
Algorithmique 3 : structures de données arborescentes

Projet

Distance de Jaccard

Généralités

La *distance de Jaccard*¹ qui sépare deux ensembles A et B est notée $J_\delta(A, B)$. Elle est définie par :

$$J_\delta(A, B) = \begin{cases} 0, & \text{si } A = \emptyset \wedge B = \emptyset; \\ 1 - \frac{|A \cap B|}{|A \cup B|}, & \text{sinon.} \end{cases}$$

Elle est à valeurs dans l'intervalle réel $[0; 1]$. Elle mesure la *dissimilarité* entre les ensembles : plus sa valeur est proche de 0, plus les ensembles sont semblables ; plus sa valeur s'éloigne de 0, plus les ensembles sont dissemblables.

Considérons les deux ensembles $A = \{a, b, c, d, e\}$ et $B = \{b, c, f\}$. Nous avons alors $A \cap B = \{b, c\}$ et $A \cup B = \{a, b, c, d, e, f\}$, et donc $|A \cap B| = 2$ et $|A \cup B| = 6$. Nous obtenons donc $J_\delta(A, B) = 1 - 2/6 = 1 - 1/3 = 2/3$.

Le projet consiste à écrire un programme en C dont le but est d'afficher la distance de Jaccard qui sépare les ensembles des mots qui figurent dans toute paire de fichiers textes signifiés sur la ligne de commande.

Un mot est, par défaut, une séquence de longueur maximale de caractères n'appartenant pas à la classe `isspace`.

Pour toute paire de fichiers textes, l'affichage est de la forme leur distance de Jaccard exprimée avec quatre décimales, le caractère de tabulation horizontale `'\t'`, l'argument associé au premier fichier texte (ou deux caractères `'\"'` consécutifs s'il s'agit de l'entrée standard), le caractère de tabulation horizontale, l'argument associé au deuxième fichier texte (même convention), le caractère de fin de ligne `'\n'`.

L'exécutable doit être nommé `jdis` (pour *Jaccard dissimilarity*).

Exemples basiques

Trois fichiers textes sont tout d'abord créés, à savoir `x0.txt`, `x1.txt` et `x2.txt`. Quatre exécutions sont ensuite lancées, les trois premières sur deux fichiers, la dernière sur les trois.

```
$ cat > x0.txt
a ee iii oooo uuuuu yyyyyy
$ cat > x1.txt
a zz eee rrrr ttttt yyyyyy
$ cat > x2.txt
a bb ccc dddd eeeee fffffff ggggg hhhh iii jj k
a bb ccc dddd eeeee hhhh iii jj k
a bb ccc hhhh iii jj k
a bb ccc jj k
```

MÀJ 22-03
Simplification : au moins une paire de noms de fichiers ou assimilés doivent figurer sur la ligne de commande.

1. Paul Jaccard est un professeur suisse de botanique, spécialiste de physiologie végétale, à l'origine de deux mesures utilisées en statistique : l'« indice (de Jaccard) » (ou le « coefficient de similarité », et, originellement, le « coefficient de communauté », défini comme le quotient des cardinaux de l'intersection et de l'union) et la distance dont il est ici question.

```

a jj k
a
$ ./jdis x0.txt x1.txt
0.8000→x0.txt→x1.txt
$ ./jdis x1.txt x2.txt
0.9375→x1.txt→x2.txt
$ ./jdis x2.txt x0.txt
0.8667→x2.txt→x0.txt
$ ./jdis x0.txt x1.txt x2.txt
0.8000→x0.txt→x1.txt
0.8667→x0.txt→x2.txt
0.9375→x1.txt→x2.txt

```

Les ensembles associés à `x0.txt` et `x1.txt` ont en commun les 2 mots `a` et `yyyyyy`; ils possèdent à eux deux les 6 mots de `x0.txt` plus les 4 mots que possède seul `x1.txt`, à savoir `zz`, `eee`, `rrrr` et `ttttt`. Leur distance de Jaccard vaut donc $1 - 2/(6+4) = 1 - 2/10 = 1 - 1/5 = 4/5 = 0,8$. C'est ce qu'affichent les première et quatrième exécutions de `jdis`.

Tous les mots différents de `x2.txt` figurent sur sa première ligne; 11 au total. Les 2 mots qu'ont en commun `x0.txt` et `x2.txt` sont `a` et `iii`. Les 4 autres mots de `x0.txt` n'apparaissent pas dans `x2.txt`. Leur distance de Jaccard vaut donc $1 - 2/(4 + 11) = 13/15 \simeq 0,866\ 7$. C'est ce qu'affichent les deux dernières exécutions.

