BREFUEL 4B
Mathis

Rapport d’évaluation des Algorithmes

Table des matières

Grille d’évaluation des algorithmes

Simplicité-Meilleur

Simplicité meilleur 26

Problème	Sanction		Simplicité meilleur 26
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**10**

Simplicité meilleur 32

Problème	Sanction		Simplicité meilleur 32
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 1 test
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**8,5**

Simplicité meilleur 53

Problème	Sanction		Simplicité meilleur 53
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 2 tests
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour erronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**7**

Simplicité-Pire

Simplicité pire 16

Problème	Sanction		Simplicité pire 16
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**2,5 HC**

Simplicité pire 61

Problème	Sanction		Simplicité pire 61
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**2,5**

Efficacité-Meilleur

Efficacité meilleur 15

Problème	Sanction		Efficacité meilleur 15
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**20**

Efficacité meilleur 44

Problème	Sanction		Efficacité meilleur 44
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 1 test
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**18**

Efficacité-Pire

Efficacité pire 43

Problème	Sanction		Efficacité pire 43
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 1 test
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**9**

Efficacité pire 57

Problème	Sanction		Efficacité pire 57
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 3 tests
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**13 HC**

Sobriété-Meilleur

Sobriété meilleur 22

Problème	Sanction		Sobriété meilleur 22
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 2 tests
Passe vos tests supplémentaires plus complets	20		Passe pas 1 test
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**6,5**

Sobriété meilleur 68

Problème	Sanction		Sobriété meilleur 68
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**1,5 HC**

Sobriété-Pire

Sobriété pire 17

Problème	Sanction		Sobriété pire 17
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		Passe pas 1 test
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**18**

Sobriété pire 57

Problème	Sanction		Sobriété pire 57
Ne compile pas	note finale divisée par 2	Et hors concours	
Ne correspond pas à une classe Exercice (cas des main ou des zip)	note finale divisée par 2		
Non respect de la nomenclature précise	-1		
Non respect de l'anonymat	-1		
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours	
Passe tous les tests fournis initialement	18		
Passe vos tests supplémentaires plus complets	20		
Fonctionne mais ne passe pas les tests fournis initialement	10		
Ne fonctionne pas (retour eronné, ou pas du bon type attendu)	5		
TOTAL (/20)			**0 HC**

Lisibilité du code

Simplicité-Meilleur

Simplicité meilleur 26

- cet algorithme est pourvu de commentaires placés judicieusement juste avant chaque boucle afin d'expliquer leur rôle.
- les indentations et les sauts de ligne sont effectuer aux bons endroits ce qui rend le code très lisible mais aussi très facile à comprendre.
- les noms des variables sont descriptifs et significatifs.

Cet algorithme est donc parfaitement lisible.

Simplicité meilleur 32

- l'algorithme utilise plusieurs fonctions chacune ayant une java doc complète.
- les indentations sont bonnes, il manque peut être un peu de sauts de ligne pour aérer le code.
- les noms des variables sont descriptifs et significatifs.
- les commentaires ne sont pas nécessaire grâce à la java doc très détaillée.

C'est un algorithme parfaitement lisible.

Simplicité meilleur 53

- l'algorithme est dépourvu de commentaire et de java doc
- les noms des variables sont ok
- l'algorithme reste court et aérer

L'algorithme n'est pas très lisible.

Simplicité-Pire

Simplicité pire 16

- le code est écrit sur une seule ligne ce qui rend la lecture impossible
- en revanche les noms des variables sont explicites

L'algorithme reste cependant illisible.

Simplicité pire 61

- cet algorithme est pourvu de commentaires placés judicieusement juste avant chaque boucle afin d'expliquer leur rôle.
- les indentations et les sauts de ligne sont effectuer aux bons endroits ce qui rend le code très lisible mais aussi très facile à comprendre.
- les noms des variables sont descriptifs et significatifs.

Cet algorithme est donc parfaitement lisible.

Efficacité-Meilleur

Efficacité meilleur 15

- le code est mal indenté mais la séparation des blocs est visible
- aucun commentaire ou java doc n'est fournis
- les noms des variables sont explicites

L'algorithme est peu lisible.

