Institut Informatique Claude Chappe

Master Informatique
Unité 177UD05
Programmation fonctionnelle

TP n°4 - Le chiffre de César

Le chiffre de César est un système simple de chiffrement basé sur des substitutions mono-alphabétiques ayant les propriétés suivantes :

- chaque lettre du message clair est décalée pour obtenir une nouvelle lettre ;
- le décalage est le même pour un même message et il est gardé secret ;
- le décalage est une opération modulaire sur la taille de l'alphabet ;
- le déchiffrement consiste à faire l'opération inverse.

Pour simplifier, on considèrera dans un premier temps, que l'on traite que des caractères majuscules (pas de minuscules, pas d'espace et pas de ponctuations).

a) Ecrivez une fonction *let2int :: Char -> Int*.
Utilisez la fonction *ord :: Char -> Int* du module *Data.Char*.

b) Ecrivez la fonction *int2let :: Int -> Char*.
Utilisez la fonction *chr :: Int -> Ord* du module *Data.Char*.

c) Ecrivez la fonction *shift :: Int -> Char -> Char* qui effectue le décalage d'un caractère. Utilisez la fonction *mod* pour faire un calcul modulaire.
Exemples : *shift 3 'A' = 'D' shift (-3) 'A' = 'X'*

d) Ecrivez une fonction *cypher :: Int -> String -> String* avec une liste par compréhension.
Exemples : *cypher 3 "MESSAGE" = "PHVVDJH"*
cypher (-3) "PHVVDJH" = "MESSAGE"

Pour décrypter un message chiffré (opération qui consiste à retrouver le message clair mais sans connaître la clé de chiffrement), on utilise une cryptanalyse par analyse fréquentielle.

L'analyse fréquentielle est basée sur le fait que, dans chaque langue, certaines lettres ou combinaisons de lettres apparaissent avec une certaine fréquence. Par exemple, en français, le *E* est la lettre la plus utilisée, suivie du *A* et du *S*. Inversement, le *W* est peu utilisé.

	A	B	C	D	E	F	G	H	I	J	K	L	M
Français	9,42	1,02	2,64	3,39	15,87	0,95	1,04	0,77	8,41	0,89	0,001	5,34	3,24
Anglais	8,08	1,67	3,18	3,99	12,56	2,17	1,80	5,27	7,24	0,14	0,63	4,04	2,60

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Français	7,15	5,14	2,86	1,06	6,46	7,90	7,26	6,24	2,15	0,001	0,30	0,24	0,32
Anglais	7,38	7,47	1,91	0,09	6,42	6,59	9,15	2,79	1,00	1,89	0,21	1,65	0,07

Ces informations permettent aux cryptanalystes de faire des hypothèses sur le texte clair, à condition que l'algorithme de chiffrement conserve la répartition des fréquences, ce qui est le cas pour des substitutions mono-alphabétiques et poly-alphabétiques, et que le message est suffisamment long.

On représentera une table des fréquences par la définition suivante :

table :: [Float]
table = [9.42, 1.02, ... , 0.32]

L'objectif des exercices suivants est d'obtenir une fonction qui retournera une liste des fréquences d'apparition de chaque caractère d'une chaîne.

e) Commencez par écrire une fonction *percent :: Int -> Int -> Float* qui calcul le pourcentage d'un entier par rapport à un autre entier. On utilisera la fonction *fromIntegral :: Int -> Float*. Exemple : *percent 5 15 = 33.333336*

f) Ecrivez la fonction *count :: Char -> String -> Int* qui compte le nombre d'occurrences d'un caractère dans une chaîne. Utilisez une liste par compréhension. Exemple : *count 's' "Mississipi" => 4*

g) Ecrivez la fonction *freqs :: String -> [Float]* qui retourne une liste des fréquences d'apparition des lettres d'une chaîne pour l'alphabet ['A'..'Z']. Utilisez une liste par compréhension.
Exemple : *freqs "ABBCCCDDDDEEEEEA" =>*
[12.5, 12.5, 18.75, 25.0, 31.25, 0.0, ... ,0.0] (liste de 26 éléments)

L'objectif est maintenant d'écrire une fonction de décryptage. On commence par écrire une fonction qui calcul la différence entre deux listes de fréquences. Plus cette différence est petite plus les deux listes sont proches.

h) Une méthode standard pour comparer une liste *os* de fréquences observées avec une liste *es* de fréquences attendus est le calcul du χ^2 (« khi-deux » ou « khi carré ») défini de la manière suivante :

$$\sum_{i=0}^{n-1} \frac{(os_i - es_i)^2}{es_i}$$

Complétez la fonction *chisqr* suivante qui calcul la valeur khi-deux.
chisqr :: [Float] -> [Float] -> Float
chisqr os es = sum [... | (o, e) <- zip os es]

i) Ecrire une fonction *rotate :: Int -> [a] -> [a]* de rotation de n positions d'une liste. *drop* et *take* sont vos amies.
Exemple : *rotate 3 [1,2,3,4,5] => [4,5,1,2,3]*

Les fonctions *chisqr* et *rotate* seront utilisées pour générer la liste de toutes les valeurs de khi-deux d'un message chiffré. Par exemple :
table' = freqs (cypher 3 "MESSAGE")
[chisqr (rotate n table') table | n <- [0..25]]

j) Ecrire une fonction *positions :: Eq a => a -> [a] -> [Int]* qui retourne une liste des positions d'un élément donné dans une liste donnée. Vous pouvez utilisez une liste par compréhension ainsi que la fonction *zip :: [a] -> [b] -> [(a, b)]*.
Exemple : *positions True [True, False, True, False]* => [0,2]

k) Enfin, complétez la fonction *crack :: String -> String* qui décrypte une chaîne chiffrée. Cette fonction va chercher la valeur de décalage probable en cherchant la valeur minimale de khi-deux.

```
crack :: String -> String
crack xs = cypher (-factor) xs
   where factor = head (positions (minimum chitab) chitab)
         chitab = ...
         table' = ...
```

Exemple : *crack (cypher 3 "CETPESTDURETLONG")*
=> "CETPESTDURETLONG"

l) Modifiez votre programme pour traiter une chaîne normale contenant des espaces, des minuscules, des majuscules et de la ponctuation. Etudiez et trouvez les fonctions du module *Data.Char* qui vous seront utiles.

Exemple : *crack (cypher 3 "Ce TP est dur et long !")*
=> "CETPESTDURETLONG"

Références web

Site officiel Haskell
https://www.haskell.org

Site des modules officiels Haskell
https://downloads.haskell.org/~ghc/latest/docs/html/libraries/