Le seul mot qu'ont en commun `x1.txt` et `x2.txt` est `a`. Les 5 autres mots de `x0.txt` n'apparaissent pas dans `x2.txt`. Leur distance de Jaccard vaut donc $1 - 1/(5 + 11) = 15/16 = 0,937\ 5$. C'est ce qu'affichent les deuxième et quatrième exécutions.

Exemples avec options

Des options peuvent être ajoutées au programme, fournies sur la ligne de commande selon les canons usuels, qui modifient les productions des exécutions.

La longueur maximale prise en compte pour les mots peut être fixée. Au delà de cette longueur, tout mot est coupé. La longueur est fixée à 2 dans l'exemple ci-dessous, les avertissements signifiant la coupure de mots étant redirigés vers le trou noir :

```

$ ./jdis -i2 x0.txt x1.txt x2.txt 2>/dev/null
0.6667→x0.txt→x1.txt
0.7857→x0.txt→x2.txt
0.8667→x1.txt→x2.txt

```

Les fichiers `x0.txt` et `x1.txt` partagent un mot de plus, `ee`; d'où la distance $1 - 3/(6 + 3) = 2/3 \simeq 0,666\ 7$. Même chose pour `x0.txt` et `x2.txt`; d'où la distance $1 - 3/(3 + 11) = 11/14 \simeq 0,785\ 7$. Même chose encore pour `x1.txt` et `x2.txt`; d'où la distance $1 - 2/(4 + 11) = 13/15 \simeq 0,866\ 7$.

Une option permet de modifier les productions : au lieu d'afficher les distances, c'est le graphe d'appartenance de chaque mot appartenant à la réunion des ensembles des textes qui est affiché.

```

$ ./jdis -g -i2 x0.txt x1.txt x2.txt 2>/dev/null
----->x0.txt->x1.txt->x2.txt
a----->x----->x----->x
bb----->------>------>x
cc----->------>------>x
dd----->------>------>x
ee----->x----->x----->x

```

```

ff----->------>------>x
gg----->------>------>x
hh----->------>------>x
ii----->x----->------>x
jj----->------>------>x
kk----->------>------>x
oo----->x----->------>-
rr----->------>x----->-
tt----->------>x----->-
uu----->x----->------>-
yy----->x----->x----->-
zz----->------>x----->-

```

Selon que le mot figurant dans la colonne de gauche apparaît ou non dans le fichier dont le nom est cité sur la première ligne dans les trois autres colonnes, un « x » ou un « - » est affiché. La tabulation sert de séparateur de colonnes. Les mots sont affichés dans l'ordre lexicographique engendré par la fonction standard `strcoll`². On retrouve très facilement sur ce graphe les résultats annoncés plus haut. Pour les fichiers `x0.txt` et `x1.txt` par exemple : 3 mots en commun, 9 mots au total, soit $1 - 3/9 = 2/3$.

Sur des textes plus « réels » maintenant, en ce sens où y figurent des mots de langue française et des symboles de ponctuation :

```

$ cat > toto0.txt
La maitresse de Toto lui demande :
- Toto, conjugue-moi le verbe savoir a tous les temps.
- Je sais qu'il pleut, je sais qu'il fait beau, je sais qu'il neige.
$ cat > toto1.txt
La maitresse demande a Toto de conjuguer le verbe manger a la premiere personne
du present, du futur et du passe compose.
Toto dit :
- Euh... je mange, je mangerai... euh... j'ai plus faim !
$ cat > toto2.txt
La maitresse demande a Toto de conjuguer le verbe marcher au present et a toutes
les personnes.
Toto lui repond :
- Je marche... tu marches... il marche... nous...
La maitresse l'interrompt et lui demande d'aller plus vite.
Alors Toto accelere :
- Je cours, tu cours, il court, nous courons, vous courez, ils courent !
$ cat > toto3.txt
La maitresse demande a Toto :
- Toto, pourrais-tu me dire ce qu'est un oiseau migrateur ?
- Oui Madame, je peux : c'est un oiseau qui ne se gratte que d'un seul cote.
$ ./jdis -g toto0.txt toto1.txt toto2.txt toto3.txt
----->toto0.txt----->toto1.txt----->toto2.txt----->toto3.txt
!----->------>x----->x----->-
------>x----->x----->x----->x

```

MÀJ 22-03
`strcoll` est imposée.
L'option `-l` est supprimée.
Les deux exemples qui suivent s'en trouvent modifiés en conséquence.

