

Corrigé du TP Informatique 29

Exercice 1

1. On saisit :

```
>>> dico={"chien":"dog","chat":"cat"}
```

On observe :

```
>>> dico
{'chat': 'cat', 'chien': 'dog'}
>>> type(dico)
<class 'dict'>
```

2. On observe :

```
>>> dico["chien"]
'dog'
>>> dico["chat"]
'cat'
>>> dico["oiseau"]
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    dico["oiseau"]
KeyError: 'oiseau'
```

On accède à la valeur associée à une clé en appelant directement la clé dans le dictionnaire. Si une clé n'existe pas, on a un message d'erreur.

3. On saisit :

```
>>> "chien" in dico
True
>>> "bird" in dico
False
```

L'instruction `in` permet de tester la présence d'une clé dans un dictionnaire.

4. On saisit :

```
>>> dico["oiseau"]="bird"
```

On observe :

```
>>> dico
{'chien': 'dog', 'chat': 'cat', 'oiseau': 'bird'}
>>> dico.keys()
dict_keys(['chien', 'chat', 'oiseau'])
>>> dico.values()
dict_values(['dog', 'cat', 'bird'])
```

5. On saisit :

```
>>> list(dico.keys())
['chien', 'chat', 'oiseau']
```

6. On saisit :

```
>>> for cle in dico:
    if "d" in dico[cle]:
        print(cle)
```

On obtient :

```
chien
oiseau
```

Exercice 2

```
def points(mot):
    scrabble={"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,"H":4,"I":1,"J":8,
    "K":10,"L":1,"M":2,"N":1,"O":1,"P":3,"Q":8,"R":1,"S":1,"T":1,"U":1,
    "V":4,"W":10,"X":10,"Y":10,"Z":10}
    tot = 0
    for s in mot:
        tot += scrabble[s]
    return tot
```

Exercice 3

1. On saisit :

```
def elt(L):
    """elt(L:list)->list
    Renvoie la liste des éléments de L sans doublon"""
    dico={}
    for x in L:
        if x not in dico:
            dico[x]=True
    return list(dico.keys())
```

2. On saisit :

```

def doublon(L):
    """doublon(L:list)->bool
    Renvoie True si L contient un doublon, False sinon"""
    dico={}
    for x in L:
        if x not in dico:
            dico[x]=True
        else:
            return True
    return False

```

3. On saisit :

```

def kpremiers(n,k):
    """kpremiers(n:int,k:int)->list
    Renvoie les k premiers éléments distincts
    générés par tirage uniforme dans [[1,n]]"""
    nb=0
    dico={}
    while nb<k:
        tirage=rd.randint(1,n+1)
        if tirage not in dico:
            dico[tirage]=True
            nb+=1
    return list(dico.keys())

```

Exercice 4

1. On saisit :

```

def frequence(texte):
    """frequence(texte:str)->dict
    Renvoie le dictionnaire des fréquences d'apparition
    des lettres du texte"""
    n=len(texte)
    dico={}
    for x in texte:
        if not x in dico:
            dico[x]=1/n
        else:
            dico[x]+=1/n
    return dico

```

2. On saisit :

```

def dist(D1,D2):
    """dist(D1:dict,D2:dict)->float
    Calcule la distance entre deux dictionnaires
    de fréquence d'apparition de lettres"""

```

```

A="abcdefghijklmnopqrstuvwxy"
res=0
for x in A:
    if x in D1:
        f1=D1[x]
    else:
        f1=0
    if x in D2:
        f2=D2[x]
    else:
        f2=0
    res+=abs(f1-f2)
return res

```

3. On obtient :

Texte 1

Extrait : chapteridowntherabbitholealicewasbeginningtogetverytiredofsi
Anglais

Texte 2

Extrait : chapitrepremierunecueilfuyantlanneefutmarqueeparunevenementb
Français

Texte 3

Extrait : elingeniosohidalgodonquijotedelamanchatasayjuangallodeandra
Espagnol

Texte 4

Extrait : derfreundlicheherrdokterdenentscheidgegebenhattedassdaski
Allemand

Exercice 5

1. On saisit :

```

def freqmax(L):
    """freqmax(L:list)->(int,int)
    Renvoie un élément d'occurrence maximale et son nombre d'occurrences"""
    dico={}
    for x in L:
        if x in dico:
            dico[x]+=1
        else:
            dico[x]=1
    nb,res=0,None
    for x in dico:
        if dico[x]>nb:
            nb,res=dico[x],x
    return res,nb

```

2. On saisit :

```
def freqmax_occ(L):
    """freqmax_occ(L:list)->(int,list)
    Renvoie un élément d'occurrence maximale
    et la liste des indices de ses occurrences"""
    dico={}
    n=len(L)
    for k in range(n):
        x=L[k]
        if x in dico:
            dico[x].append(k)
        else:
            dico[x]=[k]
    nb,res=0,None
    for x in dico:
        if len(dico[x])>nb:
            nb,res=len(dico[x]),x
    return res,dico[res]
```

Exercice 6

On exécute :

```
for k in range(N):
    deb=time.time()
    if k not in dico:
        dico[k]=k
    fin=time.time()
    t_time.append(fin-deb)
```

On observe :

