableau.pas** listé ci-dessous donne un exemple d'utilisation de certaines des fonctions et procédures de l'unité U_TABLEAU.

```
// auteur : EW
// date   : mars 2006
// objet  : montrer l'utilisation des fonctions et procédures
//          de l'unité U_TABLEAU
program essaiTableau;

uses U_TABLEAU;

var
   t1,t2 : TABLEAU;
begin

   t1 := genereTableau(10,5);
   writeln('Un tableau de 10 entiers choisis aléatoirement entre 1 et 5 : ');
   afficherTableau(t1);

   t2 := copieTableau(t1);
   writeln('Une copie du tableau précédent : ');
   afficherTableau(t2);


   t1[0] := 30;
   writeln('Le premier tableau dont le premier élément est changé : ');
   afficherTableau(t1);

   writeln('La copie n''a pas changée : ');
   afficherTableau(t2);

   t1 := genereTableauTrie(15);
   writeln('Un tableau trié contenant les entiers de 0 à 15-1 : ');
   afficherTableau(t1);

   t1 := genereTableauAntiTrie(20);
   writeln('Un tableau trié dans l''ordre inverse contenant les entiers de 0 à 20-1 : ');
   afficherTableau(t1);

end.
```

2 Compter les affectations et les comparaisons

On souhaite connaître le nombre d'affectations et de comparaisons d'éléments du tableau à trier, éléments de type ELEMENT. On veut donc compter le nombre de fois qu'une instruction de la forme

$x := y$;

est exécutée, et le nombre de fois qu'une expression de la forme

$x <= y$

est évaluée (x et y sont des variable ou expression de type ELEMENT).

Pour cela,

1. on déclare deux variables globales qui doivent être visibles de toutes les procédures et fonctions concernées (on les déclare donc avant ces procédures et fonctions) ;

```
var
  aff  : CARDINAL; // pour compter les affectations
  comp : CARDINAL; // pour compter les comparaisons
```

2. ensuite, dans toutes ces procédures et fonctions, on ajoute l'instruction

```
inc(aff);
```

après chaque affectation qui nous intéresse, et on ajoute l'instruction

```
inc(comp);
```

après toute comparaison ;

3. enfin, le programme principal initialisera à 0 les deux compteurs avant de faire appel à un tri. Par exemple, le programme principal suivant effectue le tri par sélection d'un tableau de n éléments initialisé avec des valeurs choisies au hasard entre 0 et VAL_MAX (constante préalablement définie et fixée à 20000 par exemple), puis affiche les nombres d'affectations et de comparaisons pour trier ce tableau.

```
var
  n : CARDINAL;
  t : TABLEAU;
begin
  write('Nbre d''élts du tableau : ');
  readln(n);
  t := genereTableau(n,VAL_MAX);
  aff := 0;
  comp := 0;
  tri_select(t);
  writeln(aff:10,comp:10);
end.
```

Exercice 1 : Adaptez vos procédures et fonctions et tester le programme principal qui précède.
Q 1 . Que vaut le nombre c_n lorsque $n = 10$?, $n = 20$?, $n = 30$? Les valeurs obtenues dépendent–elles du contenu du tableau ?
Q 2 . Que vaut a_n pour $n = 10$, 20 et 30 ? Les valeurs obtenues dépendent–elles du contenu du tableau ?

Exercice 2 : Modifiez le programme précédent pour calculer a_n et c_n avec un tableau déjà trié, puis avec un tableau trié dans l'ordre inverse. Utilisez les procédures genereTableauTrie et genereTableauAntiTrie de l'unité `U_TABLEAU`.

Exercice 3 : Reprenez les deux exercices précédents pour le tri par insertion, puis pour le tri par fusion.

3 Fabrication de tables de mesures

On va maintenant construire des tables avec les valeurs des nombres a_n et c_n pour n compris entre 2 et une constante TAILLE_MAX que l'on fixera à 512.

```
const
   TAILLE_MAX = 512;
```

Cette table sera produite par un programme qui triera des tableaux de taille n comprise entre 2 et TAILLE_MAX et affichera pour chacune de ces tailles les trois nombres n, a_n et c_n séparés par des espaces et sur une seule ligne (instruction **writeln**(n:3,aff:10,comp:10);).

Voici un extrait de l'affichage produit

```
  2         3         1
  3         6         3
  4         9         6

  ...

510       1527    129795
511       1530    130305
512       1533    130816
```

Exercice 4 : Réalisez un programme nommé `eval_tri_select_qcque.pas` pour le tri par sélection de tableaux remplis d'entiers choisis aléatoirement entre 0 et VAL_MAX (constante préalablement définie et fixée à 20000 par exemple) .

Afin de pouvoir exploiter ultérieurement les nombres a_n et c_n ainsi calculés, il est intéressant de *rediriger* l'affichage produit par ce dernier programme dans un fichier.

