Nº de laboratoire	Laboratoire 1
Nº d'équipe	06
Étudiant(s)	Marc-Olivier Allard
	Guy Martial Nzali
	Francis Ouellet
Code(s) permanent(s)	ALLM05089305
	NGOG18039008
	OUEF03099307
Cours	LOG320
Session	Hiver 2016
Groupe	01
Professeur	Patrick Cardinal
Chargés de laboratoire	Francis Cardinal

Description de l'algorithme

Notre algorithme[1] utilise une propriété des nombres premiers pour former un hachage numérique permettant de donner une valeur unique à un mot composé d'un certain ensemble de lettres. Comme chaque combinaison de lettres possède la même signature, deux *hash* identiques indiquent un anagramme. Il est aussi possible de n'effectuer la fonction de hachage qu'une seule fois pour chacun des mots avant d'effectuer les comparaison des *hash*.

La fonction de hachage utilise la valeur numérique d'un caractère selon la table ASCII pour déterminer la valeur de son nombre premier. La valeur de chacun des caractères est multipliée ensemble pour obtenir la signature du mot.

Analyse asymptotique des algorithmes

Algorithme de base

```
EstUnAnagramme(chaine1,chaine2)
pour c1 ∈ chaine1
        trouvé ← faux
        pour c2 ∈ chaine2
                si c1 = c2
                        chaine2 ← chaine2 – c2
                        trouvé ← vrai
                si trouvé = faux
                        retourner faux
        si chaine2 n'est pas vide
                retourner faux
retourner vrai
```

Soit N1 le nombre de caractères dans la chaîne1
Soit N2 le nombre de caractères dans la chaîne2

[1] Inspiré d'une solution sur StackOverflow http://stackoverflow.com/a/28948975

N° de ligne	Code source	Ordre Asymptotique des baromètres
1	EstUnAnagramme(chaine1,chaine2)	1
2	pour c1 ∈ chaine1	N1+1
3	pour c2 ∈ chaine2	(N2+1)*N1
TOTAL	//	N2*N1+N1
	O	N2*N1

Si N1 = N et N2 = N, alors N1 * N2 = N^2
O(N^2) est le pire des cas

Notre algorithme

```
EstUnAnagramme(chaine1, chaine2)
hash1 ← 1
hash2 ← 1
pour c1 ∈ chaine1
        hash1 ← hash1 * PREMIERS[c1]
pour c2 ∈ chaine2
        hash2 ← hash2 * PREMIERS[c2]
si hash1 = hash2
        retourner vrai
sinon
        retourner faux
```

Soit N1 le nombre de caractères dans C1
Soit N2 le nombre de caractères dans C2

N° 1 de ligne	Code de source	Ordre Asymptotique des barometres
1	EstUnAnagramme(chaine1, chaine2)	1
2	pour c1 ∈ chaine1	N1+1
3	pour c2 ∈ chaine2	N2+1
TOTAL	//	N1+N2 +2
	O	N1+N2

Si N1 = N et N2 = N, alors N1+N2= 2N;
O(N) est le pire des cas.

Analyse expérimentale des algorithmes

Pour l'analyse expérimentale des algorithmes, nous avons généré 15 listes de mots. Chaque fichier généré contient des mots d'une longueur fixe (1 à 15 lettres). Afin de pouvoir bien interpréter les résultats, chaque fichier contient le même nombre de mots, soit 58110 mots. Nous avons ensuite exécuté notre algorithme et l'algorithme de base sur chacune des listes de mots, et nous avons enregistré les résultats.

** * Il est important de garder en tête que notre analyse ne tiens pas compte du nombre d'anagrammes par liste, ce qui pourrait affecter le temps d'exécution des algorithmes.*

	Temps d'exécution en millisecondes (ms)		
Nombre de lettres par mots	Algorithme de base	Notre algorithme	Nombre d'anagrammes
1	45262	1615	203522242
2	86503	1422	49224112
3	125189	1412	11095556
4	171979	1454	2633372
5	137457	1310	1041784
6	315480	1302	401364
7	330161	1381	220996
8	375615	1317	143416
9	426781	1365	104654
10	457490	1334	84260
11	494196	1378	73116
12	529971	1359	68948
13	556074	1348	65186
14	578273	1376	63820

Tableau 1 : Temps d'exécution d'un algorithme de résolution d'anagramme en fonction du nombre de lettres par mots.