Efficacité meilleur 44

- l'algorithme est pourvu de commentaires placés judicieusement juste avant chaque boucle afin d'expliquer leur rôle.
- les boucles sont correctement indentées
- les noms des variables sont descriptifs et significatifs.

L'algorithme est très lisible.

Efficacité-Pire

Efficacité pire 43

- le code est correctement indenté et la séparation des blocs est visible
- aucun commentaire ou java doc n'est fournis
- les noms des variables sont explicites

L'algorithme est lisible.

Efficacité pire 57

- l'algorithme est dépourvu de commentaire et de java doc
- les noms des variables sont significatifs

- les indentations sont bonnes et les sauts de lignes rendent l'algorithme plus lisible
- il y a beaucoup trop d'import inutile

L'algorithme n'est pas très lisible.

Sobriété-Meilleur

Sobriété meilleur 22

- l'algorithme est pourvu de commentaires placés judicieusement juste avant chaque boucle afin d'expliquer leur rôle.
- les noms des variables sont descriptifs et significatifs.

L'algorithme est très lisible.

Sobriété meilleur 68

- l'algorithme est pourvu de commentaires placés judicieusement juste avant chaque boucle afin d'expliquer leur rôle.
- les boucles sont correctement indentées
- les noms des variables sont descriptifs et significatifs.

L'algorithme est très lisible.

Sobriété-Pire

Sobriété pire 17

- l'algorithme utilise plusieurs fonctions chacune ayant une java doc complète.
- cet algorithme est pourvu de commentaires placés judicieusement juste avant chaque boucle afin d'expliquer leur rôle.
- les indentations et les sauts de ligne sont effectuer aux bons endroits ce qui rend le code très lisible mais aussi très facile à comprendre.
- les noms des variables sont descriptifs et significatifs.

Cet algorithme est donc parfaitement lisible.

Sobriété pire 57

- l'algorithme est dépourvu de commentaire et de java doc
- les noms des variables sont significatifs
- les indentations sont bonnes et les sauts de lignes rendent l'algorithme plus lisible

L'algorithme est lisible.

Qualité du code

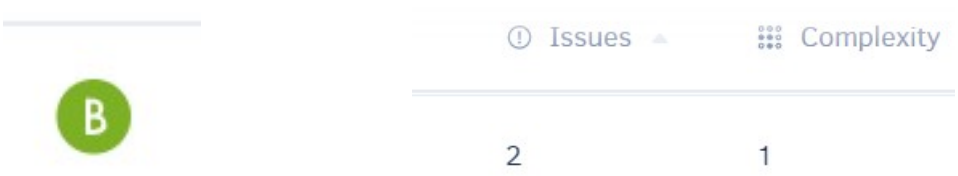
Pour tester la qualité des codes, nous allons utiliser l'outil « ***Codacy*** ».

