TD n°5

Tri fusion - Comparaison de fonctions

Exercice 1 *Tri fusion.*

Le tri fusion est un algorithme de tri qui repose sur le fait que pour fusionner deux listes/tableaux trié(e)s dont la somme des longueurs est n, $n-1$ comparaisons au maximum sont nécessaires. L'algorithme peut s'effectuer récursivement :
– On découpe en deux parties à peu près égales les données à trier ;
– On trie les données de chaque partie ;
– On fusionne les deux parties.
La récursivité s'arrête à un moment car on finit par arriver à des listes de 1 élément et alors le tri est trivial.

1. Appliquer cet algorithme sur le tableau : | 9 | 0 | 3 | 5 | 8 | 13 | 3 | 2 | 7 |
2. Écrire une méthode statique `int[] fusion(int[] t1, int[] t2)` qui prend en argument deux tableaux triés et retourne un tableau contenant les éléments de `t1` et `t2` triés en ordre croissant.
3. Quels sont les invariants de boucle de la fusion ? Y a-t-il des pré-conditions requises ? Prouver la correction de l'algorithme et sa terminaison.
4. Avant d'écrire la méthode `triFusion`, dessinez l'arbre des appels récursifs pour le tableau-exemple ci-dessus. Combien y en a-t-il ? Quelle estimation pouvez-vous faire dans le cas d'un tableau de taille n ?
5. Écrire la méthode `int[] triFusion(int[] tab)` qui effectue les tâches suivantes : séparation de `tab` en deux, appel récursif puis fusion des deux tableaux obtenus. Vous pourrez, si vous le désirez, écrire une fonction `int[] sousTableau(int[] tab, int d, int f)` qui retourne le sous-tableau de `tab` compris entre les indices `d` et `f`.

Exercice 2 *Examples de complexité.*

En algorithmique, on s'intéresse aux ordres de grandeur du temps d'exécution (et/ou de l'espace mémoire) nécessaire à un algorithme en fonction de la taille des données en entrée n. Dans cet exercice on considère plusieur méthodes et on cherche d'estimer le nombre d'istrutions qu'ils exécutent.

1. Considerez la méthode suivante :

```
public static int[][] example(int n) {
        int[][] tab = new int[n][n];

        for (int i = 0; i<n; i++){
                tab[i][i] = i;
        }
        return tab;
}
```

Qu'est-ce qu'elle fait ? Combien d'iterations fait la boucle ? Combien d'instructions sont exécutées en fonction de n ?

2. Considerez la méthode suivante :

```
public static int[][] example2(int n) {
        int[][] tab = new int[n][n];

        for (int i = 0; i<n; i++){
                for (int j = 0; j<n; j++)
                tab[i][j] = j;
        }
        return tab;
}
```

Qu'est-ce qu'elle fait ? Combien d'instructions sont exécutées en fonction de n ?

3. Considerez la méthode suivante :

```
public static int[][] example3(int n) {
        int[][] tab = new int[n][n];

        for (int i = 0; i<n; i++) {
                for (int j = 0; j<n; j++) {
                        tab[i][j] = j;
                }
        }

        for (int i = 0; i<n; i++) {
                tab[i][i] = 0;
        }
        return tab;
}
```

Qu'est-ce qu'elle fait ? Combien d'instructions sont exécutées en fonction de n ?

4. Considerez la méthode suivante :

```
public static int[][] example4(int n) {
        int[][] tab = new int[n][n];

        for (int i = 0; i<n; i++) {
                for (int j = i; j<n; j++) {
                        tab[i][j] = j;
                }
        }
        return tab;
}
```

Qu'est-ce qu'elle fait ? Combien de fois elle écrit sur `tab` ? Donner une formule. Estimez combien d'instructions sont exécutées en fonction de n.

5. Considerez la méthode suivante :

```
public static int[] example5(int n) {
        int[] tab = new int[n];
        i = n-1;

        while (i != 0) {
                tab[i] = i;
                i = i/2;
        }
        return tab;
}
```

Qu'est-ce qu'elle fait ? Combien de fois elle écrit sur `tab` si $n = 16$? pour $n = 32$? pour $n = 2^k$ (avec $k > 0$) ? Estimez combien d'instructions sont exécutées en fonction de n.

