Rapport SAE 2.02 : Comparaison d'approches algorithmiques

Table des matières

I. Fonctionnement des programmes

Programmes	Problèmes de code	Problèmes de résultat
Efficacité 117 (python)	Pas de problème	Ne fonctionne pas si juste 1 espace
Efficacité 124 (java)	Ne retourne pas de chaîne, juste un affichage. Ne compile pas	Ne fonctionne pas si 4 espaces
Efficacité 151 (python)	Pas de problème	Pas d'erreur
Simplicité 84 (java)	Pas de problème	Pas d'erreur
Simplicité 103 (java)	Pas de problème	Mange 1 espace quand 3 ou 7 espaces
Simplicité 116 (java)	Pas de problème	Pas d'erreur
Simplicité 139 (java)	Que le main, affichage seulement. Ne compile pas	Mange 1 espace quand 3 ou 7 espaces
Sobriété 76 (c)	Pas de problème	Ne fonctionne pas si chaine vide
Sobriété 92 (java)	Pas de problème	Oubli d'un espace
Sobriété 102 (python)	Pas de problème	Pas d'erreur

Les programmes qui fonctionnent entièrement selon mes tests sont : le 151, le 84, le 116 et le 102.
Les programmes 117, 103, 76 et 92 compilent mais ont eu des erreurs lors de mes tests.
Les programmes 124 et 139 ne compilent pas et, une fois corrigés, ont des erreurs.

Les tests effectués ont pour nom « test_algos ».

II. Efficacité

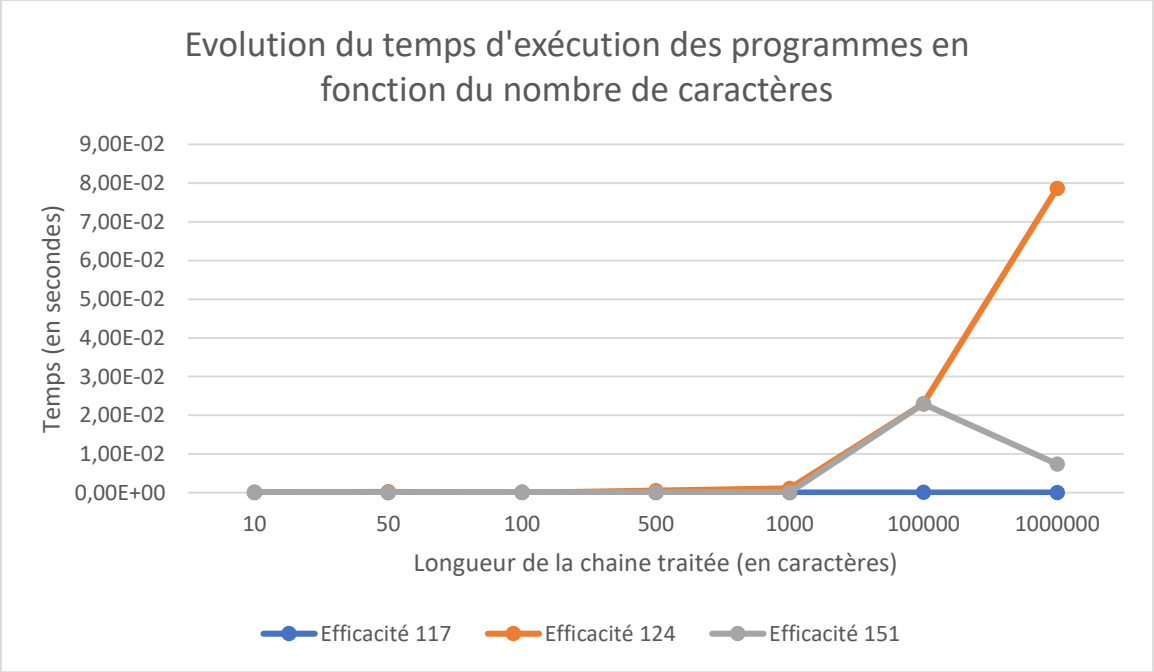
1. Temps d'exécution

Pour calculer l'efficacité des algorithmes, j'ai calculé leur temps d'exécution. Cette donnée reste variable en fonction de l'IDE utilisé pour les tests (différent en Java et en Python). Cela permet toutefois d'avoir une idée de l'évolution du temps d'exécution en fonction de la longueur de la chaine à traiter. Le nom des tests effectués ici est « test_temps ». Les chaînes utilisées sont les mêmes pour chaque algorithme.

