Correia Mendes Leonardo

SAE 2.02
Livrable Rapport

Table des matières

Introduction

L'objectif de cette SAÉ est d’approfondir la réflexion sur l’approche algorithmique des problèmes rencontrés pendant les phases de développement. Cette activité vise à enrichir nos compétences en programmation et en analyse critique, tout en nous offrant l'opportunité de nous familiariser avec divers aspects de l'algorithmique dans un contexte pratique et compétitif.

Classement de la catégorie Efficacité

Pour une raison que j’ignore je n’arrivais à lier Codacy à mon GitHub. Un ami m’a alors aidé en mettant tous mes algos sur son Codacy et en me donnant leur score. Pour chaque complexité donnée par Codacy je l’ai divisé par 2 pour soustraire à la note (ex. Si la note est de 15 et la complexité Codacy de 5 alors je divise 5 par 2 ce qui est égale à 2.5 et je soustrait 2.5 à 15. Note finale = 12 .5 et donc arrondie à 13.

5ème place

28.efficacite.c

Cet algo n’était pas hors concours et je ne soupçonne pas l’utilisation de ChatGPT.

Celui-ci compilait bien et était anonyme mais ne fonctionnait pas et par conséquent ne passait pas les test. Je l’ai noté sur une base de 5 ducoup. Codacy ne m’a malheuresement pas donné sa complexité mais les algos non récursifs m'ont l'air d'être d'une compléxité de O(n) et les récursifs sont de O(n*k) avec n la taille de la chaine et k le nombre d'itérations. Je pense que l'algo aurait pu être amélioré alors j'ai retiré un point.

Note finale : 4/20

4ème place

61.efficacite.c

Cet algo n’était pas hors concours et je ne soupçonne pas l’utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test ne passaient pas tous (ceux de RLE avec itérations) alors je l’ai noté sur une base de 10. Codacy ne m’a aussi malheureusement pas

donné sa complexité mais les algos non récursifs m'ont l'air d'être d'une complexité de O(n) et les récursifs sont de O(n*k) avec n la taille de la chaîne et k le nombre d'itérations. Comme cet algo passait mes test supplémentaires (9W4W au lieu de 13W pour RLE) alors j'ai décidé de laisser la note à 10.

Note finale : 10/20

3ème place

16efficacite.java

Cet algo n'était pas hors concours et je ne soupçonne pas l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme, fonctionnait et tous les test passaient alors je l'ai noté sur une base de 20. Codacy m'indiquait une complexité de 14 alors j'ai retiré 7 points (14 / 2).. Les algos non récursifs m'ont l'air d'être d'une complexité de O(n) et les récursifs sont de O(n*k) avec n la taille de la chaîne et k le nombre d'itérations.

Note finale : 13/20

1ère place ex æquo

27efficacite.py

Cet algo n'était pas hors concours et je ne soupçonne pas l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme, fonctionnait et tous les test passaient alors je l'ai noté sur une base de 20. Codacy m'indiquait une complexité de 5 alors j'ai retiré 2.5 points (5 / 2).. Les algos non récursifs m'ont l'air d'être d'une complexité de O(n) et les récursifs sont de O(n*k) avec n la taille de la chaîne et k le nombre d'itérations. Cependant je pense que l'algo pourrait être un peu amélioré donc j'ai retiré 0.5 points

Note finale : 17/20

59efficacite.java

Cet algo n'était pas hors concours et je ne soupçonne pas l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme, fonctionnait et tous les test passaient sauf les miens, alors je l'ai noté sur une base de 18. Codacy m'indiquait une complexité de 5 alors j'ai retiré 2.5 points (5 / 2).. Les algos non récursifs m'ont l'air d'être d'une complexité de O(n) et les récursifs sont de O(n*k) avec n la taille de la chaîne et k le nombre d'itérations. Cependant je pense que l'algo

pourrait être un peu amélioré donc j’ai retiré 1.0 point. Ce qui nous amène à une note de 16.5 que j’ai arrondie à 17.

Note finale : 17/20

Tableau catégorie Efficacité

Algos/Places	1ère place	2ème place	3ème place	4ème place	5ème place	Note
59efficacite.java	X	-	-	-	-	17/20
27efficacite.py	X	-	-	-	-	17/20
16efficacite.java	-	-	X	-	-	13/20
61efficacite.c	-	-	-	X	-	10/20
28efficacite.c	-	-	-	-	X	4/20

Classement de la catégorie Simplicité

Pour noter cette catégorie, seulement ma subjectivité me l’a permis.

5ème place

19simplicite.java

Cet algo n’était pas hors concours et je ne soupçonne pas l’utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test ne passaient pas tous car il a fait l’algo de RLE seulement. Alors je l’ai noté sur une base de 10 et j’ai retiré 2.5 points par algo manquant (2.5 x 3 = 7.5). Le code est très simple et compréhensible. De plus, même dans l'algo réalisé il y des problèmes avec les test donc - 1.0 point. Ce qui nous amène à une note de 1.5 que j’ai arrondie à 2.

Note finale : 2/20

4ème place

64simplicite.java

Cet algo n’était pas hors concours et je ne soupçonne pas l’utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test ne passaient pas alors je l'ai noté sur une base de 10. Le code est très simple et compréhensible. Mais comme mes tes ne passaient pas non plus alors j'ai retiré - 1.0 point.

Note finale : 9/20

3ème place

20simplicite.py

Cet algo n'était pas hors concours et je soupçonne l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient sauf les miens alors je l'ai noté sur une base de 18. Le code pourrait être plus simple et compréhensible donc je retire 3.0 points.