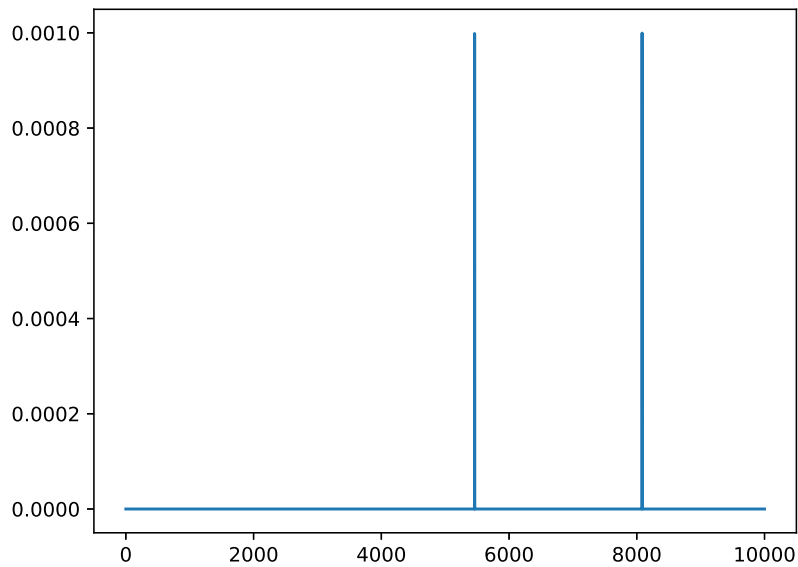


FIGURE 1 – Nouvelles affectations dans un dictionnaire

On peut conjecturer que l'instruction `dico[k]=k` qui crée une nouvelle entrée dans `dico`, à savoir l'association `k : k`, s'effectue à coût constant, en $O(1)$. Puis, on exécute :

```
for k in range(N):
    tirage=rd.randint(1,100)
    deb=time.time()
    if tirage in dico:
        dico[tirage].append(k)
    else:
        dico[tirage]=[k]
    fin=time.time()
    t_time.append(fin-deb)
```

On observe :

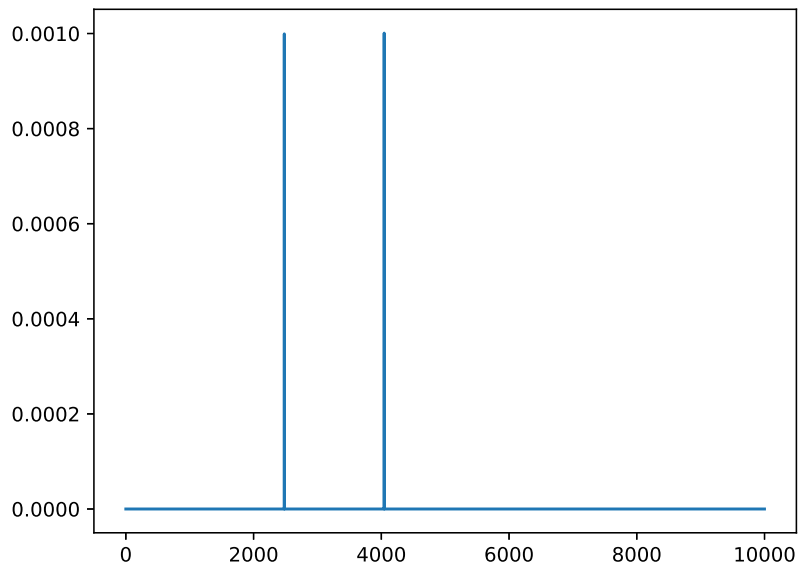


FIGURE 2 – Recherches et nouvelles affectations dans un dictionnaire

On réalise un test d'appartenance puis on crée une nouvelle entrée dans `dico` et on peut conjecturer que les deux instructions sont en $O(1)$. Enfin, on exécute :

```

for k in range(N):
    tirage=rd.randint(1,100)
    deb=time.time()
    if tirage in dico:
        dico[tirage].append(k)
    else:
        dico[tirage]=[k]
    fin=time.time()
    t_time.append(fin-deb)

```

On observe :

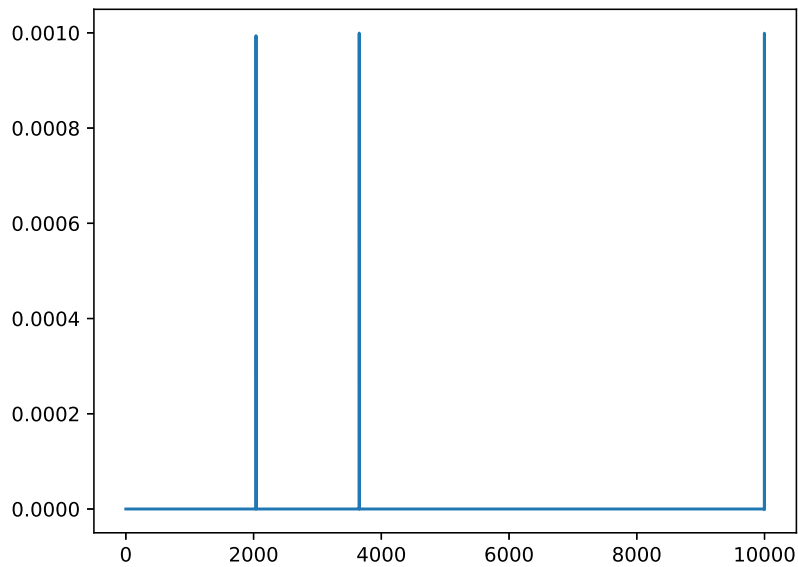


FIGURE 3 – Recherches, modifications et nouvelles affectations dans un dictionnaire

On conjecture, comme pour les situations précédentes que les instructions de test d'appartenance, création d'une nouvelle entrée ou modification d'une entrée existante avec augmentation de taille sont toutes en $O(1)$.

C'est en effet une des prouesses du type `dict` : il s'agit d'une implémentation de ce qu'on appelle des *tables de hachage*, structures conçues pour réaliser efficacement le stockage de couples (clé, valeur).