SAE 2.02 - Exploration Algorithmique

# Phase 2

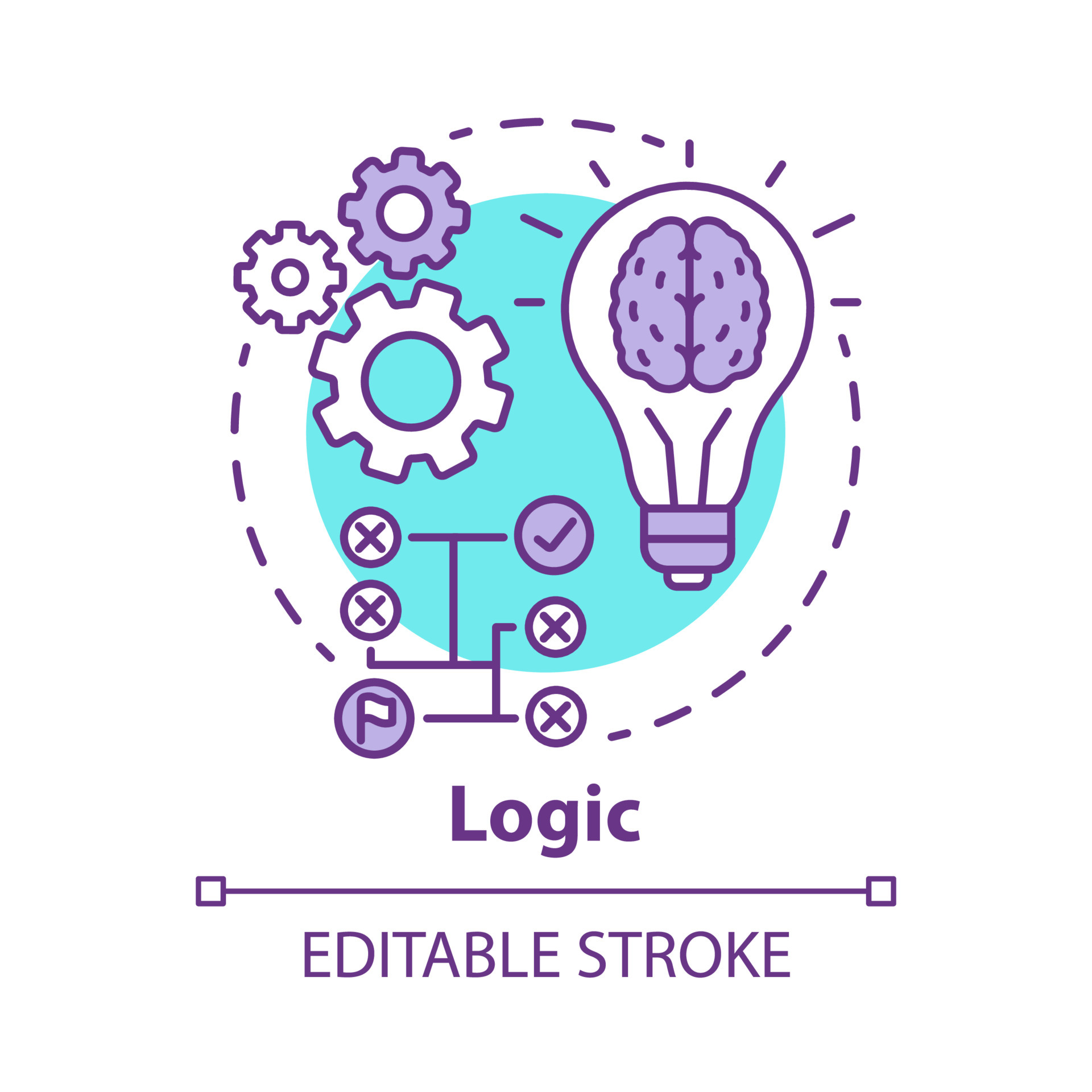


Table des matières

[Introduction 3](#_Toc169888079)

[I/ Evaluation des algorithmes 3](#_Toc169888080)

[II/ Classement des algorithmes 4](#_Toc169888081)

[II – 1 Efficacité 4](#_Toc169888082)

[II – 2 Simplicité 5](#_Toc169888083)

[II – 3 Sobriété 5](#_Toc169888084)

# Introduction

Le jeudi 6 juin 2024, nous a été présenté la SAE 2.2 – Exploration Algorithmique. Celle-ci est composée de deux phase la première consistait à l’écriture de trois algorithmes de compression RLE, un simple, un efficace et un sobre énergétiquement. Cette phase s’est terminée le 16 juin 2024, la phase qui suit consiste à noter puis classer les algorithmes de camarades selon les catégories précédemment citées.

