Exercices de Programmation & Algorithmique 1

Série 9 – Les dictionnaires
(23 novembre 2021)

Département d'Informatique – Faculté des Sciences – UMONS

Pré-requis : Dictionnaires et leurs méthodes (cours jusqu'au **Chapitre 10**).

Objectifs : Etre capable de manipuler les dictionnaires et d'en comprendre l'utilité.

---

1 Contexte : Chiffrement et déchiffrement de textes

Le chiffrement a pour but de transformer un texte afin que seules les personnes ayant la "clé" puissent le comprendre. Nous allons vous décrire deux méthodes de chiffrement ainsi qu'une méthode de déchiffrement permettant, à partir d'un texte chiffré, de deviner la clé à utiliser afin de rendre lisible ce texte.

1.1 Chiffrement par décalage

Cette méthode de chiffrement consiste à décaler chacune des lettres d'un texte d'une même valeur. Par exemple, un décalage de 3 unités pour le mot "bonjour" donne "erqmrxu" ("e" s'obtient en décalant "b" de 3 unités, et ainsi de suite). Le décalage est cyclique sur les lettres de l'alphabet ("z" décalé de 5 devient "e").

1.2 Chiffrement par substitution

Cette méthode, consiste non pas à décaler chacune des lettres d'une même valeur, mais à travailler par substitution : à chaque lettre de l'alphabet, une autre lettre de l'alphabet est associée et l'ensemble du texte est modifié en conséquence. Par exemple, si on remplace "a" par "h" et "p" par "s", le mot "papa" devient "shsh".

1.3 Découverte de clé : Analyse de fréquence

Bien qu'il n'existe à priori que 26 décalages possibles pour un texte chiffré avec un chiffrement par décalage, il existe une méthode plus rapide que de tester chaque possibilité pour déchiffrer un texte. Cette méthode est basée sur l'analyse de fréquence et, principalement, sur l'observation faite que, pour un texte francophone quelconque d'une longueur suffisante, la lettre "e" est la lettre apparaissant le plus de fois dans ce texte [1]. Dès lors, si le texte chiffré respecte cette hypothèse, il suffit d'identifier au sein de ce texte chiffré la lettre qui apparait le plus souvent, et de considérer qu'elle correspond à la lettre "e" du texte déchiffré. De là, il est aisé d'en déduire le décalage nécessaire pour déchiffrer le texte.

2 Le contrat

Pour chacune des méthodes de chiffrement, vous devez concevoir des fonctions permettant de chiffrer et déchiffrer un texte, à partir d'une clé donnée. Veuillez réfléchir à la manière dont vous allez structurer votre code avant d'implémenter et pensez à utiliser au mieux les dictionnaires, ils ont des propriétés intéressantes pour ce genre de problèmes.

1. Voici les fréquences de la langue française (dans l'ordre alphabétique) : 9.42 ; 1.02 ; 2.64 ; 3.39 ; 15.87 ; 0.95 ; 1.04 ; 0.77 ; 8.41 ; 0.89 ; 0.00 ; 5.34 ; 3.24 ; 7.15 ; 5.14 ; 2.86 ; 1.06 ; 6.46 ; 7.90 ; 7.26 ; 6.24 ; 2.15 ; 0.00 ; 0.30 ; 0.24 ; 0.32

Pour tester vos fonctions de chiffrement/déchiffrement, il vous est demandé de créer un script python `chiffrement.py` qui utilisera les arguments en ligne de commande (`sys.argv`) pour chiffrer/déchiffrer du texte. Ce script utilisera les fonctions que vous aurez implémentées ci-dessous.

1 Implémentez une fonction `lecture(nom)` permettant d'obtenir le contenu d'un fichier texte en une chaine de caractère.
Entrée : Une chaine de caractères `nom`.
Sortie : Une chaine de caractère représentant le texte contenu dans le fichier texte dénommé `nom`.

