ACADÉMIE
DE NORMANDIE

Liberté
Égalité
Fraternité

Olympiades d'informatique 2022

- Classes de première -

Normandie

Durée de l'épreuve : 3 heures

L'épreuve se déroule par équipe et sur ordinateur.

Le sujet est constitué de deux parties indépendantes qui peuvent être traitées dans l'ordre souhaité.

Un dossier est mis à disposition des équipes avec les fichiers requis pour certaines questions.

Les fichiers créés sont à joindre dans un unique dossier compressé qui seront nommés de la façon suivante :

NomEtablissement_VilleEtablissement_EquipeXX_Signature.py

NomEtablissement_VilleEtablissement_EquipeXX_Automates.py

où le numéro de l'équipe vous sera attribué par votre professeur.

À l'intérieur de chacun de ces fichiers, devront figurer en commentaire les noms de chaque membre de l'équipe.

Les calculatrices sont autorisées selon la réglementation en vigueur.

Société Informatique de France

UFR SCIENCES & TECHNIQUES

GREYC
Laboratoire de recherche en sciences du numérique

Signature numérique d'un tableau

Présentation

Le musée Mauritshuis de La Haye aux Pays-Bas dispose d'une collection de centaines de tableaux néerlandais, en particulier des oeuvres de Rembrandt et Vermeer. Vous devez réaliser le projet suivant pour ce musée : à partir de la numérisation d'un tableau et de la signature numérique de l'artiste, produire un nouveau fichier intégrant la signature, invisible à l'oeil nu.

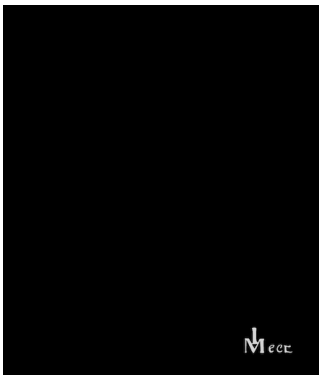
Afin de prouver le bon fonctionnement de votre code, vous devrez de plus, à partir de ce dernier fichier et du tableau original, reconstruire la signature numérique...

Mise en oeuvre

Fichiers fournis

Le musée a mis à disposition deux fichiers : `vermeer1.jpg` représentant le célèbre tableau "La jeune fille à la perle" du peintre Vermeer, ainsi que le fichier `signature1.png`, fichier image en niveaux de gris, aux mêmes dimensions, contenant uniquement des pixels noirs sauf la signature de l'artiste :

La jeune fille à la perle



Signature de Vermeer

Rappels sur les pixels et la bibliothèque PIL

On rappelle que la couleur d'un pixel est représentée par trois entiers variant entre 0 et 255, qui constituent la quantité de rouge, de vert et de bleu. Une image est en niveaux de gris lorsque n'importe lequel de ses pixels a la même quantité de rouge que de vert et de bleu. On rappelle également que la couleur noire correspond aux valeurs (0, 0, 0).

Pour la réalisation de votre code, vous allez utiliser la bibliothèque PIL, dont voici un usage possible (le fichier Python correspondant devant se trouver dans le même dossier que les deux fichiers `vermeer1.jpg` et `signature1.png`) :

```
from PIL import Image

img1 = Image.open("vermeer1.jpg")
img2 = Image.open("signature1.png")
img3 = Image.new("RGB", img1.size)

largeur, hauteur = img1.size


img3.save("vermeer1_avec_signature.png")
```

La variable `img3` représente une image, pour l'instant composée uniquement de pixels noirs, aux mêmes dimensions que les deux images fournies.

Votre code utilisera `getpixel` et `putpixel` de la bibliothèque PIL. