Tri fusion - Tri rapide

Philippe Langevin

Novembre 2007

Abstract

Le document est basé sur les programmes de Stephen Labbe et Valentin Schmitt. Il s'agit de mettre en évidence les résultat du cours sur le tri fusion et le tri rapide. Le tout est trés sommaire, idéalement l'étudiant devrait compléter la source `tri.tex` pour obtenir un document correctement finalisé.

Contents

1 table.c

Les sources `tri-fusion.c` et `tri-rapide.c` réalisent des mesures des temps de calcul des tris de tableaux aléatoires ou ordonnés. Le fichier `table.c` ci-dessous contient l'ensemble des fonctions communes aux deux sources.

```
[pl@msnet] ./tri-fusion.exe -h
usage : -n nmax -v vmax -check
      : -l nbignes -random
      : -help
```

L'exécution `./tri-fusion.exe -n nmax -v vmax -l nbl -r` mesure le temps d'exécution du tri de `nbl` tables aléatoires (option -r) de tailles inférieures à `nmax` dont les valeurs sont inérieurs à `vmax`.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <math.h>

int *S, check=0, verbose=0, vmax=0, nmax=100, nblignes = 1, randopt=0, pas = 1;

int * randtable( int n ) {
  int i,* res, max;
     res=(int *) malloc(n*sizeof(int));
     if ( ! vmax ) max = n;
     for(i = 0; i < n ; i++)
          res[i]=random() % max;
     return res;
     }

int * sorttable( int n ) {
  int i,* res;
     res=(int *) malloc(n*sizeof(int));
     for(i = 0; i < n; i++ )
          res[i] = i;
     return res;
     }

int verification(int * t, int n) {
     int i;
     for(i=0;i<n-1;i++)
          if(t[i]>t[i+1]) return (0);
     return (1);
     }

void printable(int * T, int n) {
  int i, max;
```

```
  max = n;
  if ( n > 10 ) max = 10;
  printf("\n");
  for(i = 0; i <= max;i++)
          printf(" %d", T[i]);
  if ( n > 10 ) printf(" etc ...");
  }

void  options(int argc ,char*argv[]) {
  int opt; char * optliste = "n:v:l:rch";
  while (( opt = getopt( argc, argv, optliste )) >= 0 ) {
      switch ( opt ) {
       case 'n' : nmax     = atoi(optarg);    break;
       case 'v' : vmax     = atoi(optarg);    break;
       case 'l' : nblignes = atoi(optarg);    break;
       case 'c' : check    = 1;               break;
       case 'r' : randopt  = 1;               break;
       case 'h' : printf("\nusage : -n nmax -v vmax -check");
                  printf("\n      : -l nbignes -random");
                  printf("\n      : -help");
       default  : exit(1);
      }
  }
  pas = nmax / nblignes;
}
```

2 tri-fusion.c

2.1 main

La fonction `main` du programme `tri-fusion.c` est minimale :

```
int main (int argc ,char*argv[])
  {

  options( argc,argv );

  if ( check ) test ();
  if (   randopt ) testranddata();
  if ( ! randopt ) testsortdata();

  return 0;
  }
```

Il fait apparaitre clairement les trois mode d'exécutions du programme : contrôle (`test ()`) , temps de calcul pour les tableaux triés (`testsortdata ()`) ou aléatoires (`testsortdata()`).

Une exécution (mode contrôle) est illustré par la figure 1.