Tableau du temps d'exécution (en secondes) des 3 algorithmes « Efficacité » en fonction du nombre de caractères de la chaîne traitée

Nombre de caractères / Algorithme	10	50	100	500	1000	100000	1000000

Efficacité 117 (python)	2.49 e-05	3.19 e-05	7.619 e-05	2.5 e-05	1.659 e-05	4.509 e-05	3.169 e-05
Efficacité 124 (java)	0.0000842	0.0002336	0.0002493	0.0005049	0.0010979	0.022932	0.078633
Efficacité 151 (python)	5.12 e-05	0.00011500000 25	7.39 e-05	9.28 e-05	0.0001034 0	0.00078	0.00739



Selon ces tests, le 124 est le moins efficace. Son temps d'exécution est raisonnable pour de courtes chaînes de caractères, mais grandit très rapidement lorsque la chaîne devient plus importante. Il en va de même pour le 151, même si son augmentation en terme de temps est beaucoup plus faible. Enfin, le plus rapide semble être le 117, dont le temps est constant pour les chaînes utilisées.

2. Complexité

Le 117 est de complexité constante, alors que le 124 et le 151 sont de complexité exponentielle.

III. Simplicité

Pour évaluer la simplicité des programmes, j'ai utilisé Codacy.

Résultats pour le 103

Size		Structure		Complexity	
Lines of code:	43	Number of Methods:	0	Complexity:	6
Source lines of code:	29	sLoC / Method:	0	Complexity / Method:	0
Commented lines of code:	0				

Résultats pour le 84

Size		Structure		Complexity	
Lines of code:	15	Number of Methods:	0	Complexity:	5
Source lines of code:	12	sLoC / Method:	0	Complexity / Method:	0
Commented lines of code:	0				

Résultats pour le 116

Size		Structure		Complexity	
Lines of code:	39	Number of Methods:	0	Complexity:	7
Source lines of code:	19	sLoC / Method:	0	Complexity / Method:	0
Commented lines of code:	13				

Résultats pour le 139

Size		Structure		Complexity	
Lines of code:	29	Number of Methods:	0	Complexity:	1
Source lines of code:	19	sLoC / Method:	0	Complexity / Method:	0
Commented lines of code:	2				

Comme on peut le voir, selon Codacy, le 139 est le moins complexe (1), loin devant le 84 (5), le 103 (6) et le 116 (7). Le 103 est le plus long (43 lignes de code), suivi du 116 (39), du 139 (29) et du 84 (15). Le 116 et le 139 sont les seuls à être commentés, le 116 l'étant beaucoup que le 139 (13 lignes commentées contre 2).

A titre personnel, je trouve que les plus simples sont le 84, car il est le plus court et compact, et le 116, qui est lui aussi assez court mais est en plus très bien commenté.