2. Après avoir pris soin de faire appel, en début d'exécution, à la fonction `setlocale` de l'en-tête standard `<locale.h>` avec comme premier paramètre la macroconstante `LC_COLLATE` définie dans le même en-tête

```

:—>x—>x—>x—>x
?—>-—>-—>-—>x
a—>x—>x—>x—>x
accelere—>-—>-—>x—>-
Alors—>-—>-—>x—>-
au—>-—>-—>x—>-
beau,—>x—>-—>-—>-
ce—>-—>-—>-—>x
c'est—>-—>-—>-—>x
compose.—>-—>x—>-—>-
conjugue-moi—>x—>-—>-—>-
conjuguer—>-—>x—>x—>-
cote.—>-—>-—>-—>x
courent—>-—>-—>x—>-
courez,—>-—>-—>x—>-
courons,—>-—>-—>x—>-
cours,—>-—>-—>x—>-
court,—>-—>-—>x—>-
d'aller—>-—>-—>x—>-
de—>x—>x—>x—>-
demande—>x—>x—>x—>x
dire—>-—>-—>-—>x
dit—>-—>x—>-—>-
du—>-—>x—>-—>-
d'un—>-—>-—>-—>x
et—>-—>x—>x—>-
euh...—>-—>x—>-—>-
Euh...—>-—>x—>-—>-
faim—>-—>x—>-—>-
fait—>x—>-—>-—>-
futur—>-—>x—>-—>-
gratte—>-—>-—>-—>x
il—>-—>-—>x—>-
ils—>-—>-—>x—>-
j'ai—>-—>x—>-—>-
je—>x—>x—>-—>x
Je—>x—>-—>x—>-
la—>-—>x—>-—>-
La—>x—>x—>x—>x
le—>x—>x—>x—>-
les—>x—>-—>x—>-
l'interrompt—>-—>-—>x—>-
lui—>x—>-—>x—>-
Madame,—>-—>-—>-—>x
maitresse—>x—>x—>x—>x
mange,—>-—>x—>-—>-
manger—>-—>x—>-—>-
mangerai...—>-—>x—>-—>-
marche...—>-—>-—>x—>-

```

```

marcher>->->x>-
marches...>->->x>-
me>->->->x
migrateur>->->->x
ne>->->->x
neige.>x>->->-
nous>->->x>-
nous...>->->x>-
oiseau>->->->x
Oui>->->->x
passe>->x>->-
personne>->x>->-
personnes.>->->x>-
peux>->->->x
pleut,>x>->->-
plus>->x>x>-
pourrais-tu>->->->x
premiere>->x>->-
present>->->x>-
present,>->x>->-
que>->->->x
qu'est>->->->x
qui>->->->x
qu'il>x>->->-
repond>->->x>-
sais>x>->->-
savoir>x>->->-
se>->->->x
seul>->->->x
temps.>x>->->-
Toto>x>x>x>x
Toto,>x>->->x
tous>x>->->-
toutes>->->x>-
tu>->->x>-
un>->->->x
verbe>x>x>x>-
vite.>->->x>-
vous>->->x>-
$ ./jdis toto0.txt toto1.txt toto2.txt toto3.txt
0,7556>toto0.txt>toto1.txt
0,7547>toto0.txt>toto2.txt
0,8043>toto0.txt>toto3.txt
0,7586>toto1.txt>toto2.txt
0,8491>toto1.txt>toto3.txt
0,8906>toto2.txt>toto3.txt

```

MÀJ 08-04

En début
d'extrait
des deux
exemples,
ajouts de
l'obtention
des textes.

La vérification « à l'œil et à la main » afin de retrouver les résultats à partir du graphe est plus fastidieuse. Mais le graphe, au format CSV, peut par exemple être édité dans un tableur ; des

colonnes « B » et « C », il vient 11 mots en commun, « x et x », pour 45 au total, « x ou x », soit $1 - 11/45 = 34/45 \simeq 0,755$ 6.

Des résultats sensiblement plus cohérents peuvent toutefois être obtenus en tentant de se rapprocher de la notion de « mot » en supprimant les caractères de ponctuation de la classe `ispunct` : (Voir le

MÀJ 10-04
postscriptum de la
question 7.)
(Sébastien)

```
$ ./jdis -g -p toto0.txt toto1.txt toto2.txt toto3.txt
----->/toto0.txt----->/toto1.txt----->/toto2.txt----->/toto3.txt
a----->x----->x----->x----->x
accelere----->------>------>x----->-
ai----->------>x----->------>-
aller----->------>------>x----->-
Alors----->------>------>x----->-
au----->------>------>x----->-
beau----->x----->------>------>-
c----->------>------>------>x
ce----->------>------>------>x
compose----->------>x----->------>-
conjugue----->x----->------>------>-
conjuguer----->------>x----->x----->-
cote----->------>------>------>x
courent----->------>------>x----->-
courez----->------>------>x----->-
courons----->------>------>x----->-
cours----->------>------>x----->-
court----->------>------>x----->-
d----->------>------>x----->x
de----->x----->x----->x----->-
demande----->x----->x----->x----->x
dire----->------>------>------>x
dit----->------>x----->------>-
du----->------>x----->------>-
est----->------>------>------>x
et----->------>x----->x----->-
euh----->------>x----->------>-
Euh----->------>x----->------>-
faim----->------>x----->------>-
fait----->x----->------>------>-
futur----->------>x----->------>-
gratte----->------>------>------>x
il----->x----->------>x----->-
ils----->------>------>x----->-
interrompt----->------>------>x----->-
j----->------>x----->------>-
je----->x----->x----->------>x
Je----->x----->------>x----->-
l----->------>------>x----->-
la----->------>x----->------>-
La----->x----->x----->x----->x
le----->x----->x----->x----->-
```