2 Implémentez une fonction `nettoyage(texte)` permettant de nettoyer un texte.
Entrée : Une chaine de caractère `texte`.
Sortie : Une chaine de caractères correspondant au texte `texte` nettoyé. Pour nettoyer un texte, il suffit de mettre toutes les lettres en minuscule et **supprimer** les espaces, les caractères de retour à la ligne et les caractères spéciaux (é,ô,...).

3 Implémentez une fonction `chiffrement_decalage(texte, u)` permettant de chiffrer un texte en utilisant la méthode de chiffrement par décalage.
Entrée : Une chaine de caractère `texte`, nettoyée, et un entier `u`.
Sortie : La chaine de caractères correspondant au texte `texte` chiffré en utilisant la méthode de chiffrement par décalage de `u` unités.

Modifiez votre script `chiffrement.py` pour qu'il puisse chiffrer du texte contenu dans un fichier en utilisant la méthode de chiffrement par décalage. Pour cela, on utilisera la commande suivante (remplacer les éléments entre <> par les valeurs souhaitées) :

```
python3 chiffrement.py d c <fichier a chiffrer> <décalage>
```

Un exemple d'utilisation pour chiffrer le fichier `texte.txt` avec un décalage de 5 serait comme suit :

```
python3 chiffrement.py d c texte.txt 5
```

4 Implémentez une fonction `dechiffrement_decalage(texte, u)` permettant de déchiffrer un texte chiffré par la méthode de chiffrement par décalage.
Entrée : Une chaîne de caractères `texte` et un entier `u`.
Sortie : La chaine de caractères correspondant au texte `texte` déchiffré, sachant que ce dernier ait été chiffré avec la méthode de chiffrement par décalage de `u` unités.

Modifiez votre script `chiffrement.py` pour qu'il puisse déchiffrer du texte contenu dans un fichier qui a été chiffré par décalage. Pour cela, on utilisera la commande suivante :

```
python3 chiffrement.py d d <fichier a déchiffrer> <décalage>
```

Test : Veuillez déchiffrer le texte se trouvant dans le fichier `decalage.txt` sachant qu'il ait été chiffré avec un décalage de 5 unités et vérifiez qu'il est lisible.

5 Implémentez une fonction `chiffrement_substitution(texte, dico)` permettant de chiffrer un texte par la méthode de chiffrement par substitution.
Entrée : Une chaîne de caractères `texte`, nettoyée, et un dictionnaire `dico`.
Sortie : La chaine de caractères correspondant au texte `texte` chiffré en utilisant la méthode de chiffrement par substitution. Les substitutions sont définies par le dictionnaire `dico` de telle manière à ce qu'une paire (*clef*, *valeur*) représente la substitution de la lettre *clef* par la lettre *valeur*. Le dictionnaire peut ne pas être complet. Dans le cas où une lettre n'a pas de valeur correspondante, elle n'est pas modifiée

Modifiez votre script `chiffrement.py` pour qu'il puisse chiffrer du texte contenu dans un fichier via la méthode de chiffrement par substitution. Pour cela, on utilisera la commande suivante :

```
python3 chiffrement.py s c <fichier a chiffrer> <fichier dictionnaire>
```

Note : Le <fichier dictionnaire> est un fichier contenant les substitutions. Chaque ligne du fichier contient la clé et la valeur d'une entrée d'un dictionnaire séparées par un espace. Par exemple, le dictionnaire `{'a': 'b', 'b': 'c'}` sera stocké dans le fichier comme suit :

```
a b
b c
```

6 Implémentez une fonction `dechiffrement_substitution(texte, dico)` permettant de déchiffrer un texte chiffré par la méthode de chiffrement par substitution.
Entrée : Une chaîne de caractères `texte` et un dictionnaire `dico`.
Sortie : La chaine de caractères correspondant au texte `texte` déchiffré, sachant que ce dernier ait été chiffré avec la méthode de chiffrement par substitution et que les substitutions ont été définies par le dictionnaire `dico` de telle manière à ce qu'une paire (*clef*, *valeur*) représentait la substitution de la lettre *clef* par la lettre *valeur*. Le dictionnaire peut ne pas être complet. Dans le cas où une lettre n'a pas de valeur correspondante, elle n'est pas modifiée. Afin de bien différenciez les lettres substituées, affichez-les en majuscule dans le texte déchiffré.