Simplicité-Meilleur

Simplicité meilleur 26

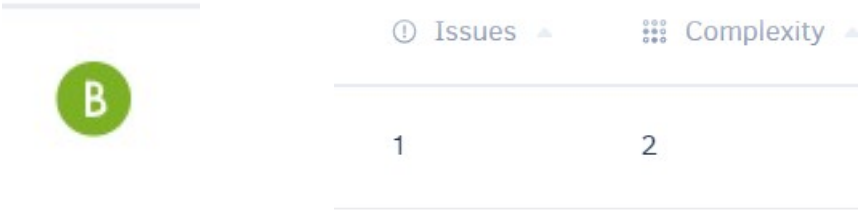
Simplicité meilleur 32

Simplicité meilleur 53



Simplicité-Pire

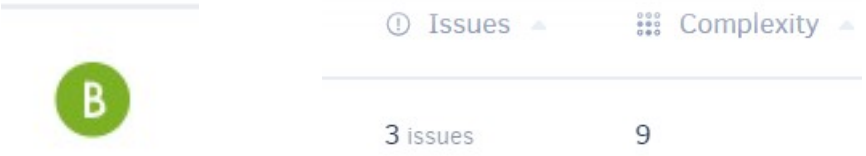
Simplicité pire 16



Simplicité pire 61

Efficacité-Meilleur

Efficacité meilleur 15

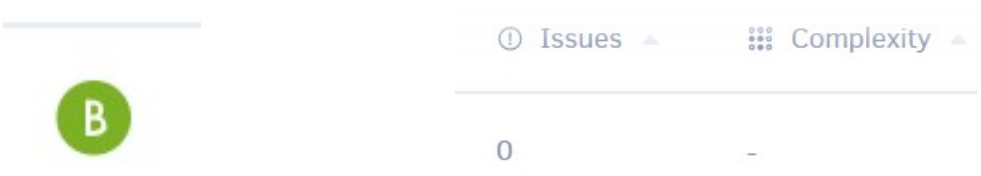


Efficacité meilleur 44

Efficacité-Pire

Efficacité pire 43

Efficacité pire 57



Sobriété-Meilleur

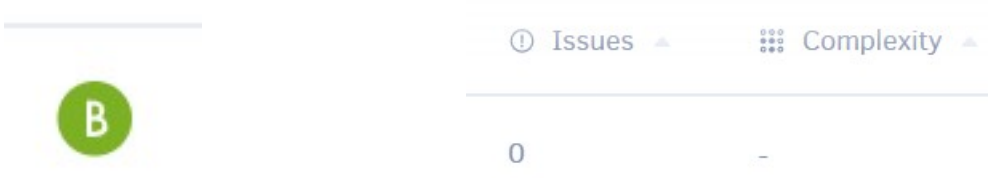
Sobriété meilleur 22

Sobriété meilleur 68

Sobriété-Pire

Sobriété pire 17

Sobriété pire 57



Efficacité

Pour tester l'efficacité des codes, nous allons rajouter dans la classe main(), celle qui lance les différents algorithmes, les lignes suivantes :

```
double startTime = System.currentTimeMillis();
(...)
double endTime = System.currentTimeMillis();
System.out.println((endTime-startTime)/1000+"s");
```

Ces lignes nous permettent d'obtenir le temps de calcul d'un algorithme, on pourra donc ajouter un gros texte et voir si le temps augmente de beaucoup ou non.

Nous utiliserons plusieurs taille de texte : celui de base à 5 mots, un texte à 100 mots et un à 1000 mots.

5 mots : "exemple de texte a classer"

100 mots : "Lete etait enfin arrive apportant avec lui le doux parfum des fleurs et le chant melodieux des oiseaux Les rayons du soleil caressaient delicatement la peau rechauffant les coeurs et chassant les soucis Les journees se remplissaient daventures et de rires tandis que les soirees invitaient a la detente et a la contemplation Les vacances tant attendues permettaient de sevader du quotidien et de decouvrir de nouveaux horizons Les plages accueillaient les baigneurs avides de fraicheur tandis que les montagnes offraient des panoramas a couper le souffle Cetait le moment parfait pour se ressourcer et creer des souvenirs inoubliables"