MÀJ 08-04
Dans le
deuxième
exemple,
suppression
de l'incon-
gru -g.
(Jugurta)

```

les      }x      }-      }x      }-
lui      }x      }-      }x      }-
Madame  }-      }-      }-      }x
maitresse      }x      }x      }x      }x
mange    }-      }x      }-      }-
manger   }-      }x      }-      }-
mangerai  }-      }x      }-      }-
marche   }-      }-      }x      }-
marcher  }-      }-      }x      }-
marches  }-      }-      }x      }-
me       }-      }-      }-      }x
migrateur      }-      }-      }-      }x
moi      }x      }-      }-      }-
ne       }-      }-      }-      }x
neige    }x      }-      }-      }-
nous     }-      }-      }x      }-
oiseau   }-      }-      }-      }x
Oui      }-      }-      }-      }x
passe    }-      }x      }-      }-
personne }-      }x      }-      }-
personnes      }-      }-      }x      }-
peux     }-      }-      }-      }x
pleut    }x      }-      }-      }-
plus     }-      }x      }x      }-
pourrais }-      }-      }-      }x
premiere }-      }x      }-      }-
present  }-      }x      }x      }-
qu       }x      }-      }-      }x
que      }-      }-      }-      }x
qui      }-      }-      }-      }x
repond   }-      }-      }x      }-
sais     }x      }-      }-      }-
savoir   }x      }-      }-      }-
se       }-      }-      }-      }x
seul     }-      }-      }-      }x
temps    }x      }-      }-      }-
Toto     }x      }x      }x      }x
tous     }x      }-      }-      }-
toutes   }-      }-      }x      }-
tu       }-      }-      }x      }x
un       }-      }-      }-      }x
verbe    }x      }x      }x      }-
vite     }-      }-      }x      }-
vous     }-      }-      }x      }-
$ ./jdis -p toto0.txt toto1.txt toto2.txt toto3.txt
0,7955 }/toto0.txt }/toto1.txt
0,7647 }/toto0.txt }/toto2.txt
0,8444 }/toto0.txt }/toto3.txt
0,7857 }/toto1.txt }/toto2.txt

```

```
0,8824→/toto1.txt→/toto3.txt
0,8833→/toto2.txt→/toto3.txt
```

La distance entre deux histoires de Toto est bien sûr égale à la tête à Toto s'il s'agit de la même histoire :

```
$ ./jdis toto0.txt ../../toto0.txt
0.0000→toto0.txt→../../toto0.txt
```

Et elle est très grande, très proche de 1 donc, entre une histoire de Toto et une production³ du Grand Homme (où « ... » signifie le chemin donnant accès au fichier en question) :

```
$ ./jdis toto0.txt ../lesmiserables.txt
0.9997→toto0.txt→../lesmiserables.txt
$ ./jdis -p toto0.txt ../lesmiserables.txt
0.9993→toto0.txt→../lesmiserables.txt
```

MÀJ 08-04
Sans puis
avec -p,
et avec
les bonnes
distances.

Mise en œuvre

Votre programme doit être écrit en C, pouvoir être compilé avec gcc branche 13.1 (minimum) et supporter les options de compilation usuelles : -std=c2x, -Wall, -Wconversion, -Werror, -Wextra, -Wpedantic, -Wwrite-strings et -O2.

MÀJ 22-03
Précision de
la branche.

Vos sources C doivent être mis en forme par uncrustify selon les fichiers de configuration fournis. Aucune ligne ne doit avoir une longueur strictement supérieure à 80.

Les types, sous-programmes et paramètres doivent être correctement nommés. Toute duplication de code est à éviter. En particulier, les éventuels caractères, chaînes et nombres magiques seront nommés.

Vous éviterez l'utilisation de variables globales.

L'exécutable doit être nommé jdis.

À la base de votre travail, vous utiliserez nécessairement soit l'en-tête "hashtable.h", soit l'en-tête "bst.h", sans les modifier. La partie implantation associée à l'en-tête "bst.h" se doit de traiter les arbres binaires de recherche comme des AVL.

Vous pouvez développer et utiliser des modules additionnels, par exemple pour gérer les options ou la partie récupération de mots dans un flot texte.

Toutes les zones mémoires explicitement allouées doivent être désallouées avant la fin de l'exécution, même en cas d'erreur. Jamais deux chaînes de caractères de même valeur ne doivent être allouées.

