

Consignes :

- * Les calculatrices sont interdites. Numérotez vos feuilles et faites apparaître les questions dans l'ordre de l'énoncé.
- * Si vous repérez une erreur d'énoncé, signalez le sur votre copie et poursuivez votre composition.
- * Attention de ne pas confondre les notions d'« entier » et de « chiffre » :
 - Un entier, aussi appelé un « nombre », est un élément de \mathbb{N} .
 - Un chiffre est un élément de $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

1. Introduction

Dans tout le sujet, on s'intéresse à une fonction de \mathbb{N} dans \mathbb{N} notée RATS dont le nom vient de l'anglais :

RReverse, **A**dd to the original, **T**hen **S**ort the digits.

Notation 1. Soient

$$R : \mathbb{N} \rightarrow \mathbb{N}, \quad S : \mathbb{N} \rightarrow \mathbb{N}, \quad RATS : \mathbb{N} \rightarrow \mathbb{N},$$

les fonctions définies pour tout $n \in \mathbb{N}$ par :

- $R(n)$ est l'entier contenant les mêmes chiffres que n , mais dans l'ordre inverse.
- $S(n)$ est l'entier contenant les mêmes chiffres que n , mais dans l'ordre croissant.
- $RATS(n) = S(n + R(n))$.

Exemple 2. On a :

$$\begin{aligned} R(577862) &= 268775, & R(174460) &= 64471, \\ S(6985537) &= 3556789, & S(10123006) &= 11236. \end{aligned}$$

et :

$$\begin{aligned} RATS(648724) &= S(648724 + R(648724)) \\ &= S(648724 + 427846) \\ &= S(1076570) \\ &= 15677 \end{aligned}$$

1. Calculer $RATS(751)$ et $RATS(6876)$.

Terminons cette introduction avec un programme qui teste si deux listes sont identiques. Cette fonction pourra être réutilisée librement dans la suite.

2. Écrire une fonction `egal(L1: list[int], L2: list[int]) -> bool` qui indique si L1 et L2 sont égales. On attend ici que vous utilisiez une boucle `for` ou `while`. En particulier, il est interdit de comparer directement les listes avec `==` ou `!=`.

2. Fonction RATS en Python

2.a. Fonction R

Le but de cette partie est d'implémenter la fonction R en Python. Pour calculer $R(n)$, nous allons convertir l'entier n en chaîne de caractères, inverser l'ordre de ses chiffres, puis convertir la chaîne de caractères obtenue en entier.

3. Soit s une chaîne de caractères et t la chaîne de caractères contenant les mêmes caractères que s , mais dans l'ordre inverse. Par exemple :

Si $s = \text{"Ceci est un test"}$ alors $t = \text{"tset nu tse iceC"}$

- (a) Rappeler la syntaxe Python permettant de définir t à partir de s . La réponse attendue est un programme d'une seule ligne.
- (b) Dans cette question, on s'interdit la syntaxe donnée dans la réponse précédente. À l'aide de concaténations et d'une boucle `for`, écrire une fonction `revStr(s: str) -> str` qui prend en entrée s et renvoie t . Par exemple :

`revStr("Ceci est un test")` vaut `"tset nu tse iceC"`

4. À l'aide de la fonction `revStr`, écrire une fonction `revInt(n: int) -> int` qui renvoie $R(n)$. Par exemple :

`revInt(427846)` vaut `648724`

2.b. Tri par comptage

Vocabulaire 3. Soit « $C: \text{list}[\text{int}]$ » une liste d'entiers. On dit que C est une *liste de chiffres* si tous ses éléments appartiennent à $\llbracket 0; 9 \rrbracket$.

5. (a) Écrire une fonction `estLC(C: list[int]) -> bool` qui renvoie `True` si C est une liste de chiffres et `False` sinon.
- (b) Donner en la justifiant la complexité de la fonction `estLC`.

Définition 4. Soit C une liste de chiffres. On appelle *dictionnaire des chiffres de C* le dictionnaire « $d: \text{dict}[\text{int}: \text{int}]$ » tel que :

- Les clés de d sont tous les entiers $i \in \llbracket 0; 9 \rrbracket$ présents dans C .
- Pour chaque clé i de d , la valeur associée à i dans d est le nombre d'occurrences de i dans C .

Exemple 5. Le dictionnaire des chiffres de $[1, 6, 9, 0, 0, 8, 6, 7, 9, 6, 6, 9, 8]$ est :

$\{1:1, 6:4, 9:3, 0:2, 8:2, 7:1\}$.

6. (a) Écrire une fonction `dictC(C: list[int]) -> dict[int:int]` qui renvoie le dictionnaire des chiffres de C . Si C n'est pas une liste de chiffres, votre fonction déclenchera une erreur.
- (b) Donner en la justifiant la complexité de la fonction `dictC`.