Note finale : 15/20

2ème place

03simplicite.java

Cet algo n'était pas hors concours et je soupçonne l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient sauf les miens alors je l'ai noté sur une base de 18. Le code est très simple et compréhensible, cependant les variables pourrais être mieux nommées donc -1.5 points. Ce qui nous amène à une note de 16.5 que j'ai arrondie à 17.

Note finale : 17/20

1ère place

56simplicite.java

Cet algo n'était pas hors concours et je ne soupçonne pas l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient sauf les miens alors je l'ai noté sur une base de 18. Le code est le plus simple et compréhensible selon moi.

Note finale : 18/20

Tableau catégorie Efficacité

Algos/Places	1ère place	2ème place	3ème place	4ème place	5ème place	Note
56simplicite.java	X	-	-	-	-	18/20
03simplicite.java	-	X	-	-	-	17/20
20simplicite.py	-	-	X	-	-	15/20
64simplicite.java	-	-	-	X	-	9/20
19simplicite.java	-	-	-	-	X	2/20

Classement de la catégorie Sobriété

Pour noter cette catégorie je me suis basé sur l'utilisation mémoire, l'utilisation du CPU et l'efficacité de celui-ci selon ChatGPT.

4ème place

27sobriete.py

Cet algo n'était pas hors concours et je soupçonne l'utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient sauf les miens alors je l'ai noté sur une base de 18. Note de ChatGPT : 16/20 (donc -2 pour les test = 14/20). Ses analyses :

Utilisation de mémoire :

- Utilisation de chaînes de caractères Python natives, qui sont immuables et peuvent être inefficaces pour de nombreuses concaténations. (j'ai retiré 1.0 point)
- Complexité de mémoire proportionnelle à la longueur de la chaîne.

Utilisation du CPU :

- Boucles simples parcourant les caractères de la chaîne.
- Complexité temporelle O(n) pour une seule itération.

Efficacité :

- Code simple et facile à lire.
- L'utilisation de chaînes immuables pourrait entraîner une utilisation excessive de mémoire pour de longues chaînes ou de nombreuses itérations. (j'ai retiré 1.0 point)

Note finale : 12/20

3ème place

03sobriete.java

Cet algo n’était pas hors concours et je soupçonne l’utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient alors je l’ai noté sur une base de 20. Note de ChatGPT : 16/20. Ses analyses :

Utilisation de mémoire :

- Utilisation de `StringBuilder` pour minimiser la création de nouveaux objets chaîne de caractères. (J’ai ajouté 0.5 points)
- Complexité de mémoire proportionnelle à la longueur de la chaîne.
- Pas d'allocations de mémoire explicites en dehors de la chaîne résultat. (J’ai ajouté 0.5 points)

Utilisation du CPU :

- Boucles simples parcourant les caractères de la chaîne.
- Complexité temporelle O(n) pour une seule itération.

Efficacité :

- Code simple et facile à lire.
- Utilisation judicieuse de `StringBuilder` pour optimiser la concaténation de chaînes.

Note finale : 17/20

1ère ex æquo

17sobriete.c

Cet algo n’était pas hors concours et je ne soupçonne pas l’utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient alors je l’ai noté sur une base de 20. Note de ChatGPT : 18/20. Ses analyses :

Utilisation de mémoire :

- Utilisation de `malloc` pour allouer dynamiquement la mémoire, ce qui est efficace. (J’ai ajouté 0.5 points)

- La libération explicite de mémoire avec `free` réduit les risques de fuites de mémoire. (J’ai ajouté 0.5 points)
- Allocation de mémoire proportionnelle à la taille de la chaîne d'entrée.

Utilisation du CPU :

- Boucles simples parcourant les caractères de la chaîne.
- Complexité temporelle O(n) pour une seule itération.

Efficacité :

- Bonne gestion de la mémoire avec `malloc` et `free`.
- Code légèrement verbeux avec des vérifications d'erreurs explicites, mais cela améliore la robustesse.

Note finale : 19/20

39sobriete.c

Cet algo n’était pas hors concours et je ne soupçonne pas l’utilisation de ChatGPT.

Celui-ci compilait bien, était anonyme et fonctionnait. Les test passaient alors je l’ai noté sur une base de 20. Note de ChatGPT : 18/20. Ses analyses :

Utilisation de mémoire :

- Utilisation de `malloc` pour allouer dynamiquement la mémoire. (J’ai ajouté 0.5 points)
- Allocation de mémoire proportionnelle à la taille de la chaîne d'entrée et des résultats. intermédiaires.
- Libération explicite de la mémoire avec `free`. (J’ai ajouté 0.5 points)

Utilisation du CPU :

- Boucles simples parcourant les caractères de la chaîne.
- Complexité temporelle O(n) pour une seule itération.

Efficacité :

- Bonne gestion de la mémoire avec `malloc` et `free`.
- Code légèrement verbeux avec des vérifications d'erreurs explicites, mais cela améliore la robustesse.

Note finale : 19/20

Tableau catégorie Sobriété

Algos/Places	1ère place	2ème place	3ème place	4ème place	Note
39sobriete.c	X	-	-	-	19/20
17sobriete.c	X	-	-	-	19/20
03sobriete.java	-	-	X	-	17/20
27sobriete.py	-	-	-	X	12/20

Conclusion

Cette SAÉ nous a permis d’approfondir notre compréhension et notre maîtrise des approches algorithmiques nécessaires pour résoudre les problèmes rencontrés pendant les phases de développement. En participant à un petit concours de codage, en lisant, comprenant et évaluant des codes qui ne sont pas les nôtres, et en comparant des algorithmes selon des critères précis, nous avons acquis des compétences essentielles pour notre future carrière en informatique.