Exercice 5 : Tapez la commande

```
$ eval_tri_select_qcque > tri_select_qcque.txt
```

où tri_select_qcque.txt est le nom du fichier dans lequel les nombres seront écrits.

Ouvrez ensuite ce fichier ainsi réalisé avec un éditeur de textes (KATE par exemple), et assurez-vous que le nombre de lignes est égal à la valeur de la constante TAILLE_MAX, et que chacune d'elles contient trois nombres espacés d'espaces.

Exercice 6 : Créez de la même façon les fichiers

1. tri_select_trie.txt
2. tri_select_anti_trie.txt
3. tri_insert_qcque.txt
4. tri_insert_trie.txt
5. tri_insert_anti_trie.txt
6. tri_fusion_qcque.txt
7. tri_fusion_trie.txt
8. tri_fusion_anti_trie.txt

4 Représentations graphiques

Il est possible de faire une représentation graphique des tables numériques ainsi construites. Un programme utile pour cela est le programme GNUPLOT.

GNUPLOT est accessible dans les salles de TP depuis le menu `Démarrer/Programmes/Utilitaires/Gnuplot`. Lorsqu'on lance l'exécution de cette application, il est alors possible d'écrire des commandes afin de produire des graphiques. Nous allons voir quelques unes de ces commandes.

4.1 Représentation graphique de données dans un fichier

La commande permettant de visualiser l'évoluion de a_n en fonction de n pour le tri par sélection de tableaux quelconques est

```
plot 'tri_select_qcque.txt' using 1:2 \
     title 'tri-select tableaux qcques : a(n)' with lines
```

dont voici quelques explications
- plot est la commande de production de graphique ;
- 'tri-select-qcque.txt' est le fichier dans lequel se trouvent les données à afficher (notez les " entourant le nom du fichier) ;
- using 1:2 indique que les points servant à construire le graphique ont leur abscisse dans la première colonne, et leur ordonnée dan la deuxième ;
- title '...' est la légende apportée au graphique ;
- with lines précise que le tracé doit être continu.

(notez l'utilisation du caractère \ pour pouvoir écrire la commande sur plusieurs lignes)

Cette commande donne le graphique illustré à la figure 1

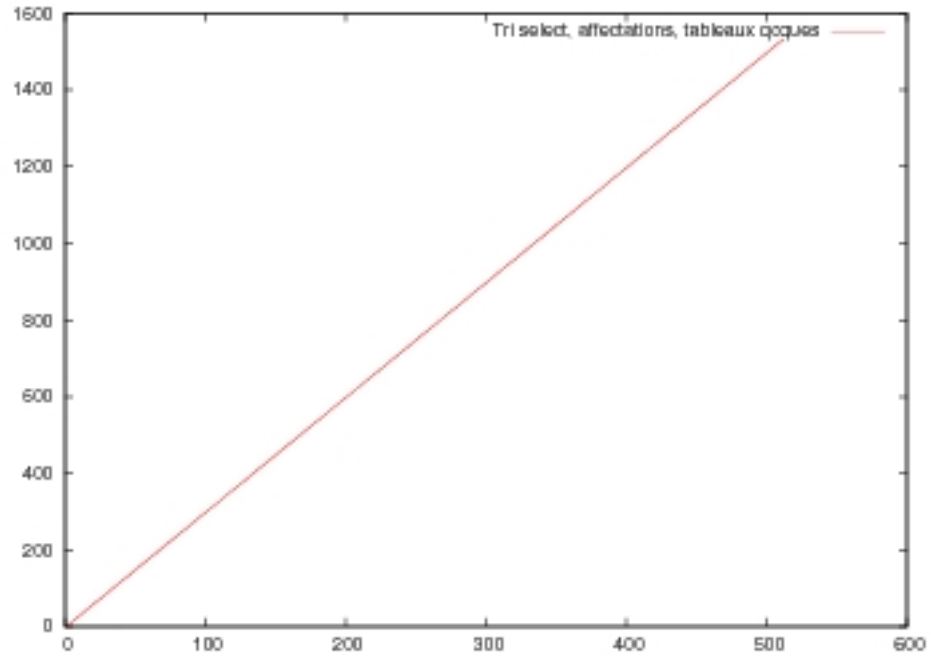

FIG. 1 – Graphique obtenu avec GNUPLOT et la commande : plot 'tri_select_qcque.txt' using 1:2 title 'Tri select, affectations, tableaux qcques' with lines

Exercice 7 : Produisez le graphique du nombre de comparaisons dans le tri par sélection pour des tableaux quelconques.

Même chose avec les autres données.

4.2 Superposition de graphiques

Il est possible de superposer plusieurs courbes sur un même graphique. Voici un exempe de commande superposant la courbe des a_n du tri par insertion pour des tableaux quelconques et des tableaux triés dans l'ordre inverse

```
plot 'tri_insert_qcque.txt' using 1:3 \
     title 'Tri insert, comparaisons, tab qcques' with lines,\
     'tri_insert_anti-trie.txt' using 1:3 \
     title 'Tri insert, comparaisons, tab anti-tries' with lines
```

On peut en suivant ce schéma superposer autant de courbes que l'on veut.

