

1 Consignes

Pour récupérer les fichiers du tp, connectez-vous au site de la vie scolaire. Un archive vous a été envoyée, contenant les documents utiles pour le tp. Le fichier `tp_template.py` contient les codes du tp prêt à compléter. Le fichier `tp_test.py` contient les différentes fonctions de test que les fonctions que vous écrivez doivent passer avec succès.

Vous sauvegarderez ces fichiers dans votre répertoire personnel, avec un chemin de la forme :

NSI/Chapitre x - nom du chapitre/TP/TP1 - nom du tp/

et vous renommerez le fichier `tp_templates.py` en `tp_nom_prénom.py`.

2 Écriture binaire

Dans la mémoire de l'ordinateur, toutes les données (nombres, chaînes de caractères, mais aussi images, son...) sont stockées en utilisant uniquement des 0 et des 1. Pour représenter un nombre dans la mémoire de l'ordinateur, on utilise donc son écriture dans la base 2.

On rappelle que si un nombre n s'écrit $\overline{10110001}^2$ en base 2, alors cela veut dire que $n = 177$. On peut utiliser le tableau ci-dessous pour présenter le calcul.

Écriture en base 2	1	0	1	1	0	0	0	1	
Puissances de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Total
Termes à ajouter	128	0	32	16	0	0	0	1	177

On remarquera que dans cette convention d'écriture, le bit de poids fort (associé à 2^7) est le bit le plus à gauche dans l'écriture du nombre et que le bit de poids faible (aussi appelé bit de parité) est le bit le plus à droite dans l'écriture du nombre. On utilisera cette convention dans tous les exercices.

1 Déterminer l'écriture en base 2 des entiers compris entre 0 (inclus) et 17 (exclu).

2.1 Parité d'un nombre écrit en base 2

Écrire une fonction `est_pair` qui étant donné un tableau `bits` non vide dont les éléments appartiennent à $\{0;1\}$ détermine si le nombre dont l'écriture en base 2 est donnée par le tableau `bits` est un nombre pair. La fonction `est_pair` renverra `True` si c'est le cas, et `False` sinon.

Code python

```

1 def est_pair(bits):
2     """ [int] -> bool
3     len(bits) > 0
4     Détermine si le nombre dont l'écriture en base 2 est donnée par le
    ↪ tableau bits est pair. """
5     pass

```

Code python

```

1 print(est_pair([0]))
2 print(est_pair([1]))
3 print(est_pair([1, 1, 0, 0, 1]))

```

2.2 Nombre de bits d'un entier

Écrire une fonction encadre qui étant donné un nombre entier k détermine la valeur du plus grand entier que l'on puisse écrire en base 2 avec k bits.

Indication. Le plus grand entier que l'on puisse écrire en base 2 sur 4 bits est $\overline{1111}^2$, soit $1 + 2 + 4 + 8 = 15$.

À l'aide de la fonction encadre, écrire une fonction nombre_bits qui étant donné un entier n détermine le nombre k de bits présents dans l'écriture en base 2 du nombre n . On ne se souciera pas de l'efficacité de la fonction nombre_bits.

Indication. Le nombre de bits k recherché est le plus petit nombre tel que $n \leq \text{encadre}(k)$.

Code python

```
1 def encadre(k):
2     """ int -> int
3     Détermine le plus grand entier que l'on puisse écrire avec k bits. """
4     pass
5
6 def nombre_bits(n):
7     """ int -> int
8     Détermine le nombre de bits nécessaire à l'écriture de n en base 2.
9     ↪ """
10    pass
```

Code python

```
1 print(nombre_bits(1))
2 print(nombre_bits(15))
3 print(nombre_bits(16))
```

2.3 Entier maximal à nombre de bits fixés

Écrire une fonction est_plus_grand_kbits qui prend en entrée un tableau bits non vide de taille k dont les éléments appartiennent à $\{0; 1\}$ et qui détermine si le tableau correspond au plus grand nombre entier que l'on puisse écrire en base 2 sur k bits.