Tout message d'avertissement ou d'erreur doit être envoyé vers la sortie erreur.

Une option courte ou une option longue doit obligatoirement être supportée : « -? » ou « --help », qui affiche l'aide selon le format usuel ainsi que les éventuelles limitations de l'exécutable.

Il est possible de ne développer que des options courtes ou que des options longues.

Les options suivantes, courtes ou longues, peuvent être développées :

-g ou --graph : pour supprimer l'affichage normal et le remplacer par l'affichage des mots appartenant à la réunion des ensembles de mots de tous les fichiers texte en signifiant pour chaque mot son appartenance aux ensembles. L'affichage se fait sur une colonne de plus que le nombre de fichiers. La première ligne affiche les noms des fichiers, chaque nom étant précédé du caractère de tabulation horizontale. Pour les autres lignes, la première colonne est réservée aux mots, les

3. Entre autres, les fichiers textes lesmiserables.txt , toto0.txt, toto1.txt, toto2.txt et toto3.txt sont disponibles sur l'espace de cours sous UniversiTICE.

MÀJ 22-03
Ajout de la
note.

autres au codage de l'appartenance. Les colonnes sont séparées par le caractère de tabulation. Selon que le mot appartient ou non à l'ensemble des mots du fichier spécifié en haut de la colonne, un « x » ou un « - » est affiché. Les mots sont affichés dans l'ordre lexicographique ;

- i **VALUE** ou --initial=**VALUE** : pour fixer la longueur maximale des mots à **VALUE**. Lorsque **VALUE** vaut 0, la longueur n'a aucune limite. La valeur par défaut de **VALUE** est 0 ;
- p ou --punctuation-like-space : pour donner aux symboles de ponctuation le même rôle que les caractères de la classe `isspace`.

Toutes les paramètres figurant sur la ligne de commande doivent être pris en compte. En particulier, l'exécutable doit pouvoir différencier parmi les paramètres ceux qui relèvent de la catégorie option et ceux qui n'en relèvent pas — des noms de fichiers donc, ou assimilés. Face à une option qu'il ne supporte pas, l'exécutable ne doit pas planter mais prévoir une fin d'exécution normale avec un message d'erreur approprié.

MÀJ 22-03
Ajout du
paragraphe.

À l'instar de certaines commandes Linux, la possibilité de lire les mots sur l'entrée standard au lieu de les lire dans un fichier de nom donné sur la ligne de commande peut être développée. Pour le signifier explicitement à la commande, il suffit d'utiliser « - » en lieu et place d'un nom de fichier. Dans l'affichage de la ligne en-tête, « "" » doit figurer à l'emplacement du nom de fichier manquant.

Enfin, « - » donné seul doit pouvoir spécifier que l'argument qui suit sur la ligne de commande doit être considéré comme un nom de fichier.

Modalités

Vous pouvez réaliser ce projet à deux, pas plus.

Vous soutiendrez le projet individuellement.

Votre projet sera compilable et exécutable sur les machines des salles de travaux pratiques du département d'informatique et sous Xubuntu.

Votre programme devra répondre aux diverses exigences signifiées dans la section « Mise en œuvre ». Il sera en partie testé avec des outils automatiques.

Votre projet devra être rendu avec tous les fichiers sources nécessaires, un fichier `makefile` pour produire l'exécutable (`make` sans paramètre ou avec les paramètres `all` ou `jdis`) et réaliser le nettoyage d'après production (`make` avec le paramètre `clean`), un rapport de développement. Le rapport sera fourni sous forme électronique (fichier au format `pdf`) qui précisera l'objet de chacun des modules, décrira l'implantation (dessins souhaités), commentera des exemples, explicitera les éventuelles limitations. Aucun autre fichier que ceux-là ne doit être joint ; en particulier, aucun des textes ayant servi aux exemples.

MÀJ 22-03
Précisions
pour
makefile.

Vous remettrez votre projet au plus tard le **jeudi 15 mai 2025 à 18 h 47** dans une archive au format `tar.gz` de nom `identifiant_projet.tar.gz`, où `identifiant` est l'identifiant de 8 caractères de l'un des membres du groupe.

MÀJ 25-04
Précision de
la date.

La notation est sensible à la propreté du code, à son homogénéité, aux performances de l'exécutable, à la clarté des explications fournies dans le rapport et lors de la soutenance, ainsi qu'à un passage sans encombre au révélateur `valgrind`.

MÀJ 22-03
Ajout du
paragraphe.

Hors jeux

La note attribuée au projet est 0/20 en cas :

- de demande expresse de la part de l'étudiant·e,
- d'absence de dépôt,
- d'erreur d'identifiant dans le nom de l'archive,

MÀJ 28-04
Ajout du
corps de la
section.