1000 mots : "Le soleil se levait lentement a l'horizon baignant le paysage d'une douce lueur doree Les oiseaux commencaient a chanter leur symphonie matinale annoncant ainsi le debut d'une nouvelle journee Dans cette petite ville tranquille la vie s'eveillait peu a peu Les rues se remplissaient de passants presses se rendant a leurs occupations quotidiennes Les commerçants ouvraient leurs boutiques et disposaient leurs produits avec soin Les étales se remplissaient de couleurs et d'odeurs alléchantes Les clients déambulaient dans les rues s'arrêtant çà et là pour admirer les étalages et effectuer leurs achats Au parc de nombreux enfants s'amusaient dans l'aire de jeux Les rires et les cris joyeux résonnaient dans l'air Les parents les observaient avec tendresse tout en échangeant des sourires complices Les étudiants se pressaient vers l'université pour assister à leurs cours Les salles de classe se remplissaient d'une foule studieuse prête à apprendre Les professeurs partageaient leur savoir avec passion et les étudiants prenaient des notes attentivement Dans les cafés et les restaurants les gens prenaient leur petit-déjeuner ou dégustaient tranquillement leur repas Des conversations animées se mêlaient aux arômes du café fraîchement moulu et des plats savoureux Certains se rendaient au travail en bus en train ou à vélo Les rues se remplissaient de voitures qui avançaient lentement dans le flot de la circulation Les klaxons retentissaient de temps en temps rompant le calme matinal Les parcs et les jardins étaient des havres de paix où les gens venaient se détendre et profiter de la nature Certains faisaient leur jogging matinal d'autres se promenaient en respirant l'air frais et en admirant les fleurs colorées qui s'épanouissaient À la bibliothèque les étudiants et les amateurs de lecture se plongeaient dans les pages de livres et de revues Le silence régnait dans cet espace dédié à la connaissance où chacun pouvait s'évader à travers les mots et les histoires Le soir venu les restaurants et les bars s'animaient de conversations animées et de rires joyeux Les amis se retrouvaient pour partager un repas ou prendre un verre ensemble dans une ambiance conviviale et chaleureuse Les étoiles scintillaient dans le ciel nocturne offrant un spectacle enchanteur Les couples se promenaient main dans la main profitant de la quiétude de la nuit Les lumières des lampadaires donnaient une ambiance romantique à la ville endormie Puis la nuit s'installait peu à peu enveloppant la ville d'un voile sombre Les rues se vidaient et le calme revenait Les habitants s'endormaient paisiblement se préparant ainsi à une nouvelle journée qui les attendait au réveil"

Algorithmes	X5	X100	X1000
Simplicité meilleur 26	0,002	0,002	0,005
Simplicité meilleur 32	0,013	0,015	0,018
Simplicité pire 16 (ne compile pas)			
Efficacité meilleur 15	0,002	0,004	0,006
Efficacité pire 43	0,001	0,006	0,024
Sobriété meilleur 22	0,02	0,018	0,029
Sobriété pire 17	0,014	0,017	0,028
Efficacité meilleur 15	0,001	0,001	0,001
Simplicité meilleur 53	0,001	0,001	0,001
Efficacité pire 57	2,255	54,496	231,232
Sobriété pire 57 (ne compile pas)			
Simplicité pire 61	0,012	0,021	0,033
Sobriété meilleur 68 (ne compile pas)			

* le deuxième efficacité meilleur 15 est en fait efficacité meilleur 44

Les temps obtenus sont en secondes.

Sobriété numérique

Afin de mesurer la consommation en ressources d'un algorithme, nous allons utiliser l'outil « ***JoularJX*** ».

Simplicité-Meilleur

Simplicité meilleur 26

Simplicité meilleur 32

Simplicité-Pire

Simplicité pire 16

Efficacité-Meilleur

Efficacité meilleur 15

Efficacité-Pire

Efficacité pire 43

Sobriété-Meilleur

Sobriété meilleur 22

Sobriété-Pire

Sobriété pire 17

Temps d’exécution

Pour tester le temps d’exécution des codes, nous allons rajouter dans la classe main(), les lignes suivantes :

```
double startTime = System.currentTimeMillis();
(...)
double endTime = System.currentTimeMillis();
System.out.println((endTime-startTime)/1000+"s");
```

Algorithmes	X5
Simplicité meilleur 26	0,002
Simplicité meilleur 32	0,013
Simplicité pire 16 (ne compile pas)	
Efficacité meilleur 15	0,002
Efficacité pire 43	0,001
Sobriété meilleur 22	0,02
Sobriété pire 17	0,014
Efficacité meilleur 15	0,001
Simplicité meilleur 53	0,001
Efficacité pire 57	2,255
Sobriété pire 57 (ne compile pas)	
Simplicité pire 61	0,012
Sobriété meilleur 68 (ne compile pas)	