Modifiez votre script `chiffrement.py` pour qu'il puisse déchiffrer du texte chiffré par substitution contenu dans un fichier. Pour cela, on utilisera la commande suivante (<fichier dictionnaire est comme dans la question précédente) :

`python3 chiffrement.py s d <fichier a déchiffrer> <fichier dictionnaire>` **Test** : Veuillez déchiffrer le texte se trouvant dans le fichier `substitution.txt` sachant qu'il ait été chiffré avec le dictionnaire suivant :

```
dico = {  'a' : 'd', 'b' : 'c', 'c' : 'q', 'd' : 'r', 'e' : 'k', 'f' : 'n',
          'g' : 'g', 'h' : 'v', 'i' : 'u', 'j' : 't', 'k' : 'x', 'l' : 'o',
          'm' : 'i', 'n' : 'l', 'o' : 'a', 'p' : 'j', 'q' : 'y', 'r' : 'b',
          's' : 's', 't' : 'z', 'u' : 'm', 'v' : 'p', 'w' : 'h', 'x' : 'w',
          'y' : 'f', 'z' : 'e'}
```

7 Implémentez une fonction `decouvre_cle(text, i)` permettant de découvrir les substitutions qui ont permis de chiffrer le fichier texte `text` en utilisant la technique d'analyse de fréquence. Le paramètre i est un entier indiquant que l'on recherche la subtitution des i lettres les plus fréquentes de la langue française.

Modifiez votre script `chiffrement.py` pour qu'il puisse déchiffrer du texte chiffré par substitution contenu dans un fichier et dont on ne connaît pas la clé. Pour cela, on utilisera la commande suivante :

`python3 chiffrement.py cle <fichier a déchiffrer>`

Test : Tentez de découvrir le texte se trouvant dans le fichier `analyse.txt`. Etant donné que les fréquences des lettres les moins fréquentes sont très proches, des erreurs de substitution peuvent facilement se produire. Nous allons donc nous intéresser à la recherche des lettres les plus fréquentes pour garantir une bonne qualité des substitutions réalisées. Il est plus intéressant de travailler avec peu de substitutions de bonnes qualités qu'avec beaucoup de mauvaises. Faites varier le paramètre `i` de votre fonction `decouvre_cle` et déchiffrez le texte en utilisant votre fonction `dechiffrement_substitution` avec le dictionnaire contenant les i substitutions obtenues. Il vous sera facile de voir les substitutions effectuées, car la fonction `dechiffrement_substitution` impose que ces lettres soient en majuscule. Le texte `analyse.txt` n'est pas suffisamment long pour correspondre à la fréquence des lettres en langue française. Voici une fréquence approximative des lettres présentes dans celui-ci (dans l'ordre alphabétique) : 7.58 ; 0.51 ; 4.05 ; 3.92 ; 13.69 ; 1.69 ; 1.11 ; 0.76 ; 9.43 ; 0.13 ; 0.18 ; 5.9 ; 4.06 ; 7.87 ; 6.35 ; 2.75 ; 1.53 ; 6.06 ; 7.66 ; 7.92 ; 5.51 ; 0.79 ; 0.06 ; 0.25 ; 0.26 ; 0.

3 Exercices supplémentaires

★☆☆ 8

```
d = {}
d[1] = 22
d[3.14] = 24
d['a'] = 25
d[(1,2)] = 'X'
d[[3,4]] = 32
d[5,4] = [5,4]
```

— Sans implémenter, quelles sont les opérations illégales dans le programme ci-dessus ?
— Écrivez sur papier ce que contient le dictionnaire `d` après exécution (en supprimant les instructions erronées) ?
— Vérifiez vos réponses.