Algorithme de base

Il suffit d'un coup d'oeil sur le tableau ci-dessus pour remarquer que le temps d'exécution de l'algorithme de base augmente significativement avec le nombre de lettres par mots. Cet algorithme perd donc de l'efficacité avec la croissance du nombre de caractères par anagramme.

Notre algorithme

À l'opposé de l'algorithme de base, le temps d'exécution de notre algorithme reste relativement constant (et beaucoup plus bas que celui de l'algorithme de base). En effet, notre algorithme de semble pas être affecté par le nombre de caractère par anagramme.

Analyse asymptotique du programme complet

Main(args)

```
dictionnaire[] ← lireFichier(args[0])                // O(n) où n le nombre de mots
mots[] ← lireFichier(args[1])                        // O(m) où m le nombre de mots
dictionnaire[] ← nettoyer(dictionnaire[])            // O(n)
mots[] ← nettoyer(mots[])                            // O(m)
hashDictionnaire[]                                   // O(1)
hashMots[]                                           // O(1)
pour mot ∈ dictionnaire[]                            // O(n)
        hashDictionnaire[i] ← calculerHash(mot)      // O(n)

pour mot ∈ mots[]                                    // O(m)
        hashMots[i] ← calculerHash(mot)              // O(m)

total ← 0                                            // O(1)
pour hash1 ∈ hashMots[]                              // O(m)
        soustotal ← 0                                // O(m)
        pour hash2 ∈ hashDictionnaire[]              // O(n*m)
                si hash1 = hash2                     // O(n*m)
                        soustotal++                  // O(n*m)
        afficher(soustotal)                          // O(m)
afficher(total)                                      // O(1)
```

L'ordre asymptotique O du programme complet est O(n*m) où *n* et *m* sont les nombres de mots des listes reçues en paramètre.

Résultat de la démonstration

Reading first file ...
Reading second file ...
Sanitizing first file ...
Sanitizing second file ...
Il y a 12 anagramme(s) du mot cree
Il y a 17 anagramme(s) du mot aussi
Il y a 47 anagramme(s) du mot sorte
Il y a 56 anagramme(s) du mot libre
Il y a 19 anagramme(s) du mot saussure
Il y a 8 anagramme(s) du mot texte
Il y a 7 anagramme(s) du mot matrice
Il y a 21 anagramme(s) du mot significations
Il y a 9 anagramme(s) du mot france
Il y a 17 anagramme(s) du mot premieres
Il y a 248 anagramme(s) du mot tables
Il y a 18 anagramme(s) du mot billard
Il y a 31 anagramme(s) du mot seraient
Il y a 58 anagramme(s) du mot cause
Il y a 11 anagramme(s) du mot climat
Il y a 31 anagramme(s) du mot nobles
Il y a 3 anagramme(s) du mot croquet
Il y a 25 anagramme(s) du mot voulant
Il y a 14 anagramme(s) du mot jouer
Il y a 74 anagramme(s) du mot temps
Il y a 74 anagramme(s) du mot version
Il y a 8 anagramme(s) du mot interieur
Il y a 21 anagramme(s) du mot suite
Il y a 74 anagramme(s) du mot version
Il y a 145 anagramme(s) du mot table
Il y a 20 anagramme(s) du mot poussant
Il y a 37 anagramme(s) du mot billes
Il y a 15 anagramme(s) du mot cannes
Il y a 3 anagramme(s) du mot croquet
Il y a 18 anagramme(s) du mot manche
Il y a 12 anagramme(s) du mot arrigo boito
Il y a 22 anagramme(s) du mot edward gorey
Il y a 17 anagramme(s) du mot vladimir nabokov
Il y a 3 anagramme(s) du mot ted morgan
Il y a 17 anagramme(s) du mot dave barry
Il y a 15 anagramme(s) du mot glen duncan
Il y a 11 anagramme(s) du mot damon duncan
Il y a 13 anagramme(s) du mot damon albarn
Il y a 14 anagramme(s) du mot anna madrigal
Il y a 20 anagramme(s) du mot tom marvolo riddle
Il y a 11 anagramme(s) du mot buckethead
Il y a 22 anagramme(s) du mot daniel clowes
Il y a 15 anagramme(s) du mot siobhan donaghy
Il y a 12 anagramme(s) du mot i have constructed a string that needs to be shuffled
Il y a un total de 1345 anagrammes
Temps d'execution : 865 millisecondes