BCPST 1 — Lycée Fénelon

Corrigé du devoir d'infomatique

Dictionnaires, Tris et Dichotomie

18-03-2023 — Informatique

Recherche d'un élément dans une liste triée

1. Les fonctions correctes sont les fonctions `find3`, `find4`, et `find7`.

Justification (non demandée) :

- `find1` fonctionne sur une liste triée dans l'ordre croissant.
- `find2` regarde si *tous* les éléments de `L` sont `x`.
- `find5` risque de tomber dans une boucle infinie, par exemple avec `find5(1, [0])`.
- `find6` regarde si *le premier* élément de `L` est `x`.

Et oui, `find7` fonctionne bien car la liste est triée dans l'ordre décroissant, donc on peut s'arrêter dès qu'on trouve un élément plus petit que `x`.

Décodage d'un texte

2. a) $v_{-1} =$ `abors`

b) $(v_{-1})_1 =$ `bcpst`. On remarque que $(v_{-1})_1 = v$.

Plus généralement, pour tout message m, pour tout $k \in \mathbb{Z}$, $(m_k)_{-k} = m$ (non demandé, mais important à remarquer pour la question 4c).

3. a)

```
def dico_cesar(k):
    dico = {}
    for clef, valeur in dico_code_cesar0.items():
        dico[clef] = (valeur + k) % 26
    return dico
```

b) `dico_cesar` est 26-périodique, donc `dico_cesar(26)` est égal à `dico_cesar(0)`, c'est à dire `dico_cesar0`.

4. a) `alphabet[dico_cesar0[c]]` est égal à `c`.

b)

```
def code_cesar(message, k):
    lettre_vers_position = dico_cesar(k)
    m_k = ""
    for c in message:
        if c == ' ':
            m_k += ' '
        else:
            m_k += alphabet[lettre_vers_position[c]]
    return m_k
```

c) Pour décoder un message, il suffit de décaler dans l'autre sens.

```
def decode_cesar(message_code, k):
    return code_cesar(message_code, -k)
```

5. a)

```
def caractere_le_plus_frequent(message):
    c_to_count = {}
    for c in message:
        if c != ' ':
            if c in c_to_count:
                c_to_count[c] += 1
            else:
                c_to_count[c] = 1
    maxcount, maxc = 0, 'e'
    for c, count in c_to_count.items():
        if count > maxcount:
            maxcount, maxc = count, c
    return maxc
```

b)

```
def devine_k(x):
    return (dico_cesar0[x] - dico_cesar0['e']) % 26
```

6. a)

```
def craque_code_cesar(message_code):
    x = caractere_le_plus_frequent(message_code)
    k = devine_k(x)
    return decode_cesar(message_code, k)
```

b) La technique de craquage par analyse de fréquence ne fonctionne que si `e` est effectivement la lettre la plus fréquente dans le message originel. Par exemple, cela échoue sur le texte suivant car les lettres les plus fréquentes sont `a` et `m`

```
j aime la bio mais j aime mieux l informatique
```