TP1 - Comptage de Schur Sans Parallélisme

Guilherme Dias da Fonseca - Programmation répartie

Important : Lisez le fichier d'informations avant commencer le TP.

1 Objectif

Dans ce TP on veut écrire des logiciels séquentiels pour, étant donné un tableau V avec n éléments de $V[1]$ à $V[n]$, calculer rapidement le nombre de valeurs i, j avec $i \leq j$ telles que

$$V[i] = V[j] = V[i + j].$$

Notez que possiblement $i = j$ et que l'index du premier élément est 1 (réfléchissez pourquoi c'est pas 0). Par exemple, considère le tableau ci-dessous avec $n = 7$.

$V[1]$	$V[2]$	$V[3]$	$V[4]$	$V[5]$	$V[6]$	$V[7]$
0	1	2	0	0	2	2

La réponse pour ce tableau est 2, puis que $V[1] = V[4] = V[5]$ ($i = 1$, $j = 4$ et $i + j = 1 + 4 = 5$) et $V[3] = V[3] = V[6]$ ($i = j = 3$ et $i + j = 3 + 3 = 6$).

Des tableaux de teste sont disponibles sur l'ENT. La taille des tableaux peut être grosse (environs 1 000 000 termes) et l'efficacité du code est très importante. La valeur des éléments ne dépasse pas 255, alors vous devez utiliser un tableau de `byte` pour stocker la suite dans la mémoire (si vous utiliser un `ArrayList` ou même un tableau de `int` le temps d'exécution augmente). Le fichier est textuel, où la première ligne indique la taille du tableau, et les lignes suivantes la valeur de chaque élément. Le nom du fichier est passé au logiciel comme paramètre en ligne de commande.

Il faut coder un algorithme séquentiel pour effectuer le calcul (après on va le paralléliser). On appelle n la taille du tableau (de 1 à n) et c le nombre de valeurs distincts (de 0 à $c - 1$). Souvent on appelle c le *nombre de couleurs*. L'algorithme utilise un compteur x initialisé à 0.

Pour chaque $i, j \geq 1$ tel que $i + j \leq n$, vérifier si $V[i] = V[j] = V[i + j]$ et si oui, incrémenter le compteur x. Il faut faire attention pour arrêter les boucles au moment correct de façon à avoir le code le plus efficace possible. Vérifiez le fonctionnement correct de l'algorithme et notez le temps d'exécution (utilisez la commande `time` sur linux) pour chaque fichier d'entrée.

2 Exécution

Vous pouvez utiliser Eclipse ou Netbeans pour écrire le code, mais il faut l'exécuter en ligne de commande. Allez dans le dossier où on trouve le dossier avec le nom du package qui contient

les fichiers `.class`. Pour exécuter votre code vous devez utiliser :

```
$ java tp1.MainTp1 nomdufichier.schur
```

Pour trouver le temps d'exécution (et l'utilisation de plusieurs cores de la CPU), utiliser :

```
$ time java tp1.MainTp1 nomdufichier.schur
```

Si vous voulez compiler manuellement, vous pouvez utiliser :

```
$ javac tp1/MainTp1.java
```

Sortie de l'outil time : Le **temps réel** est défini comme étant le temps mesuré à partir d'un instant fixé : soit un instant standard du passé, soit un instant (par exemple, le début) de la vie d'un processus (temps écoulé).

Le **temps processus** est défini comme le temps CPU consommé par un processus. Il est parfois divisé en deux parties : le **temps utilisateur** et le **temps système**. Le temps CPU utilisateur est le temps passé à exécuter du code en mode utilisateur. Le temps CPU système est le temps passé par le noyau à exécuter en mode système au nom du processus (par exemple, exécuter les appels système).

3 Curiosité

Une suite de Schur est une suite $v_1, \ldots, v_n$ où il n'y a pas $v_i = v_j = v_{i+j}$ (alors, votre logiciel calcule 0). Un exemple de suite de Schur de $c = 3$ couleurs et $n = 13$ termes est :

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
0	1	1	0	2	2	2	2	2	0	1	1	0

On peut aussi la représenter comme (pas bien pour les daltoniens) :

1 2 3 4 5 6 7 8 9 10 11 12 13

Il n'existe pas de suite de Schur de 3 couleurs et 14 termes. En utilisant 4 couleurs, le maximum sont 44 termes. Personne ne connait la la taille n maximum pour 5 couleurs !