# I/ Evaluation des algorithmes

Pour évaluer les algorithmes il était nécessaire de complémenter les tests donnés initialement de nouveaux tests plus complets. Ainsi vous pouvez trouver les nouveaux tests ainsi que leur version dans d’autres langages dans le dossier « analyse » du projet. Les tests sont contenus dans les fichiers :

* RLECompressionTest.java
* AlgoTest.py

Ainsi grâce à ceux-ci et la grille d’évaluation fournis voici les notes des algorithmes :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Algorithme | Note(?/20) | Commentaire |
| Java | 42Efficacite | 20 | Parfait |
| 43Efficacite | 14 | Passe tous les tests sauf deux algorithmes |
| 44Efficacite | 20 | Parfait |
| 59Efficacite | 18 | Passe les tests initiaux mais pas les miens |
| 60Efficacite | 20 | Parfait |
| 25sobriete | 20 | Parfait |
| 51sobriete | 18 | Passe les tests initiaux mais pas les miens |
| 61sobriete | 20 | Parfait |
| 04Simplicite | 20 | Parfait |
| 21simplicite | 20 | Parfait |
| 50simplicite | 18 | Passe les tests initiaux mais pas les miens |
| 59simplicite | 18 | Passe les tests initiaux mais pas les miens |
| Python | 61simplicite | 14 | Passe tous les tests sauf plusieurs algorithmes |
| 27sobriete | 16 | Passe tous les tests sauf plusieurs algorithme |

# II/ Classement des algorithmes

Viens maintenant le classement des algorithmes par Efficacité, simplicité et sobriété.

## II – 1 Efficacité

Pour classer par efficacité j’ai créé des données de tests dans le fichier DataSet.java grâce à la fonction generateDataFile() dans le fichier RLEBenchmark.java. Ce dernier permet ensuite de comparer la rapidité des différents algorithmes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Efficacité |  |  |  |  |
|  |  |  |  |  |
| Rang | Algo | Nb Algo | Tps Total(ns) | Tps Test |
| 1 | 25sobriete | 4 | 16572105 |  |
| 2 | 04Simplicite | 4 | 16919534 |  |
| 3 | 44Efficacite | 4 | 17352933 |  |
| 4 | 42Efficacite | 4 | 18040105 |  |
| 5 | 61sobriete | 4 | 19598512 |  |
| 6 | 60Efficacite | 4 | 29609983 |  |
| 7 | 21simplicite | 4 | ? | 15 |
| 8 | 43Efficacite | 3 | 31471472 |  |
| 9 | 27sobriete.py | 3 | ? | 60 |
| 10 | 51sobriete | 2 | 17363325 |  |
| 11 | 59simplicite | 2 | 18800519 |  |
| 12 | 59Efficacite | 2 | 20173017 |  |
| 13 | 61simplicite.py | 2 | ? | 20 |
| 14 | 50simplicite | 2 | ? | 26 |

Ainsi le Tps Total est le temps de tests avec mes données additionnelles.

Ici les algorithmes sont triés d’abords par nombre d’algorithmes réellement fonctionnel. Puis par la vitesse à remplir mes tests et enfin pour les algorithmes trop lents par le temps de completion des tests initiaux.

## II – 2 Simplicité

Pour trier les algorithmes par simplicité j’ai décidé de prendre en compte la lisibilité perçue, la qualité donnée par Codacy, les commentaires et la javaDoc ainsi je suis arrivé à ce classement de simplicité :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Simplicité |  |  |  |  |  |
|  |  |  |  |  |  |
| Rang | Algo | Qualité | Commentaires | JavaDoc | Additionnel |
| 1 | 61Simplicite.py | A | Généreux | Moitié des Fichiers |  |
| 2 | 21simplicite.java | A | Généreux | Absente |  |
| 3 | 59simplicite.java | B | Généreux | Partout |  |
| 3 | 59efficacite.java | B | Généreux | Partout |  |
| 5 | 43efficacite.java | A | Plus que correct | Partout |  |
| 6 | 61sobriete.java | A | 1 quart | Moitié des Fichiers |  |
| 7 | 44efficacite.java | A | Fragiles | Fragiles | Format de javadoc incorrect! |
| 8 | 04simplicite.java | A | 1 commentaire | Absente |  |
| 8 | 50Simplicite.java | A | 0 commentaire | Absente |  |
| 10 | 25sobriete.java | A | 0 commentaire | Absente | Nom de variable corrects |
| 11 | 51sobriete.java | A | 0 commentaire | Absente | Nom de variable corrects |
| 12 | 42efficacite | A | 0 commentaire | Absente | Nom de variable corrects |
| 13 | 27sobriete.py | A | 0 commentaire | Absente | Nom de variable fragiles |
| 13 | 60efficacite.java | A | 0 commentaire | Absente | Nom de variable fragiles |

## II – 3 Sobriété

N’ayant réussi à faire fonctionner l’outils fourni par manque de droits sur les pcs de l’iut je les ai classés ex-aequo.