```
./tri-fusion.exe -n 20 -l 5 -c

Verification :
taille:4
 3 2 1 3 0
 1 2 3 3 0
taille:8
 1 7 2 4 1 5 2 3 0
 1 1 2 2 3 4 5 7 0
taille:12
 2 7 11 10 0 6 4 4 11 8 3 etc ...
 0 2 3 4 4 6 7 8 9 10 11 etc ...
taille:16
 6 10 14 3 3 15 9 10 6 2 13 etc ...
 1 2 3 3 3 6 6 7 8 9 10 etc ...
eot...
```

Figure 1: Un exemple d'exécution: `./tri-fusion.exe -n 20 -l 5 -c`

```
./tri-fusion.exe -n 1000000 -l 5 -r

Temps de calcul table aleatoire :
n= 200000 duree= 0.00
n= 400000 duree= 1.00
n= 600000 duree= 1.00
n= 800000 duree= 3.00
facteur cache : 1.189939e-07
```

Figure 2: Un exemple d'exécution: `./tri-fusion.exe -n 100000 -l 5 -r`

Une exécution (mode random) est illustré par la figure 2.

2.2 fusion

```
void fusion(int* T, int p, int q, int r) {
     int i,j,k;
     i=p;
     j=q+1;
     k=p;
     while( ( i <= q ) && ( j <=r ) ) {
          if ( T[i] <= T[j] ) {
               S[k] = T[i];
               i++;
               }
          else {
               S[k] = T[j];
               j++;
               }
          k++;
          }
     while( i <=q ) {
          S[k] = T[i];
          i++;
          k++;
          }
     while( j <=r ) {
          S[k] = T[j];
          j++;
          k++;
          }
     for( k = p; k <= r; k++)
          T[k] = S[k];
     }

void tri_fusion(int* T, int p, int r) {
     int q;
     if ( p < r ){
          q = ( p + r ) / 2;
          tri_fusion( T,  p, q );
          tri_fusion( T, q+1,r );
          fusion(T, p, q, r);
          }
     }
```

3 tri-rapide.c

Le tri rapide se comporte mal sur les instances triées. Un fait illustré par la figure 3, ainsi que par le profil d'exécution (Fig. 4) obtenu avec l'option de compilation `-pg` et la commande de profilage `gprof`.

```
./tri-rapide.exe -n 20000 -l 5 -r

Temps de calcul table aleatoire :
n= 4000 duree= 0.00
n= 8000 duree= 0.00
n= 12000 duree= 0.00
n= 16000 duree= 0.00
facteur cache : 0.000000e+00
```

```
./tri-rapide.exe -n 20000 -l 5

Temps de calcul table en ordre :
n= 4000 duree= 0.00
n= 8000 duree= 1.00
n= 12000 duree= 1.00
n= 16000 duree= 2.00
facteur cache : 6.076389e-09
```

Figure 3: Les temps de calcul de `./tri-raide.exe -n 20000 -l 5 -r` sont beaucoup plus rapide que ceux de `./tri-rapide.exe -n 20000 -l 5 -r`.

3.1 fusion

```
int Partition(int * T, int p, int r)
{
int pivot, j, i;

pivot = T[p];
i = p;
j = r;
        while(T[j]>pivot)
        {
                j=j-1;
        }
        while(i<j)
        {
                Echange(T,i,j);
                do j=j-1;
                while (T[j]>pivot);
                do i=i+1;
                while (T[i]<pivot);
        }
return(j);

}


void tri_rapide (int * T, int p, int r)
{
int q;
        if(p<r)
        {
                q=Partition(T,p,r);
                tri_rapide(T,p,q);
                tri_rapide(T,q+1,r);
```

```
Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00       99     0.00     0.00  Partition
  0.00      0.00     0.00        1     0.00     0.00  options
  0.00      0.00     0.00        1     0.00     0.00  sorttable
  0.00      0.00     0.00        1     0.00     0.00  testsortdata
  0.00      0.00     0.00        1     0.00     0.00  tri_rapide