6. Considerez la méthode suivante :

```
public static int[][] example6(int n) {
        int[][] tab = new int[n][n];
        i = n-1;

        while (i != 0) {
                for (int j = 0; j<n; j++) {
                        tab[i][j] = j;
                }
                i = i/2;
        }
        return tab;
}
```

Qu'est-ce qu'elle fait ? Estimez combien d'instructions sont exécutées en fonction de n.

Exercice 3 ***Comparaison de fonctions.*** En algorithmique, il y a une notation precise pour exprimer la complexité d'un programme. Si $f(n)$ est la fonction qui exprime le nombre d'istruction d'une méthode donné on note :

- $f(n) = O(g(n))$ s'il existe une constante $c \geq 0$ tel que $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$. L'intuition est que si $f(n) = O(g(n))$ alors f est bornée, par le dessus, par g (asymptotiquement).
- $f(n) = \Omega(g(n))$ s'il existe une constante $c > 0$ tel que $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ ou si $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$. L'intuition est que si $f(n) = \Omega(g(n))$ alors f est bornée, par le dessous, par g (asymptotiquement).
- $f(n) = \Theta(g(n))$ si $f(n) = O(g(n))$ et $f(n) = \Omega(g(n))$.

Et quelques propriétés de $O(n)$:

- Si $f_1(n) = O(g_1(n))$ et $f_2(n) = O(g_2(n))$ alors $f_1(n) + f_2(n) = O(max\{g_1(n), g_2(n)\})$.
- Si $f_1(n) = O(g_1(n))$ et $f_2(n) = O(g_2(n))$ alors $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.

1. Quel est la condition sur $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ pour definir directment $f(n) = \Theta(g(n))$?
2. Est-ce que $1 = O(7)$? $7 = O(1)$?
3. Est-ce que $7n + 5 = O(n)$? $7n + 5 = O(n^2)$? $7n + 5 = \Omega(n)$? $7n + 5 = \Omega(n^2)$?

4. Est-ce que $7n^2 + 3n = \Theta(n^2)$?
5. Est-ce que $\sqrt{n} = O(n)$? $\sqrt{n} = \Omega(n)$? $n \cdot \sqrt{n} = O(n^2)$?
6. Est-ce que $n = O(2^n)$? $n = \Omega(2^n)$? $n^{100} = O(2^n)$?
7. Est-ce que $3^n = O(2^n)$? $2^{n+1} = O(2^n)$? $2^{2n} = O(2^n)$?

Rappel sur le logarithme :
- $\log_2 x$ est la fonction qui à x associe la puissance à laquelle il faut élever 2 pour trouver x, c'est-à-dire tel que $2^{\log_2 x} = x$.
- Plus en general $\log_a x$ est la fonction qui à x associe la puissance à laquelle il faut élever a pour trouver x, c'est-à-dire que tel que $a^{\log_a x} = x$.
- On a $\lim_{n\to\infty} \frac{n^\alpha}{\log_b x} = +\infty$ si $\alpha > 0$ et $b > 0$.
- On a $\log_b x = k \cdot \log_a x$ pour à certain $k > 0$. Precisement : $\log_a x = \frac{1}{\log_a b} \cdot \log_b x$, c'est-à-dire $k = \frac{1}{\log_a b}$.
- On a $\log_a(x \cdot y) = \log_a(x) + \log_a(y)$ et plus en general $\log_a(x^k) = k \cdot \log_a(x)$.

Questions :

1. Est-ce que $\log_2(n^2) = O(\log_2(n))$? $\log_2(n^2) = O(\log_2(n^7))$? $\log_2(n^2) = O(\log_3(n))$?
2. Est-ce que $n^{\log_n 3} = \Omega(\sqrt{n})$? $n^{\log_3 n} = O(n \cdot \sqrt{n})$?
3. Est-ce que $(\log_2 \sqrt{n})^3 = O(\log_2 n)$?
4. Est-ce que $3^{\log_9 n} = O(n \cdot \sqrt{n})$?
5. Est-ce que $\log_2 \log_2 n = O(\log_2 n)$? $\sqrt{2}^{\log_2 n} = O(n \cdot \log_2 n)$?
6. Pour chaque méthode de l'exercice 2 trouvez la fonction $g(n)$ telle que la méthode est un $\Theta(g(n))$.