Soit « $C: \text{list}[\text{int}]$ » une liste de chiffres. Notre objectif est de construire « $T: \text{list}[\text{int}]$ » la liste triée contenant tous les entiers non nuls de C . Par exemple :

Si $C = [1, 6, 9, 0, 0, 8, 6, 7, 9, 6, 6, 9, 8]$ alors $T = [1, 6, 6, 6, 6, 7, 8, 8, 9, 9, 9]$

Pour cela, on utilise l'algorithme du *tri par comptage* : le principe est de calculer $d = \text{dictC}(C)$, puis d'en déduire T (c'est à vous de déterminer comment construire T à partir de d).

7. Écrire une fonction `triCPT(C: list[int]) -> list[int]` qui applique l'algorithme du tri par comptage sur C . Votre fonction doit renvoyer la liste T et être la plus efficace possible. On pourra supposer sans le vérifier que C est une liste de chiffres.

2.c. Fonction S

On souhaite maintenant implémenter la fonction S décrite dans l'introduction. Dans toute cette partie, on s'interdit les conversions entre chaînes de caractères et entiers.

8. Écrire une fonction `int_to_LC(n: int) -> list[int]` qui renvoie la liste contenant les chiffres de n . On pourra supposer sans le vérifier que $n \in \mathbb{N}$.

Rappel : on s'interdit les conversions entre chaînes de caractères et entiers. Par exemple :

n	0	4	99	48546
int_to_LC(n)	[0]	[4]	[9, 9]	[4, 8, 5, 4, 6]

9. (a) Écrire une fonction `estTrieec(L: list[int]) -> bool` qui renvoie `True` si L est croissante, et `False` sinon. Ici, le terme "croissante" est à prendre au sens large : si e_1 et e_2 sont deux éléments consécutifs de L alors on doit avoir $e_1 \leq e_2$.

- (b) Écrire une fonction `LC_to_int(C: list[int]) -> int` telle que :

→ Si `estTrieec(C)` vaut `False`, alors `LC_to_int` déclenche une erreur.

→ Si $C = []$, alors `LC_to_int` renvoie 0.

→ Sinon, `LC_to_int` renvoie l'entier dont les chiffres sont les éléments de C .

On pourra supposer sans le vérifier que tous les éléments de C appartiennent à $\llbracket 1; 9 \rrbracket$. Par exemple :

C	[]	[4]	[9, 9]	[4, 8, 5, 4, 6]	[4, 4, 5, 6, 8]
LC_to_int(C)	0	4	99	Erreur	44568

10. Écrire une fonction `triInt(n: int) -> int` qui renvoie $S(n)$. Par exemple :

`triInt(2601422370)` vaut 12223467

2.d. Fonction RATS

11. En utilisant les fonctions des parties précédentes, programmer la fonction `RATS(n: int) -> int`. Si $n < 0$, votre fonction déclenche une erreur.

3. La suite RATS

3.a. Définitions

Notation 6. Dans cette partie, $k_0 \in \mathbb{N}$ est un entier et $(u_n)_{n \in \mathbb{N}}$ est la suite définie par :

$$\begin{cases} u_0 = k_0 \\ \forall n \in \mathbb{N} : u_{n+1} = \text{RATS}(u_n) \end{cases}$$

12. (a) Lorsque $k_0 = 1$, calculer $u_0, u_1, u_2, u_3, u_4, u_5$.
 (b) Lorsque $k_0 = 44466666$, calculer u_n pour tout $n \in \mathbb{N}$.

Définition 7. Soit $T \in \mathbb{N}^*$. On dit que la suite $(u_n)_{n \in \mathbb{N}}$ est *ultimement périodique de période T* si :

$$\exists n_0 \in \mathbb{N}, \forall n \geq n_0 : u_{n+T} = u_n.$$

Remarque 8. La terminologie "ultimement périodique" est synonyme de "périodique à partir d'un certain rang".

13. Soit $T \in \mathbb{N}^*$. Supposons qu'il existe un entier $m \in \mathbb{N}$ tel que $u_m = u_{m+T}$. Montrer que $(u_n)_{n \in \mathbb{N}}$ est ultimement périodique de période T .

Dans la suite, on admet que pour tout $k_0 \in \mathbb{N}$, la suite $(u_n)_{n \in \mathbb{N}}$ vérifie un et un seul de ces deux cas :

(cas 1) La suite $(u_n)_{n \in \mathbb{N}}$ tend vers $+\infty$.

(cas 2) Il existe $T \in \mathbb{N}^*$ tel que $(u_n)_{n \in \mathbb{N}}$ est ultimement périodique de période T .

