TD Informatique TSI2

DICTIONARY

SOLUTION

EXERCICE 01

Donnons deux exemples

```
In [1]: d1 = {'true': 'vrai', 'and': 'et', 'or ': 'ou'}
In [2]: d2 = {}
In [3]: for c,v in d1.items():
            d2[v] = c
In [4]: d2
Out[4]: {'vrai': 'true', 'et': 'and', 'ou': 'or '}

In [5]: d1 = {'true' : 'vrai', 'and' : 'vrai', 'or' : 'ou' }
In [6]: d2 ={}
In [7]: for c,v in d1.items():
            d2[v] = c
In [8]: d2
Out[8]: {'vrai': 'and', 'ou': 'or'}
```

EXERCICE 02

```
In [3]: def min_max(nombres):
   ...:     d = {"min": nombres[0], "max": nombres[0]}
   ...:     for n in nombres:
   ...:         if n > d["max"]:
   ...:             d["max"] = n
   ...:         elif n < d["min"]:
   ...:             d["min"] = n
   ...:     return d
   ...:
In [4]: min_max([8,1,4,7,10,-7,-8,4,0])
Out[4]: {'min': -8, 'max': 10}
```

EXERCICE 03

1.

```
In [1]: import sys
In [2]: sys.hash_info
Out[2]: sys.hash_info(width=64, modulus=2305843009213693951,
inf=314159, nan=0, imag=1000003, algorithm='siphash24',
hash_bits=64, seed_bits=128, cutoff=0)

In [3]: 2**61-1
Out[3]: 2305843009213693951
```

2.

```
In [4]: M=2**61-1
In [5]: def hachage(p,q):
            h = int(abs(p)*M/abs(q)) % M
            if p*q <0 :
                  h = -h
            if p/q == -1 :
                  h = -2
            return h
```

3.

```
In [6]: hachage(3,2)
Out[6]: 1152921504606846977

In [7]: hash(3/2)
Out[7]: 1152921504606846977

In [8]: hash(1.5)
Out[8]: 1152921504606846977

In [9]: hachage(115,100), hash(1.15)
Out[9]: (345876451382053889, 345876451382053889)

In [10]: hachage(-7,3),hachage(7,-3),hash(-7/3)
Out[10]: (-768614336404564994, -768614336404564994, -768614336404564994)
```

EXERCICE 04

1. $h(5) = 5$ et $5\%8 = 5$ alors à l'indice 5, on met $5, 2$.
$h(34) = 34$ et $34\%8 = 2$, on met à l'indice 2, 34, -3.
Enfin $h(105) = 105$ et $105\%8 = 1$, on met à l'indice 1, 105, 4.
Enfin, on met -1 et -1 à l'indice 0, Al'indice 3 et l'indice 4. Cela donne :

indice 0	-1	-1
indice 1	105	4
indice 2	34	-3
indice 3	-1	-1
indice 4	-1	-1
indice 5	5	2
indice ...	...	...

2. On a $2\%8 = 2$ et $10\%8 = 2$. Les degrés et les coefficients de x^2 et x^{10} sont placés à l'indice 2, on a un problème de collision.

3. Un exemple est le polynôme $1 + x^8 + x + x^2 + x^3$. En effet, le premier monôme utilise l'indice 0. Puis comme $8\%8 = 0$, on met le deuxième monôme à l'indice 1. Puis c'est le même pb pour le suivant.

À la fin, on a le tableau :

indice 0	0	1
indice 1	8	1
indice 2	1	1
indice 3	2	1
indice 4	3	1
indice ...	...	...

