Cycle de Transition
1A S2 INF1032
29/01/2018

ESIEA Paris/Ivry-sur-Seine
Année Académique 2017/2018

Méthodes de Conception d'Algorithmes

C. TRABELSI & E. MARTINS & M. FRANÇOIS

TD2_O - - Optimisation d'algorithmes / Calcul de complexité

EXERCICE 1. (Optimisation d'algorithmes)

Optimiser un algorithme revient souvent à repérer une répétition d'instructions **constantes** ou non **nécessaires**. Considérons l'algorithme suivant :

```
//1 unique affectation, n + 1 tests (y compris le saut de boucle), n additions et n affectations ⇒ 3n + 2
pour compteur allant de 0 à n-1 inclus faire
    Affecter (val1 + val2) à TAB[compteur] //1 addition + 1 affectation répétées n fois ⇒ 2n
fin
```

À l'intérieur de cette boucle on effectue n fois deux instructions : l'addition et l'affectation. Au total la complexité en temps est de $5n + 2$.

► **1.** Proposer une optimisation de cet algorithme, afin d'améliorer sa complexité en temps.

Voici un autre exemple avec une complexité en temps de $7n + 2$:

```
//1 unique affectation, n + 1 tests (y compris le saut de boucle), n additions et n affectations ⇒ 3n + 2
pour compteur allant de 0 à n-1 inclus faire
    //3 comparaisons répétées n fois ⇒ 3n
    si (compteur = 0 OU compteur = n-1) alors
        Afficher * /*1 méta-action répétée conditionnellement ⇒ 1 quand elle est effectuée*/
    fin
    sinon
        Afficher + /*1 méta-action répétée conditionnellement ⇒ 1 quand la précédente n'est pas
                     effectuée*/
    fin
fin
```

► **2.** Proposer une optimisation de cet algorithme, afin d'améliorer sa complexité en temps.

EXERCICE 2. (Algorithme du Tri à bulles)

Récupérer sur `http://learning.esiea.fr` l'archive : `fichiers_tri_a_bulles.tar.gz`. On y trouve le code source du tri à bulles `Tri_a_bulles.c` et les fichiers de test `11.txt`, `1000.txt`, `2000.txt`, etc. qui contiennent respectivement 11 nombres, 1000 nombres, 2000 nombres aléatoires, etc. On y trouve également le fichier `11_solution.txt` qui contient les nombres du fichier `11.txt` déjà triés.

Le programme consiste à :

- récupérer au clavier un entier strictement positif ;
- déclarer un tableau d'entiers dont la taille correspond à l'entier récupéré précédemment ;
- remplir le tableau avec les entiers passés au clavier ;
- trier le tableau avec l'algorithme du Tri à Bulles ;
- afficher le tableau trié en mettant un nombre par ligne.

► **1.** Tester le programme en saisissant quelques nombres au clavier.

► **2.** Lancer le programme avec le fichier `11.txt`, en redirigeant la sortie vers le fichier `ma_solution.txt`

► **3.** Comparer les deux fichiers contenant la solution à savoir `ma_solution.txt` et `11_solution.txt` Vous pouvez utiliser les outils de comparaison de fichiers à savoir `meld` ou bien `diff`.

EXERCICE 3. (Calcul de la complexité d'un programme en C avec `Valgrind`)

Calculer le nombre de cycles d'exécution du programme précédent en utilisant l'outil `callgrind` du logiciel `valgrind` via la commande :

```
$ valgrind --tool=callgrind --dump-line=no --callgrind-out-file=out.txt
./EXEC < 11.txt 2>&1 > ma_solution.txt | grep "Collected" | awk '{print $NF}'
```

Remarque 1 : en réalité `callgrind` compte le nombre de cycles du programme en entier (chargement des données, calcul et affichage) ce qui peut fausser le calcul si le chargement/affichage est plus coûteux que la fonction de calcul elle même.

► **1.** Calculer le nombre de cycles d'exécution **d'une fonction** avec l'outil `callgrind`. On suppose que la fonction s'appelle `Tri_a_Bulles`. Pour cela, il suffit de rajouter les options : `--collect-atstart=no --toggle-collect=Tri_a_Bulles`

► **2.** Calculer le nombre de cycles d'exécution du programme et de votre fonction avec les fichiers `1000.txt`, `2000.txt`, `3000.txt`, `4000.txt`, etc. **Que constatez-vous ?** Compiler également le programme en utilisant les options -O1, -O2, -O3 vues en cours, et tester le sur le fichier `10000.txt`

► **3.** Le script `callgrindPlot.sh` aussi présent dans l'archive vous permet de visualiser via `Gnuplot`, la courbe du nombre de cycles relativement à la taille des fichiers en entrée. D'autres courbes mathématiques ont également été rajoutées. Pour cela lancer le script `callgrindPlot.sh` avec comme argument le nom de votre programme. Le résultat se trouve dans le fichier `sortie.png`.

Remarque 2 : si cela est nécessaire, ne pas oublier dans ce cas de donner les droits d'exécution au script `callgrindPlot.sh`. Ne pas hésiter à ouvrir ce script avec un éditeur de texte, car le code est commenté et compréhensible à votre niveau.