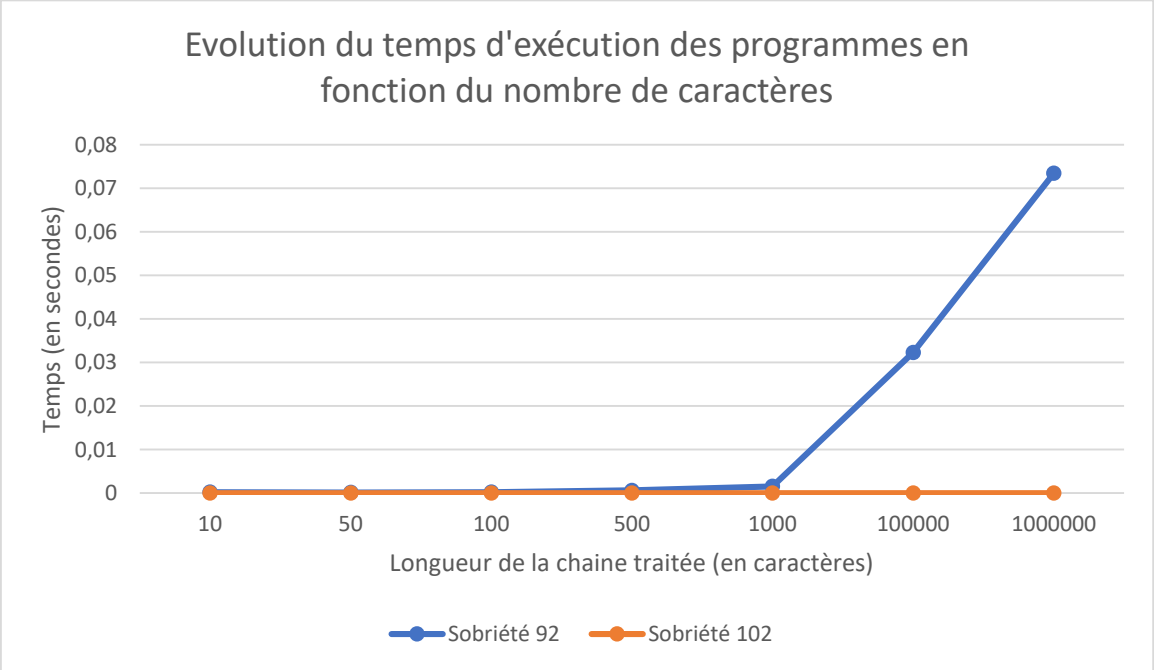
IV. Sobriété

1. Temps d'exécution

Pour calculer la sobriété des algorithmes, j'ai calculé leur temps d'exécution, comme pour l'efficacité. Cette valeur peut varier selon l'IDE est utilisé, mais demeure toutefois pertinente pour évaluer la sobriété de l'algorithme.

Tableau du temps d'exécution (en seconde) des 3 algorithmes de la catégorie « Sobriété » en fonction du nombre de caractères de la chaîne traitée

Nombre de caractères / Algorithme	10	50	100	500	1000	100000	1000000
Sobriété 76 (c)	Je ne suis pas parvenue à réutiliser mes fichiers texte de test en tant que chaîne de caractères.						
Sobriété 92 (Java)	0.0002061	0.0001631	0.0002209	0.0006241	0.0015514	0.032285	0.073455
Sobriété 102 (Python)	2.15 e-05	1.49 e-05	3.87 e-05	1.9 e-05	1.8 e-05	2.8 e-05	1.74 e-05



Selon ces tests, pour des petites chaînes, ces 2 programmes sont aussi rapides. Cependant, dès lors que la taille des chaînes commence à fortement augmenter, le temps d'exécution du programme 102 augmente considérablement, contrairement à celui du 92, qui reste stable.

Les tests se nomment « test_temps ».

2. Ressources

Calcul de la quantité de mémoire utilisée

Nombre de caractères / Algorithme	10	50	100	500	1000	100000	1000000
Sobriété 76 (c)	Je n'ai pas trouvé de moyen d'évaluer la sobriété						
Sobriété 92 (Java)	0	0	0	0	0	0.48 Mb	9.43 Mb
Sobriété 102 (Python)	Je n'ai pas trouvé de moyen d'évaluer la sobriété						

J'ai eu du mal à calculer les ressources utilisées par les algorithmes. Je n'ai pu le faire que pour le programme Java. On constate que la quantité de mémoire utilisée augmente fortement avec le nombre de caractères de la chaîne traitée. Le nom du test Java est « test_ressources ».

3. Complexité

Le 102 est de complexité constante alors que le 92 et le 76 sont de complexité exponentielles.

V. Synthèse

Efficacité			
Algorithme	Position	Note	Raison
117	2	1	Le plus efficace, mais erreur lors des tests
124	3	0	Ne compile pas
151	1	5	Pas forcément le plus efficace, mais n'a pas d'erreur

Simplicité			
Algorithme	Position	Note	Raison
103	3	1	Erreurs lors des tests
84	1	5	Le plus court et l'un des plus simples
116	2	4	L'un des plus court et de splus simples, mais plus complexe que le 84 selon Codacy
139	4	0	Ne compile pas

Sobriété			
Algorithme	Position	Note	Raison
76	2	1	Erreur lors des tests. Sobriété non calculée
92	2	1	Erreur lors des tests
102	1	5	Fonctionnel