Classement par catégorie

Simplicité-Meilleur

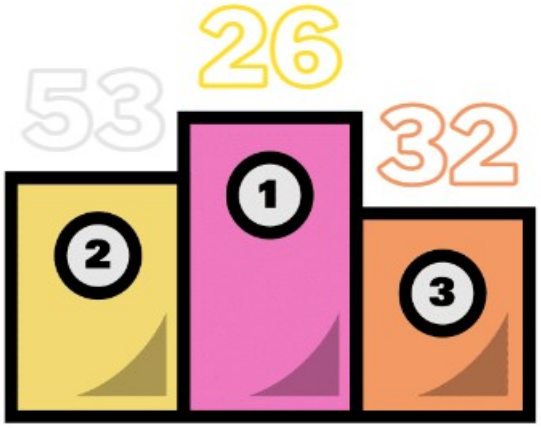
Pour pouvoir affirmer qu'un algorithme est simple, il faut selon moi qu'il remplisse plusieurs cases :

- premièrement, l'algorithme doit être lisible.

- il doit être court

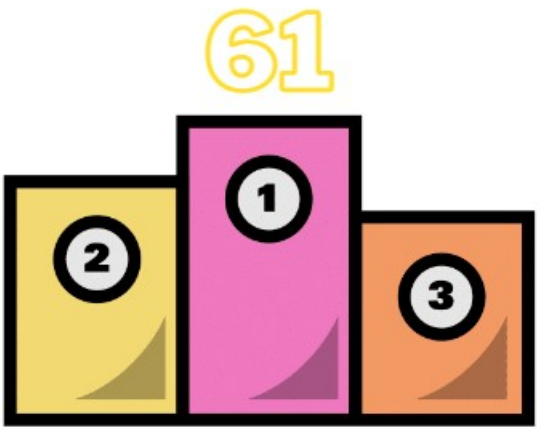
- et il doit être compréhensible

De ces critères et des analyses faites précédemment je peux en déduire le classement suivant :



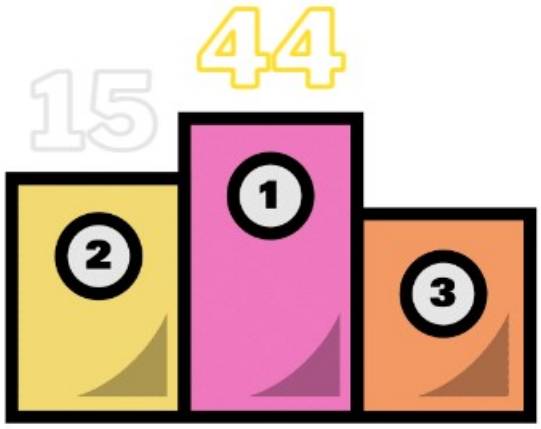
Simplicité-Pire

Le pire algorithme simple doit être exactement l'inverse.

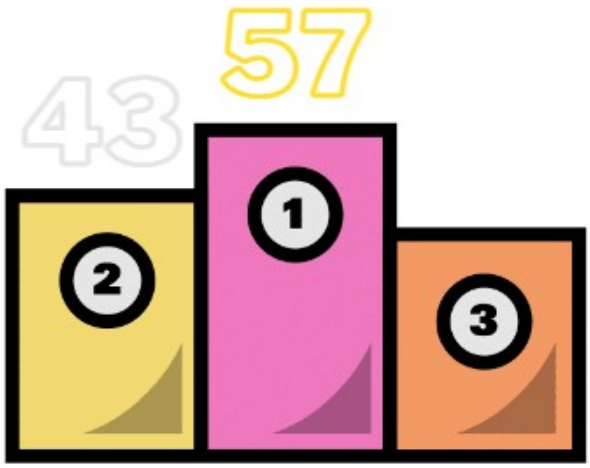


Efficacité-Meilleur

Voir le tableau efficacité.



Efficacité-Pire



Sobriété-Meilleur

Nous n’avons pas trouvé de moyen d’évaluer la sobriété.

Sobriété-Pire

Nous n’avons pas trouvé de moyen d’évaluer la sobriété.

Hors Concours

Simplicité pire 16/ Sobriété pire 57/ Sobriété meilleur 68