Université d'El Oued
Département d'Informatique

Complexité des algorithmes
Correction de l'Examen du 08 Janvier 2024

Question 1 (6 pts):

Fonction	**Notation asymptotique en O**	Classe
$f_1(n) = \frac{1}{2}n^2 + 3n + 11$	$O(n^2)$	Quadratique
$f_2(n) = 2n^3 + n^3 \log_3(n)$	$O(n^3 \log n)$	$n^3 \log n$
$f_3(n) = n \log_4(5n^2)$	$O(n \log n)$	$n \log n$
$f_4(n) = 5^n + 5^{\log_2(n)}$	$O(5^n)$	Exponentiel
$f_5(n) = \log_2(\sqrt{n}) + \log_2(2n)$	$O(\log n)$	Logarithmique

Ordre : $f_5 \ll f_3 \ll f_1 \ll f_2 \ll f_4$

Question 2 (4 pts):

Le tri par fusion utilise le principe "diviser pour régner"

Méthode

- On divise le tableau en deux
- On trie les deux sous-tableaux
- On produit un tableau trié contenant tous les éléments des deux sous-tableaux triés

La complexité de ce tri est donnée par l'équation suivante :

$$c(n) = 2 \times c(n/2) + tfusion(n)$$

où $tfusion$ est le coût de l'algorithme d'interclassement

$$tfusion(n) = n - 1 \text{ comparaisons}$$

Problème (10 pts)

Algorithme naïf

a) Le scénario pire cas : pour tous les alignements possibles de $motif$ avec $source$ à partir de la position i dans $source$, $motif$ privé de son dernier caractère, i.e., $motif[0..m-2]$, est identique au sous-mot $source\ [i..(i+m-2)]$; uniquement le dernier caractère de motif n'est pas identique au caractère $source\ [i+m-2]$.
Dans ce cas, l'algorithme fait pour chaque étape (alignement) m comparaisons de caractères. Donc, pour $(n - m + 1)$ alignements, l'algorithme fait $(n-m+1) \times m$ comparaisons.

b) Un exemple de $source$ et $motif$ sur un alphabet $A = \{a, b\}$ pour ce scénario est donné par :

$source$ $(n = 10)$	a	b	a	b	a	b	a	b	a	b
$motif$ $(m = 3)$	a	b	b							

Algorithme récursive

a) Equation récursive de $Est\_Prefixe$

$$C_{EP}(n,m) = \begin{cases} 0 & m = 0 \\ 1 + C_{EP}(n-1, m-1) & m > 0 \end{cases}$$

La solution de cette équation est : m
La complexité au cas pire est $O(m)$.

b) Equation récursive de $Recherche\_Recursive$

$$C_{RR}(n,m) = \begin{cases} 0 & n < m \\ C_{EP}(n,m) + C_{EP}(n-1,m) & m > 0 \end{cases}$$

La solution de cette équation est : $(n - m + 1) \times m$
La complexité au cas pire est $O(nm)$.
En comparaison avec l'algorithme 1, cette réalisation a une complexité similaire.