★☆☆ 9 Soit un texte utilisant des abréviations ("B1" pour "Bac1", "P." pour "Professeur", ...). Les abréviations et leur signification sont stockées dans un dictionnaire. Veuillez écrire une fonction `substitue(s, d)` qui, à partir d'une chaine de caractères `s` et d'un dictionnaire `d`, retourne une copie de la chaine de caractères dans laquelle les mots figurant dans le dictionnaire (clés) ont été remplacés par leur équivalent (valeur).

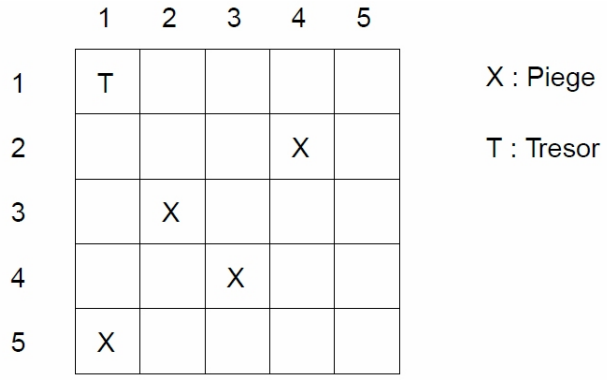
Exemple d'utilisation :

```
abbrev = {'B1' : 'Bac1', 'P.' : 'Professeur', 'M.' : 'Melot',
          'progAlgo1' : 'programmation et algorithmique 1'}
print(substitue('Les élèves de B1 aiment le cours de progAlgo1 '
                'donné par le P. M.', abbrev))
```

Astuce : Utilisez les fonctions `split()` et `join()` du module `str`.

★☆☆ 10 **Êtes-vous un bon pirate ?**

Il n'est pas toujours souhaitable d'utiliser des chaînes de caractères comme clé de dictionnaire. Par exemple, on peut utiliser un dictionnaire pour stocker un tableau partiel (une carte).



La carte ci-dessus peut être implémentée comme suit :

```
tresor = {}
tresor[(1,1)] = "trésor"
pieges = [(2,4),(3,2),(4,3),(5,1)]

for piege in pieges:
  tresor[piege] = "piège"
```

Il vous est demandé d'implémenter les fonctions suivantes :

(a) `cree_carte(n, nb_pieges)` qui prend deux naturels `n` et `nb_pieges` ≥ 2 comme paramètres et retourne un dictionnaire représentant une carte de taille `n` $\times$ `n` dans laquelle `nb_pieges` pièges et un trésor ont été placés de manière aléatoire.

(b) `recherche_tresor(n, nb_pieges)` qui crée une carte de taille `n` $\times$ `n` dans laquelle `nb_pieges` pièges et un trésor ont été placés de manière aléatoire et qui demande à l'utilisateur d'entrer une coordonnée. S'il tombe sur le trésor, l'utilisateur est millionnaire. S'il tombe sur un piège, il est mort. Sinon, vous lui proposez de tenter à nouveau.

En utilisant les dictionnaires, écrivez les fonctions suivantes :

★☆☆ 11 `words_by_length()` qui retourne un dictionnaire associant à une valeur entière v, la liste des mots dans `words.txt` de longueur v. Quels sont les mots de longueur 21 ? Quel est la taille la plus fréquente ?

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
26	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

FIGURE 1 – Carré de Vigenère

★☆☆ 12 `file_histogram(file)` qui prend en paramètre le nom d'un fichier texte `file` (chaine de caractères) et qui retourne un dictionnaire contenant la fréquence de chaque lettre dans `file` (inspirez-vous de la fonction `histogram` vue au cours). Testez la fonction précédente avec le fichier `CorbRenard.txt` et `words.txt`.

★☆☆ 13 `compte_mots(texte)` qui, étant donné un texte, retourne un dictionnaire dans lequel chaque clé est un mot du texte et la valeur associée à cette clé est le nombre de fois que ce mot apparait dans le texte, sans utiliser la fonction `count()` sur le texte. Cette fonction doit être insensible à la casse.

