DS2

Exercice I : Analyse d'algorithmes

Étant donné deux entiers naturels a et b, on cherche à calculer leur produit p.

Partie 1 : Méthode élémentaire

Cette première méthode consiste à décomposer le produit ab comme la somme $b+b+\cdots+b$ (avec a termes). Elle n'utilise que des additions.

```
1  def enfant(a, b) :
2      p, q = 0, a
3      while q > 0 :
4          p = p + b
5          q = q - 1
6      return p
```

1. Indiquer la spécification de l'algorithme.

2. Donner, en justifiant, un *variant* qui prouve la terminaison de cet algorithme.

3. Choisir un *invariant de boucle* judicieux parmi les propositions suivantes :

a) $(I_1) : p = ab$

b) $(I_2) : (p+b)q = ab$

c) $(I_3) : p+qb = ab$

d) $(I_4) : (p+q)b = ab$

4. Prouver la correction de cet algorithme.

Partie 2 : Méthode du paysan russe

La méthode du paysan russe est un très vieil algorithme déjà décrit (sous une forme légèrement différente) sur un papyrus égyptien rédigé vers 1650 av. J.-C. En comparaison de l'algorithme décimal classique, elle présente l'intérêt de n'utiliser que la table de multiplication par 2.

```
1  def paysan(a, b) :
2      p, x, y = 0, a, b
3      while x > 0 :
4          if x % 2 == 1 :
5              p = p + y
6          x = x // 2
7          y = y * 2
8      return p
```

5. Calculer 18×13 à l'aide de cet algorithme. On donnera dans un tableau les valeurs successives des variables p, x, y.

6. Donner, en justifiant, un *variant* qui prouve la terminaison de cet algorithme.

7. Vérifier que $ab = p + xy$ est un *invariant de boucle* et en déduire la correction.

8. On note $T(a)$ le nombre d'itérations de la boucle `while`.

a) Que vaut $T(0)$? Si $a \geqslant 1$, justifier : $T(a) = 1 + T(\lfloor \frac{a}{2} \rfloor)$.

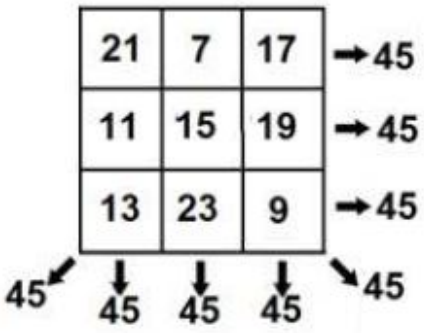
b) Calculer, pour tout $k \in \mathbb{N}$, la valeur de $T(2^k)$.

c) Montrer finalement : $T(a) = O\big(\ln(a)\big)$.

9. Comparer l'efficacité des deux algorithmes.

Exercice II : Programmation et complexité

Soit $n \in \mathbb{N}^*$. Un carré magique d'ordre n est une matrice carrée d'ordre n qui contient des nombres entiers strictement positifs. Ces nombres sont disposés de sorte que les sommes sur chaque ligne, les sommes sur chaque colonne et les sommes sur chaque diagonale soient égales. La valeur de ces sommes est appelée *constante magique*.



Carré magique d'ordre 3 et de constante magique 45.

Pour représenter une matrice carrée d'ordre n, on utilisera une liste qui contient n listes toutes de même longueur n. Pour l'exemple précédent, on a donc :

- `M=[[21,7,17],[11,15,19],[13,23,9]]`,
- `M[1]=[11,15,19]`,
- `M[1][0]=11`.

Dans tout le sujet, on considèrera que les matrices carrées sont représentées par de telles listes et on ne vérifiera ni que la matrice est bien carrée, ni que les coefficients sont bien des entiers strictement positifs.

Partie A : Sommes d'éléments sur une matrice carrée

10. Écrire une fonction `somme_ligne` qui prend en paramètre une matrice carrée `M` sous forme de liste de listes et un entier `i`, et qui renvoie la somme des termes de la ligne d'indice `i` de la matrice `M`.

Par exemple, `somme_ligne([[1,2,3],[4,5,6],[7,8,9]], 1)`, renvoie : $4+5+6=15$.

11. Écrire une fonction `somme_colonne` qui prend en paramètre une matrice `M` sous forme de liste de listes et un entier `j`, et qui renvoie la somme des termes de la colonne d'indice `j` de `M`.

Par exemple, `somme_colonne([[1,2,3],[4,5,6],[7,8,9]], 0)` renvoie : $1+4+7=12$.