                        Call graph


granularity: each sample hit covers 2 byte(s) no time propagated

index % time    self  children    called     name
                0.00    0.00      99/99          tri_rapide [5]
[1]      0.0    0.00    0.00      99         Partition [1]
-----------------------------------------------
                0.00    0.00       1/1           main [13]
[2]      0.0    0.00    0.00       1         options [2]
-----------------------------------------------
                0.00    0.00       1/1           testsortdata [4]
[3]      0.0    0.00    0.00       1         sorttable [3]
-----------------------------------------------
                0.00    0.00       1/1           main [13]
[4]      0.0    0.00    0.00       1         testsortdata [4]
                0.00    0.00       1/1           sorttable [3]
                0.00    0.00       1/1           tri_rapide [5]
-----------------------------------------------
                                 198             tri_rapide [5]
                0.00    0.00       1/1           testsortdata [4]
[5]      0.0    0.00    0.00       1+198     tri_rapide [5]
                0.00    0.00      99/99          Partition [1]
                                 198             tri_rapide [5]
-----------------------------------------------


Index by function name

   [1] Partition               [3] sorttable               [5] tri_rapide
   [2] options                 [4] testsortdata
```

Figure 4: Profil d'exécution `./tri-rapide.pg -n 100 -l 1`, récupéré par la commande `gprof -b tri-rapide.pg`. On constate que la fonction partition a été appelée 99 fois, alors qu'en moyenne, le nombre d'appels sur les instances de taille 100 est de l'ordre de $\log_2(100)$.

```
        }
}
```

4 Makefile

```
all : qsort.exe tri-rapide.pg  tri.pdf

tri.pdf : urls.tex data.pdf tri.tex
        pdflatex tri.tex

tri-fusion.exe : table.c tri-fusion.c
        gcc -Wall  tri-fusion.c  -o $@ -lm

tri-rapide.exe : table.c tri-rapide.c
        gcc -Wall  tri-rapide.c  -o $@ -lm

tri-rapide.pg : table.c tri-rapide.c
        gcc -pg  tri-rapide.c  -o $@ -lm

qsort.exe : table.c qsort.c
        gcc -Wall  qsort.c  -o $@ -lm

out  :
        echo "\begin{verbatim}"   > output.txt
        echo "./tri-fusion.exe -n 20 -l 5 -c" >> output.txt
        ./tri-fusion.exe -n 20 -l 5 -c>> output.txt
        echo "\end{verbatim}"     >> output.txt

        echo "\begin{verbatim}"   > outrand.txt
        echo "./tri-fusion.exe -n 1000000 -l 5 -r" >> outrand.txt
        ./tri-fusion.exe -n 1000000 -l 5 -r>> outrand.txt
        echo "\end{verbatim}"     >> outrand.txt

        echo "\begin{verbatim}"   > outraprand.txt
        echo "./tri-rapide.exe -n 20000 -l 5 -r" >> outraprand.txt
        ./tri-rapide.exe -n 20000 -l 5 -r>> outraprand.txt
        echo "\end{verbatim}"     >> outraprand.txt

        echo "\begin{verbatim}"   > outrap.txt
        echo "./tri-rapide.exe -n 20000 -l 5 " >> outrap.txt
        ./tri-rapide.exe -n 20000 -l 5 >> outrap.txt
        echo "\end{verbatim}"     >> outrap.txt
data.txt :
        ./tri-fusion.exe -n 2500000 -l 25 -r | grep duree | sed 's/[a-z]*=//g'> fusion.
        ./tri-rapide.exe -n 2500000 -l 25 -r | grep duree | sed 's/[a-z]*=//g'> rapide.
        join fusion.txt rapide.txt > data.txt

