Représentation des données

1 Les entiers

Décomposition en base b

Exercice 1 : Calculs en base 2

Réaliser les opérations suivantes en base 2, sans passer par la base 10, à l'aide des algorithmes appris à l'école primaire.

$$\underline{101010}_2 + \underline{11000}_2, \quad \underline{110101}_2 - \underline{11001}_2, \quad \underline{11101}_2 \times \underline{1011}_2, \quad \underline{1100101}_2 / \underline{1011}_2.$$

Exercice 2 : Somme et produit en base b

Dans cet exercice, un entier $d \in \mathbb{N}$ est représenté par sa décomposition en base $b \geqslant 2$, c'est-à-dire par une liste d'entiers $d_k \in [\![0, b[\![$ pour $0 \leqslant k < n$, telle que

$$d = \sum_{k=0}^{n-1} d_k b^k.$$

Les chiffres de poids faible sont situés en début de liste alors que les chiffres de poids fort sont quant à eux en fin de liste. Le but de cet exercice est d'implémenter l'addition et la multiplication en base b, comme on l'a appris à l'école primaire.

1. Écrire une fonction `chiffre(d: list[int], k: int) -> int` renvoyant le chiffre d_k du nombre d. Si k est plus grand que la longueur de la liste `d`, cette fonction devra renvoyer 0.
2. Écrire une fonction `addition(d: list[int], e: list[int], b: int) -> list[int]` réalisant l'addition de deux nombres d et e en base b.
3. (a) Écrire une fonction `multiplication_chiffre(d: list[int], c: int, i: int, b: int) -> list[int]` réalisant la multiplication en base b du nombre $d \in \mathbb{N}$ par cb^i, où $c \in [\![0, b[\![$ et $i \in \mathbb{N}$.
   (b) En déduire la fonction `multiplication(d: list[int], e: list[int], b: int) -> list[int]` réalisant la multiplication de deux nombres d et e en base b.

Exercice 3 : Incrément binaire

Écrire une fonction `increment(d: list[int]) -> NoneType` prenant en entrée la décomposition binaire

$$d = \sum_{k=0}^{n-1} d_k 2^k$$

du nombre d et la transformant en celle de $d + 1$.

Exercice 4 : Espace binaire

On appelle espace binaire d'un entier naturel n toute séquence consécutive de 0 délimités par deux 1 dans la décomposition en base 2 de n. Par exemple, le nombre 529 possède deux espaces binaires de longueurs respectives 3 et 4 car $529 = \underline{1000010001}_2$. En revanche, 32 ne possède pas d'espace binaire puisque $32 = \underline{100000}_2$.

Écrire une fonction `espace_binaire(n: int) -> int` qui prend pour argument un entier naturel et renvoie la longueur du plus grand espace binaire présent dans n s'il existe, et la valeur 0 sinon.

Exercice 5 : Factorion

On appelle factorion, tout entier naturel qui est égal à la somme des factorielles de ses chiffres. Par exemple, 145 est un factorion en écriture décimale, car

$$1! + 4! + 5! = 1 + 24 + 120 = 125.$$

1. Écrire une fonction `factorielle(n : int) -> int`, calculant la factorielle d'un entier n.
2. Écrire une fonction `factorion(m: int, b: int) -> bool`, déterminant si m est un factorion en base b.

3. En déduire une fonction `liste_factorions(b: int, p:int) -> list[int]`, renvoyant l'ensemble des factorions inférieurs ou égaux à p.
4. (a) Montrer que si $m \in \mathbb{N}$ est un factorion de n chiffres en base b, alors

$$b^{n-1} \leqslant m \leqslant n(b-1)!.$$

En particulier, si

$$u_n \coloneqq \frac{n(b-1)!}{b^{n-1}} < 1,$$

alors il n'existe aucun factorion de n chiffres.

(b) Montrer que la suite (u_n) est décroissante et tend vers 0, puis écrire une fonction

`les_factorions(b: int) -> list[int]`

renvoyant l'ensemble des factorions en base b.

Exercice 6 : Toblerone

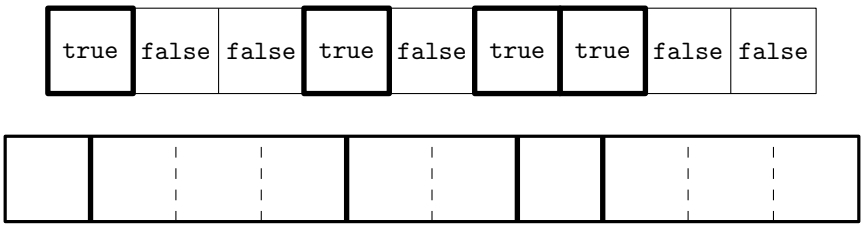
Après un changement de l'équipe dirigeante, l'entreprise Mondelez International, qui fabrique les barres Toblerone, décide de rationaliser sa production pour maximiser ses revenus. En effet, leur chaine de production fabrique des barres de n « carreaux » qui sont ensuite coupées en barres plus petites avant d'être vendues. Mais une récente étude de marché a déterminé le prix auquel on pouvait vendre des barres de longueur k (pour $0 \leqslant k \leqslant n$), et ce prix s'avère ne pas avoir de relation simple avec k. Le problème est donc de décider comment découper la barre initiale de n carreaux en des barres plus petites pour maximiser le prix de vente total.