★★★ 14 `vowels_histogram(file)` qui prend en paramètre le nom d'un fichier texte `file` (chaine de caractères) et qui retourne un dictionnaire contenant la fréquence de chaque suite de voyelles (par ex. "i", "e", "oe", "oui", "eui", ...) dans `file`

3.1 Chiffrement de Vigenère

Source : Simon Singh, `"Histoire des codes secrets"`, JC Lattès, 1999, p.73-77.

La première étape de ce chiffrement consiste à construire un *carré de Vigenère*, tel que montré en figure 1, fait de l'alphabet clair suivi de 26 alphabets chiffrés, chacun d'eux étant décalé d'une lettre supplémentaire par rapport au précédent. Ainsi la ligne 1 est l'alphabet ayant un décalage d'une unité, la ligne 2 un décalage de deux unités, et ainsi de suite. La ligne supérieure du carré en minuscules, est l'alphabet clair, et vous pouvez en chiffrer chaque lettre selon l'un des 26 alphabets chiffrés. Par exemple, si le numéro choisi est 2, la lettre a est chiffrée par c, mais si l'on choisit le 12, alors la lettre a est transcrit m.

Si l'expéditeur n'utilisait qu'un seul de ces alphabets pour chiffrer le message tout entier, ce serait l'application simple du chiffrement par décalage. Mais le chiffrement de Vigenère impose d'utiliser une ligne différente du carré de Vigenère pour chiffrer chaque lettre. Plus précisément, l'expéditeur pourrait chiffrer la première lettre selon la ligne 5, la deuxième selon la ligne 14, la troisième selon la ligne 21, et ainsi de suite.

Pour débrouiller le message, il est important que le destinataire ait connaissance de la ligne choisie

pour chiffrer chaque lettre, et donc il doit y avoir un système convenu de passage de l'une à l'autre. Cet accord est obtenu par un mot-clef. Pour voir comment on utilise une clef afin de chiffrer un bref message, chiffrons la phrase "appeler nord troupes ville", en utilisant "ROUGE" comme mot-clef.

Tout d'abord, le mot-clef est épelé bien clairement au-dessus du message, et répété en boucle de sorte que chaque lettre du message soit associée à une lettre de la clef. Le texte chiffré est alors construit comme suit. Pour chiffrer la première lettre, a, commencez par identifier la lettre de la clef placée juste au dessus, à savoir R, qui détermine une ligne particulière du carré de Vigenère. La ligne commençant par R est la ligne 17, est c'est elle qui va définir l'alphabet à utiliser pour substituer une lettre à la lettre originale a. A partir de là, on repère la colonne commençant par a et l'on voit qu'elle coupe la ligne 17 sur R.

Mot-clef	`ROUGEROUGEROUGEROUGEROU`
Clair	`appelernordtroupesville`
Crypté	`RDJKPVFHUVUHLUYGSMBMCZY`

On recommence ensuite le processus pour les lettres suivantes. Lorsque l'on a épuisé toutes les lettres du mot-clef, on recommence toujours avec le même mot-clé, et donc les mêmes lignes du tableau. Donc pour la sixième lettre du message, on revient à la première lettre de la clef.

★★☆ 15 Implémentez une fonction `dechiffrement_vigenere(texte, mot)` permettant de déchiffrer la chaîne de caractères `texte` sachant qu'il ait été chiffré en utilisant la méthode du chiffrement de Vigenère avec le mot-clef `mot`. Cette fonction retourne la chaîne de caractères correspondant au texte déchiffré.

Test : Veuillez déchiffrer le texte se trouvant dans le fichier `vegenere.txt` sachant qu'il ait été chiffré avec le mot-clef `algorithme` et vérifiez qu'il est lisible.

★★☆ 16 Implémentez une fonction `chiffrement_vigenere(texte, mot)` permettant de chiffrer la chaîne de caractères `texte`, nettoyée, en utilisant la méthode de chiffrement de Vigenère avec le mot-clef `mot`. Cette fonction retourne la chaîne de caractères correspondant au texte chiffré.