Exemple 9. Si $k_0 = 3$, alors $u_6 = u_{14} = 444$ (voir le tableau ci-dessous). Ainsi, $u_m = u_{m+T}$ avec $m = 6$ et $T = 8$. En vertu de la question 13, la suite $(u_n)_{n \in \mathbb{N}}$ est ultimement périodique de période 8.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
u_n	3	6	12	33	66	123	444	888	1677	3489	12333	44556	111	222	444	888

Définition 10. Soit $k_0 \in \mathbb{N}$ et $(u_n)_{n \in \mathbb{N}}$ la suite associée. On appelle **période RATS de k_0** et on note $P(k_0) \in \mathbb{N}$ l'entier défini par :

(cas 1) Si $(u_n)_{n \in \mathbb{N}}$ tend vers $+\infty$, alors $P(k_0) = 0$.

(cas 2) Sinon, $P(k_0)$ est le plus petit $T \in \mathbb{N}^*$ tel que $(u_n)_{n \in \mathbb{N}}$ est ultimement périodique de période T .

Exemple 11. En reprenant les exemples précédents :

$$P(3) = 8 \quad \text{et} \quad P(4446666) = 3$$

14. Calculer $P(27)$.

3.b. Calcul de la période RATS - Version 1

Soit $k_0 \in \mathbb{N}$ un entier. Pour déterminer $P(k_0)$, on utilise la procédure suivante :

→ On part d'un dictionnaire vide \mathbf{d} dans lequel on va ajouter successivement les clés u_0, u_1, u_2, \dots

→ À chaque étape, avant d'ajouter u_n dans \mathbf{d} :

- Si $n = 1000$, la procédure s'arrête et renvoie 0. C'est à dire qu'on considère que $P(k_0) = 0$.
- Sinon, si u_n est déjà une clé de \mathbf{d} , la procédure s'arrête et calcule $P(k_0)$ à l'aide de la question 13.
- Sinon, on ajoute u_n dans \mathbf{d} en lui associant une valeur bien choisie (c'est à vous de déterminer la valeur adéquate).

15. À l'aide de la procédure décrite ci-dessus, écrire une fonction `periodeV1` qui prend en entrée $k_0 \in \mathbb{N}$ et renvoie $P(k_0)$.

16. Écrire une fonction `toutesPeriodes(k0_max: int) -> list[int]` qui renvoie la liste contenant les $P(k_0)$ pour k_0 compris entre 0 et k_0_max .

Conjecture 12. Il est conjecturé (mais ça n'a pas été prouvé) que pour tout $k_0 \in \mathbb{N}$:

$$P(k_0) \in A \quad \text{avec} \quad A = \{0, 1, 2, 3, 8, 14, 18\}$$

17. Écrire une fonction `minPeriode(p: int) -> int` qui renvoie :

$$\min \{ k_0 \in \mathbb{N} : \text{periodeV1}(k_0) = p \}$$

Si p n'appartient pas à l'ensemble A , votre fonction déclenchera une erreur.

3.c. Calcul de la période RATS - Version 2

La procédure décrite au début de la partie précédente suppose que si les 1000 premiers termes de $(u_n)_{n \in \mathbb{N}}$ sont différents, alors $P(k_0) = 0$. Cette hypothèse étant fautive, notre objectif est de trouver un autre critère d'arrêt pour la fonction `periode`.

Notation 13. Pour tout $m \in \mathbb{N}$ vérifiant $m \geq 2$, soient $a_m \in \mathbb{N}$ et $b_m \in \mathbb{N}$ les entiers :

$$a_m = \underbrace{123 \dots 3}_{m \text{ fois}} 4444 \quad \text{et} \quad b_m = 55 \underbrace{6 \dots 6}_{m \text{ fois}} 7777$$

18. Soit $k_0 \in \mathbb{N}$ et $(u_n)_{n \in \mathbb{N}}$ la suite associée. On suppose qu'il existe $n \in \mathbb{N}$ et $m \geq 2$ tels que $u_n = a_m$ ou $u_n = b_m$. Montrer que $P(k_0) = 0$.

Conjecture 14. Il est conjecturé (mais ça n'a pas été prouvé) que l'implication réciproque de la question précédente est vraie : si $P(k_0) = 0$, alors il existe $n \in \mathbb{N}$ et $m \geq 2$ tels que $u_n = a_m$ ou $u_n = b_m$.

19. Dans cette question, on suppose que la conjecture 14 est vraie. Écrire une fonction `periodeV2` qui prend en entrée $k_0 \in \mathbb{N}$ et renvoie $P(k_0)$. Bien sûr, vous devez exploiter la conjecture 14, et ne pas utiliser la fonction `periodeV1`.