12. Écrire une fonction `somme_diag1` qui prend en paramètre une matrice `M` sous forme de liste de listes, et qui renvoie la somme des termes de la diagonale de `M`.

Par exemple, `somme_diag1([[1,2,3],[4,5,6],[7,8,9]])` renvoie : $1+5+9=15$.

13. Écrire une fonction `somme_diag2` qui prend en paramètre une matrice `M` sous forme de liste de listes, et qui renvoie la somme des termes de l'antidiagonale de `M` (celle qui va du coin en haut à droite jusqu'au coin en bas à gauche).

Par exemple, `somme_diag2([[1,2,3],[4,5,6],[7,8,9]])` renvoie : $3+5+7=15$.

14. Déterminer la complexité de ces quatre dernières fonctions en fonction de n.

Partie B : Carré magique

15. Écrire une fonction `carre_magique` qui prend en paramètre une matrice `M` sous forme de liste de listes, et qui renvoie `True` si `M` est un carré magique et `False` sinon.

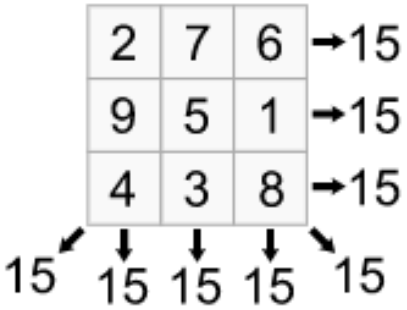
Par exemple :

- l'appel `carre_magique([[1,2,3],[4,5,6],[7,8,9]])` renvoie : `False`,
- l'appel `carre_magique([[21,7,17],[11,15,19],[13,23,9]])` renvoie : `True`.

16. Déterminer la complexité de la fonction `carre_magique` dans le pire cas en fonction de n.

Partie C : Carré magique normal

Un carré magique *normal* d'ordre n est un carré magique d'ordre n constitué de tous les entiers positifs compris entre 1 et n^2.



Carré magique normal d'ordre 3.

17. Écrire une fonction `magique_normal` qui prend comme argument une matrice carrée `M` sous forme de liste de listes et qui renvoie `True` si `M` est un carré magique normal et `False` sinon.

Par exemple :

- si `M = [[21,7,17],[11,15,19],[13,23,9]]`, on a : `magique_normal(M) = False`,
- si `M = [[2,7,6],[9,5,1],[4,3,8]]`, on a : `magique_normal(M) = True`.

18. Que vaut la constante magique d'un carré magique normal d'ordre n ?

Partie D : Construction d'un carré magique normal d'ordre impair

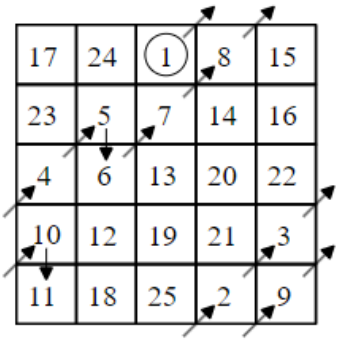
La méthode siamoise est une méthode qui permet de construire un carré magique normal d'ordre n impair. Le principe de cette méthode est le suivant :

i. Créer une matrice carrée d'ordre n, remplie de 0.

ii. Placer le nombre 1 au milieu de la ligne d'indice 0.

iii. Se décaler en diagonale d'une case vers la droite et une case vers le haut pour placer le nombre 2, puis faire de même pour le nombre 3, puis le nombre 4, ... jusqu'à n^2.
Le déplacement doit respecter les deux règles suivantes :

- × si la pointe de la flèche sort du carré, revenir de l'autre côté, comme si le carré était enroulé sur un tore (voir la figure suivante).
- × si la prochaine case contient un entier non nul, se déplacer d'une case vers le bas.



Création d'un carré magique normal d'ordre 5.

19. Écrire une fonction `matrice_nulle` qui prend en paramètre un entier `n`, et qui renvoie la matrice carrée d'ordre n dont tous les coefficients sont nuls, représentée sous forme de liste de listes.

20. Écrire une fonction `decalage` qui prend en paramètres trois entiers `n`, `i` et `j`, et qui renvoie la prochaine position `(pi, pj)` qui succède à `(i, j)` après décalage vers la droite et le haut.

21. Écrire une fonction `siamoise`, étant donné un entier `n` impair strictement positif en paramètre, renvoie un carré magique normal d'ordre n en utilisant la méthode siamoise.