I6 - TP Complexité et efficacité expérimentale

Licence Math / Info S4 - 2007

Problématique

Le but de ce projet est de réaliser une étude expérimentale comparative de différents algorithmes de tris de tableaux d'entiers (les tris que vous devrez implémenter vous seront présentés dans une planche ultérieure) et de mettre cette étude en relation avec l'analyse théorique de la complexité.

Objectifs éducatifs

- Mettre en pratique les algorithmes de tris étudiés en cours et en td
- Apprendre à écrire des algorithmes et à les implémenter
- Etudier les différences entre complexité théorique et complexité en pratique
- Créer un cadre réel d'expérimentation

Recommandations

Il est préférable d'implémenter chaque algorithme de tri dans un fichier séparé.[1] L'utilisation des pointeurs sur fonction permet d'implémenter de manière élégante les multiples tests. Un pointeur sur fonction se définit de la manière suivante :

```
type  (*pf)(type arg1, ..., type argn);
```

où : `type` est le type de retour de la fonction pointée, `type arg1`, ..., `type argn` sont les types des arguments de la fonction pointée.
Exemple :

```
int f(int i, int j)   /* Définit une fonction. */
{    return i+j; }

int (*pf)(int, int);  /* Déclare un pointeur de fonction. */

int main(void)
{
    int l, m;         /* Déclare deux entiers. */
    pf = &f;          /* Initialise pf avec l'adresse de la fonction f.*/
    printf("Entrez le premier entier : ");
    scanf("%u",&l);   /* Initialise les deux entiers. */
    printf("\nEntrez le deuxième entier : ");
    scanf("%u",&m);
/* Utilise le pointeur pf pour appeler la fonction f
   et affiche le résultat : */
    printf("\nLeur somme est de : %u\n", (*pf)(l,m));
    return 0;
}
```

Types et structures de données :

```
#define MAX 1000000
typedef int tableau[MAX];
```

Fonctions à utiliser pour remplir les tableaux avec des nombres aléatoires :

- .h à inclure : *stdlib.h*
- void srand(unsigned int racine) est une fonction qui initialise le générateur de nombres aléatoires. Elle doit être appelée juste avant la boucle servant à remplir les tableaux, sinon les tableaux contiendront les mêmes nombres.
- int rand(void) est une fonction qui ne prend pas de paramètre et renvoie un entier généré aléatoirement

[1] L'utilisation d'un fichier *makefile* est fortement conseillée.

```
#include time.h
#include stdlib.h
...
  srand(time(0));
  for(i=0;i<tailleTab;i++){
    tab[i]=(unsigned int)rand();
    // tab[i]=((unsigned int)rand()%x); Si on veut un nombre plus petit que x
  }
```

Fonctions à utiliser pour calculer les temps d'exécution :

- librairie à inclure : *time.h*
- définir deux variables de type `clock_t` : `debut` et `fin`
- initialiser `debut` juste avant l'appel à la fonction dont on souhaite calculer le temps d'exécution (`debut=clock()`)
- initialiser `fin` juste après l'appel à la fonction dont on souhaite calculer le temps d'exécution (`fin=clock()`)
- temps d'exécution en secondes : $(t2 - t1) * 1.0/CLOCKS\_PER\_SEC$; la multiplication par 1.0 est ici utilisée pour transformer le résultat en flottant afin de pouvoir disposer des millisecondes. Une autre solution consiste à "caster" le résultat $(t2 - t1)$ en double : $((double)(t2 - t1))/CLOCKS\_PER\_SEC$
  temps d'exécution en ms :
  $(((double)(t2 - t1))/CLOCKS\_PER\_SEC) * 1000.0$

Chaque tris devra être effectué sur au moins 20 tableaux de même taille mais contenant des valeurs différentes générées aléatoirement. Le temps retenu pour une taille donnée sera la moyenne des temps d'éxécution pour les 20 tableaux de cette taille. Une série de test sera stoppée dès lors que son temps d'exécution dépasse 5 min, que les 20 tests soient terminés ou non (la moyenne est exécutée sur le nombre de tris terminés et une indication permettra de se souvenir que la totalité des tests n'a pu être menée à bien). La taille des tableaux variera entre 100 et plus de 1 000 000 d'éléments. Les limitations étant les temps de calculs trop longs. Les tailles des tableaux à tester sont : 100, 500, 5000, 10000, 50000, 100000, 200000, ..., 1000000 (donc 15 tailles de tableau différentes. A raison de 20 tests par taille, si on considère 5 tris, cela représente 1500 tests à lancer $\Rightarrow$ pensez à automatiser les processus de test ! ! !)
La reflexion sur le programme est très importante, une architecture bien conçue permet d'effectuer l'ensemble des tests (pour un tri donné) en une seule exécution du programme. Il sera apporté une attention particulière lors de la notation à l'architecture des programmes.

Evaluation

L'évalutation du tp se fera en deux étapes :

- Entretien individuel de 5 min avec chaque membre de binôme

- Remise d'un rapport contenant
  - algorithmes et études de leur complexités pour les tris non vus en cours ou en TD
  - code source commenté :
    indiquer les tâches réalisées par chaque membre du binôme (notament le nom de l'auteur doit figurer dans chaque fonction C avec les spécifications)
  - pour chaque tris :
    - tableaux et courbes (les tests qui ne termine pas les séries de test devront être indiqués ici)
    - comparaison entre complexité théorique et complexité pratique
  - tableaux et courbes comparatives des tris dit "lents"
  - tableaux et courbes comparatives des tris dit "rapides"
  - tableaux et courbes comparatives entre le plus rapide des tris "lents" et le plus rapide des tris "rapides"
  - Conclusion en fonction des résultats obtenus

  Pour les comparaisons, les courbes seules ne suffisent pas, un commentaire est exigé.