- d'archive non conforme : format, contenu (fichier manquant, tel un source, le fichier `makefile` ou le rapport; fichier en trop, tel un exécutable, un fichier objet, un fichier temporaire, un fichier texte qui n'est ni un source ni un fichier `makefile`),
- d'erreur apparaissant à compilation,
- de non respect de la mise en forme standard,
- pour toute recopie dans un source quelconque d'une structure normalement cachée dans la partie implantation d'un module,
- hors interface `"holdall.h"`, d'utilisation d'une interface dont les types élémentaires « à la base de votre travail » ne sont pas `const void *`;
- d'utilisation d'un forceur de type pour contrecarrer l'aspect non mutable d'une donnée (sauf pour se conformer au type de retour d'une fonction et, dans ce cas, avec une utilisation de ce genre de forceur de type immédiatement après un `return`),
- pour tout plagiat (recopie totale ou partielle d'un source tiers, dont IA⁴),
- pour méconnaissance du projet présenté lors de la soutenance.

Grille d'évaluation

En dehors des cas réhabilitaires précédemment cités, la réalisation du développement et de sa présentation est notée sur un maximum de 20 points répartis comme suit :

MÀJ 28-04
Ajout du
corps de la
section.

<i>critère</i>	<i>points</i>
pour 2 fichiers uniquement	6
pour un nombre de fichiers ou équivalents supérieur ou égal à 2 et éventuellement inférieur à une constante d'au moins 16	12
option <code>-i</code>	1
option <code>-p</code>	1
utilisation de « - » et « -- »	1
rapport et soutenance	5

Des pourcentages de points peuvent être retirés en fonction des critères suivants :

<i>critère</i>	<i>pénalités</i>
absence d'aide ou aide en inadéquation avec ce qui est développé	−20 %
absence de l'affichage normal ou de l'affichage associé à l'option « <code>-g</code> »	−50 %
plantage dans la phase de reconnaissance des options (une option non supportée doit donner lieu à un message d'erreur et à l'arrêt de l'exécutable)	−20 %
mots découpés en tranches de longueur maximale constante	−50 %
mots découpés en tranches de longueur maximale fixée par l'utilisateurice (ce qui correspondrait à une mauvaise interprétation du rôle de « <code>-i</code> »)	−40 %
longueur des mots limitée par une constante C (ce qui correspondrait à « <code>-i C</code> » avec C non nul par défaut au lieu du « <code>-i 0</code> » par défaut)	−30 %
longueur des mots limitée par une constante fixée par l'utilisateurice sans que l'option « <code>-i 0</code> » ne soit fonctionnelle	−20 %
gestion non correcte d'opérations sur les fichiers ou plantage consécutif à cette mauvaise gestion (toutes les opérations sur les fichiers doivent être testées et correctement traitées)	−30 %
erreurs signalées par <code>valgrind</code> , hors limitation propres de <code>valgrind</code>	−50 %

4. Toute expression de la forme `x` ou `!x` là où, de manière standard, est écrit en C, à l'inversion des objets de la comparaison près, `x != 0`, `x != '\0'`, `x != nullptr` ou `x == 0`, `x == '\0'`, `x == nullptr` sera considérée au moins comme suspecte.

désallocations non faites signalées par <code>valgrind</code> , même en cas d'échec	−25 %
messages d'erreur inadaptés (les textes des messages d'erreur doivent décrire les erreurs de la manière la plus appropriée possible et doivent être envoyés vers la sortie erreur)	−10 %
complexités non satisfaisantes à l'exécution	−25 %
utilisation de variables globales (hors <code>errno</code>)	−10 %
présence de nombres ou chaînes magiques	−10 %
répétitions de parties de code (le code doit être factorisé)	−20 %
fonctions auxiliaires non spécifiées ou de spécification incohérente ou incomplète	−20 %
code mal présenté, malgré l'utilisation de <code>uncrustify</code>	−25 %

Foire aux questions

- **1 Bien que vous n'avez pas encore publié le barème pour le projet, je voudrais savoir si l'on peut s'attendre à des points (bonus) si les deux implémentations (ABR et table de hachage) sont réalisés. Si mes souvenirs ne me trompent pas, c'était le cas l'année dernière. ?** MÂJ 22-03
Ajout 1-1.

Non pour cette année. Mais si vous tenez à faire voir votre maîtrise des deux implantations, utilisez la même syntaxe que l'année dernière :

-a : même chose que --words-processing=avl-binary-tree ;
-h : même chose que --words-processing=hash-table ;
-w TYPE ou --words-processing=TYPE : pour signifier le traitement des mots selon la structure de données spécifiée par TYPE. Les valeurs possibles pour TYPE sont explicites : avl-binary-tree et hash-table. La valeur par défaut de TYPE est hash-table.