Exercice 8 : **Q 1 .** Superposez les courbes des a_n pour les trois types de tableaux avec le tri par sélection.
Q 2 . Idem avec le tri par insertion.
Q 3 . Idem avec le tri par fusion.

Exercice 9 : Même exercice avec les courbes des c_n.

Exercice 10 : Trouvez un polynôme $P(n)$ du second degré qui approxime au mieux c_n pour le tri par sélection. Vous pourrez superposer les courbes c_n et $P(n)$ avec la commande suivante de GNUPLOT

```
plot 'tri_select_anti-trie.txt' using 1:3 title 'c(n)' with lines,\
     P(x) title 'P(n)' with lines
```

en remplaçant P(x) par son expression.

Exercice 11 : Trouvez un polynôme en n du second degré qui approxime au mieux c_n pour le tri par insertion dans le pire des cas.

Exercice 12 : *Premières conclusions*
Q 1 . Pour les tableaux triés, quel est le moins coûteux des tris ? le plus coûteux ?
Q 2 . Pour le tri par sélection, y a-t-il des distinctions entre les trois sortes de tableaux ?
Q 3 . Pour le tri par insertion, quel est le meilleur des cas ? le pire ? Quel rapport semble-t-il exister entre le nombre de comparaisons pour les tableaux triés dans l'ordre inverse et celui por les tableaux quelconques ?
Q 4 . Que dire du comporetement du tri par fusion relativement aux trois sortes de tableaux ?

5 Étude plus approfondie du tri par fusion

5.1 Calculs des coûts minimaux et maximaux

En cours il a été établi que le nombre c_n de comparaisons dans le tri par fusion d'un tableau de taille n est donné par

$$\begin{aligned} c_1 &= 0 \\ c_n &= c_{\lfloor \frac{n}{2} \rfloor} + c_{\lceil \frac{n}{2} \rceil} + c'_n \quad \forall n \geq 2 \end{aligned}$$

avec les notations
- $\lfloor x \rfloor$ = plus grand entier inférieur ou égal à x (« plancher » de x, ou encore partie entière de x) ;
- $\lceil x \rceil$ = plus petit entier supérieur ou égal à x (« plafond » de x) ;
- c'_n est le coût de la fusion de deux tranches de longueur totale égale à n.

On sait que ce dernier coût est compris entre $\lfloor \frac{n}{2} \rfloor$ et $n-1$.

$$\lfloor \frac{n}{2} \rfloor \leq c'_n \leq n-1$$

On en déduit que c_n sera compris entre deux suites $cmin_n$ et $cmax_n$ définies par les mêmes équations que ci-dessus avec
- $c'_n = \lfloor \frac{n}{2} \rfloor$ pour $cmin_n$;
- et $c'_n = n-1$ pour $cmax_n$.

Exercice 13 : Écrivez une fonction récursive pour calculer $cmin_n$. Utilisez-la pour calculer $cmin_n$ pour n compris entre 2 et 512. Sauvegardez ces valeurs dans un fichier.
Même chose avec $cmax_n$.

5.2 Comparaison avec les mesures effectuées

Exercice 14 : Produisez un graphique superposant les courbes $cmin_n$, $cmax_n$ et c_n pour le tri par fusion de tableaux quelconques.

5.3 Évaluation asymptotique

Exercice 15 : Comparez $cmax_n$ et $n\log_2(n) - n$. En particulier, faites une représentation graphique de $cmax_n - n\log_2(n) + n$. Avec GNUPLOT, la commande pour réaliser cette représentation graphique est (en supposant que le fichier contenant les valeurs de $cmax_n$ se nomme `cmax.txt`)

```
plot 'cmax.txt' \
     using 1:(column(2) - column(1)*log(column(1))/log(2) + column(1)) with lines
```

Comparez aussi $cmin_n$ et $n\log_2(n) - n$.

6 Sauvegarde des graphiques

Il est possible de sauvegarder dans des fichiers les graphiques produits avec GNUPLOT. Il suffit pour cela d'utiliser les deux commandes suivantes préalablement à la commande produisant le graphique.

```
set terminal <format>
set output <fichier>
```

où

- <*format*> est le format de sauvegarde de l'image. Il peut être (entre autres) : *png*, *eps*, *svg*, ...
- <*fichier*> est le nom du fichier de sauvegarde.

Voici un exemple complet de sauvegarde dans un fichier au format *png* nommé `tri_select_aff_qcque.png`.

```
set terminal png
set output 'tri_select_aff_qcque.png'
plot 'tri_select_qcque.txt' using 1:2 \
     title 'Tri select, affectations, tableaux qcques' with lines
```