Dans tout le problème, on considèrera que l'on dispose d'un tableau `p`, indicé de 0 à n, tel que `p[k]` est le prix de vente d'une barre de longueur k. Bien entendu, le prix d'un morceau de taille 0 est 0. Remarquez que si `p[n]` est suffisamment grand, la solution optimale peut très bien être de ne pas découper la barre.

On donne un exemple de tableau p pour $n = 10$.

k	0	1	2	3	4	5	6	7	8	9	10
p_k	0	1	5	8	9	10	17	17	20	24	26

Pour résoudre ce problème, on commence par établir une correspondance entre les découpes d'un Toblerone composé de n carreaux et les tableaux d de $n-1$ booléens : on coupe après le carreau d'indice k si et seulement si d_k est vrai.



La découpe $1, 3, 2, 1, 3$ et le tableau de booléens correspondant.

1. Écrire une fonction `decomposition(d: int, n: int) -> list[bool]` prenant en entrée un entier $n \in \mathbb{N}$ ainsi qu'un entier $d \in [\![0, 2^n[\![$, et renvoyant une liste de booléens d_k de longueur n telle que

$$d = \sum_{k=0}^{n-1} d_k 2^k$$

où le booléen `True` représente le bit 1 et le booléen `False` représente le bit 0.

2. Écrire une fonction `prix_decoupe(d: list[bool], p: list[int]) -> int` prenant en entrée un tableau d de longueur $n-1$ représentant une découpe d'une barre de longueur n, ainsi qu'un tableau p de longueur $n+1$ représentant la liste des prix des différentes longueurs et renvoyant le prix de revente de la découpe d.
3. En déduire une fonction `meilleur_prix(p: list[int]) -> tuple[list[bool], int]` renvoyant une meilleure découpe d'une barre de longueur n ainsi que son prix de revente correspondant.
4. Donner le prix ainsi qu'une découpe optimale associée pour l'exemple de tableau p donné plus haut.

Nous verrons dans l'année des algorithmes dit de « programmation dynamique » permettant de résoudre ce type de problème plus efficacement.

Représentation mémoire des entiers non signés

Représentation mémoire des entiers signés

Exercice 7 : Complément à 2

Dans une représentation en complément à 2 sur 8 bits, quels sont les entiers relatifs représentés par 01101101 et 10010010 ?

Exercice 8 : Machine 16 bits

On considère une architecture 16 bits. On considère les opérations suivantes entre les entiers signés. Donner leur résultat mathématique (en les considérant comme entiers naturels) puis leur résultat sur 16 bits signés.

1. 10×10
2. $32767 + 1$
3. $256 \times (-256)$
4. $32767 - (-32768)$

2 Les nombres flottants

Représentation mémoire des flottants

Exercice 9 : Le type float16

Dans le type `float16` utilisé par Numpy, les nombres flottants sont représentés sur 16 bits : 1 bit pour le signe, 5 bits pour l'exposant, 10 bits pour la mantisse.

1. Donner la représentation machine dans de type de 1, de -2, puis du plus petit nombre strictement supérieur à 1, ainsi que se valeur.
2. Donner les représentations machine et la valeur des plus petits et des plus grands nombres normalisés.
3. Déterminer quel nombre est représenté par 0|01110|1001001000.

Problèmes liés à l'arithmétique des nombres flottants

Exercice 10 : Hamster jovial

À votre grand bonheur, vous avez reçu pour Noël une balance d'excellente qualité : elle offre trois chiffres décimaux de précision, et ce autant pour des masses de l'ordre du gramme que de l'ordre de la tonne. Vous décidez d'utiliser cette balance pour mesurer la masse m_h de votre hamster h. On suppose pour simplifier que m_h est de l'ordre de 100 grammes (un gros hamster, d'après Wikipedia).

1. Si h accepte de monter docilement sur la balance et d'y rester le temps qu'elle fasse sa mesure, avec quelle précision obtiendrez-vous m_h ?
2. h, qui est d'une intelligence assez rare pour un rongeur, vous soupçonne, à tort ou à raison, de vouloir utiliser cette pesée pour justifier une mise au régime. Il descend donc immédiatement de la balance à chaque fois que vous l'y posez, et ce avant que la mesure n'ait été faite. Vous décidez alors de le peser indirectement : vous vous pesez une première fois avec h dans la main, puis une deuxième fois sans h, et vous faites la différence. Avec quelle précision obtenez-vous m_h ?

Exercice 11 : Ordre de sommation

On définit la suite (u_n) par

$$\forall n \in \mathbb{N}^*, \quad u_n := \sum_{k=2}^{n} \frac{1}{k(k-1)}.$$

1. En remarquant que
$$\forall k \geqslant 2, \quad \frac{1}{k(k-1)} = \frac{1}{k-1} - \frac{1}{k},$$
calculer explicitement u_n.
2. Afin de calculer u_n à l'aide d'une somme, on écrit :

```
def sum_1(n):
    s = 0.0
    for k in range(2, n + 1):
        s = s + 1 / (k * (k - 1))
    return s
```

Écrire le programme `sum_2` calculant la même somme, mais sommant les $1/(k(k-1))$ non pas avec k allant de 2 à n de manière croissante,d mais allant de n à 2 de manière décroissante.

3. Comparer le résultat des deux fonctions précédentes pour $n = 10\,000\,000$. Quelle est la fonction la plus précise ? Comment expliquez-vous ce phénomène ?

3 Caractères et chaines de caractères

Codes Ascii et Unicode

Lecture et écriture dans un fichier