Code python

```
1 def est_plus_grand_kbits(bits):
2     """ Détermine si le tableau bits correspond au plus grand entier que
3     ↪ l'on puisse écrire sur k bits. """
4     pass
```

Code python

```
1 print(est_plus_grand_kbits([1, 0, 1]))
2 print(est_plus_grand_kbits([1, 1, 0]))
3 print(est_plus_grand_kbits([1, 1, 1]))
```

2.4 Base 2 vers base 10

Écrire une fonction base2_vers_base10 qui étant donné un tableau bits non vide dont les éléments appartiennent à $\{0; 1\}$ renvoie le nombre dont l'écriture en base 2 est donnée par le tableau bits.

Code python

```
1 def base2_vers_base10(bits):
2     """ [int] -> int
3     Détermine le nombre dont l'écriture en base 2 est donnée par bits. """
4     pass
```

Code python

```
1 print(base2_vers_base10([0]))
2 print(base2_vers_base10([1]))
3 print(base2_vers_base10([1, 0]))
4 print(base2_vers_base10([1, 0, 1, 0, 1, 0]))
```

3 Algorithmes classiques

3.1 Recherche d'occurrences

3.1.1 Appartient

Écrire une fonction appartient qui étant donné un tableau (éventuellement vide) tab d'entiers et entier e, détermine si l'élément e est présent dans le tableau tab.

Code python

```
1 def appartient(tab, e):
2     """ [int], int -> bool
3     Détermine si l'élément e est présent dans le tableau tab. """
4     pass
```

Code python

```
1 print(appartient([0, 1, 2, 3], 1))
2 print(appartient([0, 1, 2, 3], 4))
```

3.1.2 Nombre d'occurrences

Écrire une fonction nombre_occurrences qui étant donné un tableau (éventuellement vide) tab d'entiers et un entier e, détermine le nombre de fois où l'élément e est apparait dans le tableau tab (on appelle ce nombre le nombre d'occurrences de l'élément e).

Code python

```
1 def nombre_occurrences(tab, e):
2     """ [int], int, -> int
3     Détermine le nombre d'éléments de tab qui sont égaux à e. """
4     pass
```

Code python

```
1 print(nombre_occurrences([0, 1, 2, 3, 1, 2, 1], 0))
2 print(nombre_occurrences([0, 1, 2, 3, 1, 2, 1], 2))
3 print(nombre_occurrences([0, 1, 2, 3, 1, 2, 1], 1))
```

3.1.3 Indice des occurrences

Écrire une fonction indices_occurrences qui étant donné un tableau (éventuellement vide) tab d'entiers de type quelconque et un entier e, détermine la liste des indices (rangés par ordre croissant) des éléments de tab qui sont égaux à e.

Rappel. On pourra soit faire appel à la fonction nombre_occurrences pour initialiser le tableau d'indice des éléments égaux à e à la bonne taille, ou bien initialiser un tableau vide et ajouter les indices des éléments égaux à e au fur et à mesure à l'aide de l'instruction `t.append(elem)`.

Code python

```
1 def indices_occurrences(tab, e):
2     """ [int], int -> [int]
3     Détermine les indices des occurrences de e dans le tableau tab. """
4     pass
```

Code python

```
1 print(indices_occurrences([0, 1, 2, 3, 1, 2, 1], 0))
2 print(indices_occurrences([0, 1, 2, 3, 1, 2, 1], 2))
3 print(indices_occurrences([0, 1, 2, 3, 1, 2, 1], 1))
```

3.2 Recherche de maximum

Écrire une fonction `maximum` qui étant donné un tableau non vide `tab` d'entiers détermine la valeur du plus grand des éléments de ce tableau, ainsi que le premier indice pour lequel ce maximum est atteint.

Code python

```
1 def maximum(tab):
2     """ [int] -> int, int
3     len(tab) > 0
4     Détermine le maximum des éléments de tab, ainsi que le premier indice
   ↪ pour lequel ce maximum est atteint. """
5     pass
```

Code python

```
1 print(maximum([1]))
2 print(maximum([1, 2, -1]))
3 print(maximum([1, 2, -1, 2]))
```