plot    :
        echo "set output \"data.fig\"" > gnuplot.cmd
```

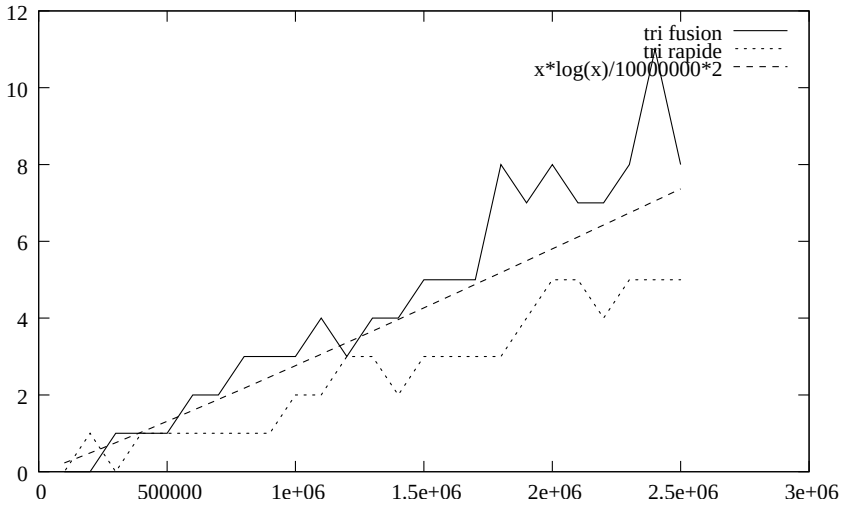

Figure 5: Graphe de temps de calculs.

```
        echo "set term  fig" >> gnuplot.cmd
        echo "set style fill solid 0.75 border -1" >> gnuplot.cmd
        echo "set boxwidth 0.5" >> gnuplot.cmd
        echo "plot 'data.txt' using 1:2 title \"tri fusion\" w lines,\
        'data.txt' using 1:3 title \"tri rapide\"  w lines,\
        x*log(x)/10000000*2" >> gnuplot.cmd

data.fig : gnuplot.cmd data.txt
        gnuplot  gnuplot.cmd
data.pdf : data.fig
        fig2dev -Lpdf data.fig > data.pdf
urls.tex    : urls.sh
        ./urls.sh > urls.tex
```

5 Graphique

Le temps de calcul moyen des fonctions de tri-rapide est asymptotiquement $\Theta(n \log(n))$, ce fait est illustré par le graphique de la figure 5. Le facteur caché, 210^{-7} dans ce cas, dépend de la machine utilisée.

6 qsort

La librairie standard fournit une fonction de tri-rapide `qsort` :

```
NOM      qsort - Trier une table.

SYNOPSIS #include <stdlib.h>
```

```
void qsort (void *base, size_t nmemb, size_t size,
       int (*compar)(const void *, const void *));
```

La fonction `qsort()` trie une table contenant `nmemb` éléments de taille `size`. L'argument base pointe sur le début de la table. Le contenu de la table est trié en ordre croissant, en utilisant la fonction de comparaison pointée par `compar`, laquelle est appelée avec deux arguments pointant sur les objets à comparer. La fonction de comparaison doit renvoyer un entier infeérieur, égal, ou supérieur à zéro si le premier argument est respectivement considéré comme inférieur, égal ou supérieur au second. Si la comparaison des deux arguments renvoie une égalite (valeur de retour nulle), l'ordre des deux éléments est indéfini.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

#include "table.c"


int cmp( const void *x, const void *y)
{ if ( *( (int*) x ) < *( (int*) y ) ) return 0;
  return 1;
}

void  test ( void )
  {
  int n, *T;
  printf("\nVerification : ");
  for ( n = pas; n < nmax; n += pas ) {
    printf("\ntaille:%d", n);
    T = randtable( n );
    printable( T, n);
    qsort(T, n , sizeof(int), cmp );
    printable( T, n);
    if( ! verification( T , n) )
           printf("\nErreur: Tableau non tri  !");
    free( T );
  }
  printf("\neot...\n");
  exit (0 );
  }
```

7 Liens vers les sources

- makefile.sh urls.sh
- data.txt fusion.txt output.txt outrand.txt outraprand.txt outrap.txt rapide.txt

- qsort.c table.c tri-fusion.c tri-rapide.c
- tri.tex urls.tex
- data.fig
- data.pdf

References

[1] Jean-Pierre Zanotti Travaux-Pratiques d'Algorithmique Module I51, licence. http://zanotti.univ-tln.fr/enseignement/I51/TP4.html

[2] Le package "listings" ftp://ftp.inria.fr/pub/TeX/CTAN/macros/latex/contrib/listings/