4. Pour l'exemple $4x^3 - 2x^5 + 4x^9$:
Pour le premier tableau, pas de souci.
Pour le second tableau :
Pour $d = 3$, on a : $3\%8 = 3$ et à l'indice 3, on met 0 car c'est à l'indice 0 du 1er tableau que l'on a le degré 3.
Pour $d = 5$, on a : $5\%8 = 5$ et à l'indice 5, on met 1 car c'està l'indice 1 du 1er tableau que l'on a le degré 5.
Pour $d = 9$, on a : $9\%8 = 1$ et à l'indice 1, on met 2 car c'est à l'indice 2 du 1er tableau que l'on a le degré 9
Cela donne bien le tableau.
Passons à $3x^5 - x^{18} + 7x^{20}$. Le premier tableau ne pose pas de souci.
Pour le second tableau :
Pour $d = 5$, on a : $5\%8 = 5$ et à l'indice 5, on met 0 car c'est à l'indice 0 du 1er tableau que l'on a le degré 5.
Pour $d = 18$, on a : $18\%8 = 2$ et à l'indice 2, on met 1 car c'est à l'indice 1 du 1er tableau que l'on a le degré 18.
Pour $d = 20$, on a : $20\%8 = 4$ et à l'indice 4, on met 2 car c'est à l'indice 2 du 1er tableau que l'on a le degré 20.
Puis on complète avec des -1. Cela donne :

indice 0	5	3
indice 1	18	-1
indice 2	20	7
indice 3	0	0
...	...	...

indice 0	-1
indice 1	-1
indice 2	1
indice 3	-1
indice 4	2
indice 5	0
indice 6	-1
indice 7	-1

EXERCICE 05

1.

```
In [1]:  def occurrences(nombres):
             d = { }
             for n in nombres:
                 if n in d :
                     d[n] = d[n] + 1
                 else :
                     d[n] = 1
             return d
In [2]: occurrences([1,5,5,2,0,-5,-4,-5,-5,7])
Out[2]: {1: 1, 5: 2, 2: 1, 0: 1, -5: 3, -4: 1, 7: 1}
In [3]: occurrences([3,5,-2,3,3,-2,3])
Out[3]: {3: 4, 5: 1, -2: 2}
```

La fonction `occurrences` qui prend en paramètre une liste de nombres renvoie un dictionnaire dont les clés sont les différents nombres de la liste avec pour valeur le nombre d'occurrences de chaque nombre. Par exemple `occurrences([3,5,-2,3,3,-2,3])` vaut $\{-2:2; 3:4, 5:1\}$.

```
In [4]: def taille(d):
            t = 0
            for n in d :
                t = t + d[n]
            return t
In [5]: taille({3: 4, 5: 1, -2: 2})
Out[5]:  7
```

On a recréé la fonction `len` appliquée au dictionnaire et non pas à une liste.

```
In [6]: def compare(nombres1, nombres2):
           d1 = occurrences(nombres1)
           d2 = occurrences(nombres2)
           for n in d1:
              if n not in d2 :
                 return False
              elif d1[n] != d2[n]:
                 return False
           return True
In [7]: compare([1,5,5,2,0,-5,-4,-5,-5,7],[3,5,-2,3,3,-2,3,0,5,1])
Out[7]: False
```

EXERCICE 06

1-a

```
In [3]: def derive1(p):
   ...:     if len(p) == 1:
   ...:         return [0]
   ...:     dp = []
   ...:     for deg in range(1,len(p)):
   ...:         dp.append(p[deg] * deg)
   ...:     return dp
In [4]: derive1([4]) , derive1([2,0,3,5])
Out[4]: ([0], [0, 6, 15])
```

1-b Dans tous les cas, la liste a pour longueur $n+1$ si n est le degré du polynôme. Donc la complexité est en $O(n)$.

2-a

```
In [5]: def derive2(p):
   ...:     if len(p) == 1 and 0 in p:
   ...:         return {0: 0}
   ...:     dp ={}
   ...:     for deg in p:
   ...:         if deg != 0:
   ...:             dp[deg - 1] = p[deg] * deg
   ...:     return dp
In [6]: derive2({0 : 4}), derive2({0: 2,2: 3,3: 5})
Out[6]: ({0: 0}, {1: 6, 2: 15})
```

2-b La complexité dépend du nombre de monômes non nuls. Donc dans le pire des cas, avec $n+1$ monômes non nuls de degré allant de 0 à n, la complexité reste linéaire en n et donc est un $O(n)$.