- **2 A-t-on le droit d'utiliser des modules déjà existant comme « getopt » ?** MÂJ 28-03
Ajout 2-3.

Oui vous pouvez utiliser getopt.

Attention cependant à ne pas introduire de caractères ou chaînes magiques. Par exemple les « g » et « graph » pour le remplacement de l'affichage normal ne doivent apparaître qu'une fois et une seule dans tout votre code, aide comprise.

- **3 Je ne comprends pas vraiment ce que fait 'setlocale(...)', pourquoi on doit la mettre pour strcoll,... J'ai cherché dans l'internet et les manuels et après quelques tests les résultats sont les mêmes : la ponctuation reste (l'option -p ne fonctionne pas, et même en mettant par défaut la locale, elle ne change rien). J'ai essayé de mettre en second paramètre nullptr, "", "fr-FR" mais rien...**

Réponse par deux exemples reproductibles. Dans le sous-dossier courstd/7/set_mp/ d'Algo2 :

```
$ ./main2 < ../toto3.txt 2>/dev/null
-
:
?
La
Madame,
Oui
Toto
Toto,
a
c'est
ce
cote.
d'un
demande
dire
gratte
je
maitresse
me
migrateur
ne
oiseau
peux
```

```
pourrais-tu  
qu'est  
que  
qui  
se  
seul  
un
```

Le même résultat est obtenu après l'inclusion de l'en-tête `<locale.h>`, après tout appel à `setlocale` ou après le remplacement de `strcmp` par `strcoll`.

Si, en revanche, la variable `LANG` de l'environnement vaut `fr_FR.UTF-8`, et que `<locale.h>` est inclus, et que `setlocale` est appelée avec les paramètres `LC_COLLATE` et `" "`, et que cet appel est un succès, et que `strcoll` est utilisée en lieu et place de `strcmp`, on obtient :

```
$ ./main2 < ../toto3.txt 2>/dev/null  
-  
:  
?  
a  
ce  
c'est  
cote.  
demande  
dire  
d'un  
gratte  
je  
La  
Madame,  
maitresse  
me  
migrateur  
ne  
oiseau  
Oui  
peux  
pourrais-tu  
que  
qu'est  
qui  
se  
seul  
Toto  
Toto,  
un
```

► 4 Est-ce que l'aide peut être dans un fichier texte ?

Non. L'aide est partie intégrante d'un source C. Il ne doit y avoir qu'un seul endroit à modifier s'il venait l'envie de modifier le caractère ou la chaîne de caractères qui donne accès à une option.

MÀJ 07-04
Ajout 4-6.

► **5 Lorsque l'on alloue de la mémoire pour stocker un mot, il faut une taille maximum (pour le buffer), donc les mots qui font plus qu'une certaine taille doivent être coupés même si il n'y a pas de coupage spécifié ? Cela n'est pas précisé dans le projet mais il est évident qu'il y a des chaînes très longues que l'on ne pourra pas stocker complètement.**

Si la longueur maximale est spécifiée, les mots dont la longueur excède cette limite doivent être coupés (et les caractères de ces mots situés au delà de la limite ignorés). Sinon, la longueur maximale n'a pas de limite ; tous les mots qui peuvent être stockés en mémoire doivent l'être ; si un mot ne peut être stocké entièrement, l'exécutable doit prendre fin immédiatement avec un message d'erreur adapté.

► **6 J'aurais une question à vous poser au sujet de l'option "--" pour le projet d'Algorithmique. Est-ce que c'est seulement l'argument qui suit qui est considéré comme un fichier ou est-ce que c'est tous les arguments qui suivent les deux tirets ?**

L'argument qui suit. Avec « ./jdis -- --help -- - -- -- toto0.txt », quatre fichiers doivent être traités : « --help », « - », « -- » et « toto0.txt ».

► **7 j'ai quelques interrogations concernant le projet d'algo 3. Pour l'exécutable déposé, est-il possible de savoir quelle est l'implémentation choisie entre les AVL et les tables de hachage ? De plus j'aurais voulu savoir s'il était possible d'avoir des seuils concernant le temps d'exécution sur des exemples donnés de notre exécutable afin de savoir si notre implémentation est performante ou non ?** MÂJ 10-04
Ajout 7-8.

ps : je pense qu'il manque la fin de la phrase en haut de la page 6 du sujet : "Des résultats sensiblement plus cohérents peuvent toutefois être obtenus en tentant de se rapprocher de la notion de « mot » en supprimant [la ponctuation.]" ?

Dans la version 25.04 de l'exécutable déposé, c'est une table de hachage qui traite les mots.

Pour les performances, quelques dixièmes de seconde sur « Les Trois Misérables », autrement dit les textes abeeillmrss.txt, lesmiserables.txt, sssrmlleeeba.txt. Mais les temps sont à mesurer sur les machines des salles de TP.

