Compte-rendu de TD1
Programmation et structures de données

Max BOILLEDIEU - Prom23

13 mars 2024

Table des matières

1 Exercices 1 et 2

Pour sélectionner le mot le plus long du dictionnaire que l'on puisse faire avec notre tirage, il faut déjà lire le fichier, en langage Python, on utilise les instructions suivantes :

```
f = open("frenchssaccent.dic", "r")
words = []
for ligne in f:
        words.append(ligne[0:len(ligne)-1])
f.close()
```

Ensuite on se propose de créer une fonction *test_word* qui va déterminer si on peut former un mot w avec un jeu de lettres t tiré aléatoirement. On propose deux versions, une utilisant la position des lettres dans les mots et une autre utilsant l'instruction *.remove*() d'une liste :

1.1 Version 1

```
def test_word(w,t):
        n = len(w)
        index = -1
        t = list(t)
        for k in range(n):
                m = len(t)
                for i in range(m):
                        if w[k] == t[i]:
                                index = i
                if index != -1:
                        del t[index]
                        index = -1
                else:
                        return False
        return True
```

1.2 Version 2

```
def test_word2(w,t):
        t = list(t) # Copie
        for l in w:
                if l in t:
                        t.remove(l)
```

```
                else:
                        return False
        return True
```

Dans cet algorithme, on vient successivement parcourir les lettres du mot w, puis si elle existe dans les lettres du tirage, on l'enlève ensuite du tirage jusqu'à ce qu'il n'y ait plus de lettres disponibles.

On crée alors la fonction *seek_longest_word* qui va trouver le mot le plus long que l'on peut former uniquement avec les lettres du tirage :

```
def seek_longest_word(w,l : list):
        m = 0
        word = ""
        for w in words:
                if test_word2(w,l) and m < len(w):
                        m = len(w)
                        word = w
        return word
```

Remarque : Une autre version plus coûteuse existe en transformant les listes de lettres en dictionnaires.

2 Exercice 3

Pour stocker les scores associés à chaque lettre, l'idée la plus logique de structure de données est le dictionnaire. En effet, pour chaque clé (lettre), on y associe une valeur (son score). Par la suite, on utilisera donc un dictionnaire. L'implémentation en Python est la suivante :

```
SCORE = {'?' : 0, 'a' : 1, 'e' : 1, 'i' : 1, 'l' : 1,
'n' : 1, 'o' : 1,'r' : 1, 's' : 1, 't' : 1, 'u' : 1,
'd' : 2, 'g' : 2, 'm' :2, 'b' : 3, 'c' : 3, 'p' : 3,
'f' : 4, 'h' : 4, 'v' : 4, 'j' : 8, 'q' : 8, 'k' : 10,
'w' : 10, 'x' : 10, 'y' : 10, 'z' : 10}
```

2.1 Calcul du score d'un mot

Pour calculer le score d'un mot, on somme le score de toutes les lettres, on crée alors la fonction *score* :

```
def score(w):
        s = 0
```

```
        for l in w:
                s +=SCORE[l]
        return s
```

2.2 Détermination du mot avec le plus grand score

On utilisera encore un algorithme de maximisation où l'on regarde tous les mots et on sélectionne celui avec le plus grand score. C'est alors un maximum local car deux mots peuvent avoir le même score. On propose le code suivant :

```
def max_score(t,w):
        m = 0
        word = ""
        for w in words:
                if test_word2(w,l) and m < score(w):
                        m = score(w)
                        word = w
        return word, m
```

3 Exercice 4

Si on ajoute une nouvelle règle, le joker "?" valant un score nul et pouvant remplacer n'importe quelle autre lettre. Il nous suffit de remplacer une entrée au dictionnaire $SCORE$ comme suit :

```
SCORE['?'] = 0
```

On modifie également la fonction $test\_word$ pour pouvoir utiliser les jokers restants, elle devient alors :

```
def test_word2(w,t):
        t = list(t) # Copie
        for l in w:
                if l in t:
                        t.remove(l)
                else:
                        if '?' in t:
                                t.remove('?')
                        else:
                                return False
        return True
```

4 Exécutions

Avec le tirage : $['s','v','u','c','f','l','i','a']$, on obtient le mot *culais* pour le plus long mot. Avec le tirage : $['h','n','?','r','a','x','b','r']$, on obtient le mot *borax* pour le plus grand score (16).