PS : fin de phrase restaurée.

► **8 Dans le cadre de l'optimisation de mon code pour le projet, je souhaitais savoir si je pouvais limiter le nombre de fichiers traités à 64. Cette limitation me permettrait de réduire les allocations dynamiques de mémoire et d'améliorer la complexité globale du programme.**

Est-ce que cette limite est autorisée dans le cadre du projet ?

64 fichiers, c'est précisément la limite de la version 25.04 de l'exécutable déposé. Toute limite est à clairement signifier dans l'aide et dans le rapport.

► **9 Pour lire un mot, ça passe nécessairement par un tampon. Donc le mot sera coupé si sa longueur dépasse la capacité du tampon. Et c'est une erreur.** MÂJ 23-04
Ajout 9-10.

Deux fois « non ». Si une limite à la longueur des mots lus est fixée — et précisée dans l'aide comme dans le rapport — il ne s'agit pas d'une erreur : la coupure doit donner lieu à un avertissement envoyé sur la sortie erreur. Si aucune limite n'est fixée, la zone mémoire qui sert au tampon doit être réallouée pour y contenir tout le mot ; il n'y a d'erreur que si une réallocation échoue par manque de place dans le tas. Pour tester la lecture et l'éventuelle mémorisation de mots longs, récupérez le source projet/glongwords.c qui figure dans l'archive algo3_src.tar.gz sur UniversiTICE, produisez l'exécutable, dirigez sa sortie vers un fichier, exécutez votre jdis sur ce fichier.

► **10 Je ne comprends pas bien ce que vous voulez que l'on fasse avec LC_ALL=C que vous donnez dans l'aide de votre exécutable.**

Rien... Mais vous pouvez tester avec l'ordre traditionnel associé à `strcmp` ou avec celui associé à `strcoll` lorsque la variable `LANG` de l'environnement vaut `fr_FR.UTF-8` (c'est le défaut sur les machines des salles de TP). Pour de gros fichiers, ça doit être sensiblement plus rapide dans le premier cas (« `LC_ALL=C ./jdis ...` », ou alors « `export LC_ALL=C` » puis « `./jdis ...` ») que dans le second, pour un résultat différent.

► **11 Bonjour, je voudrais savoir si je peux imposer le placement des options (au moins certaines) avant de mettre les noms de fichiers ou l'utilisation de l'entrée standard ? La coupure (et prochainement la ponctuation) nécessitent que je mette ces options avant tout ce qui est manipulation de données. À part ces options cités précédemment qui doivent impérativement être en premiers, les options "-" (l'entrée standard), "--" (les noms de fichiers), "- ?" (l'aide) et "-g" (le graphe) n'ont pas d'ordre imposés, donc ne contredisent pas vos exemples. Merci, cordialement.**

MÀJ 25-04
Ajout 11-14.

Oui. S'agissant d'une limitation du programme, elle doit être indiquée dans l'aide et dans le rapport. De manière générale, tous les exécutables testés automatiquement le seront avec les options placées avant les noms des fichiers.

► **12 J'avais une petite question (ou plutôt une demande de confirmation) à propos du projet.**

Il est indiqué que nous ne pouvons pas modifier les fichiers `hashtable.h` et `bst.h`. En revanche, est-il possible de modifier `hashtable.c` et `bst.c` ? Et si nous reprenons le fichier `hashtable.c` du TP1 sans le modifier, cela aura-t-il un impact sur notre note finale ? Doit-on nécessairement proposer une implémentation personnelle, étant donné que celle du TP1 a été réalisée par vous ?

Merci d'avance pour votre réponse.

Vous pouvez modifier les parties implantation des modules `hashtable` et `holdall` si vous le souhaitez ; mais ça ne peut être que pour les rendre plus performantes encore. Quant à l'impact sur la note (au projet ?), cela dépendra de la pertinence des modifications.

► **13 Est-il possible d'avoir des indications sur la réelle possibilité de réaliser le projet avec un AVL ? et si oui, de combien est-ce moins performant qu'avec une table de hachage ?**

Une nouvelle version (25.04.1) de l'exécutable donné en démonstration (dépôt UniversiTICE) inclut la possibilité d'un traitement des mots à l'aide d'un AVL. Sur des textes dont le cardinal de l'ensemble des mots est conséquent, le temps d'exécution semble être de 2 à 3 fois supérieur ; ce qui est conforme à l'attendu.

► **14 Est-ce que l'output doit être le même que celui de votre exécutable ou pas ? Genre ça passe avec des blancs au lieu de tabulations par exemple ?**

Les formats des sorties sont spécifiés. Vos sorties (hors l'aide) doivent être exactement les mêmes que celles de l'exécutable disponible sur UniversiTICE. Faites appel à `diff`.

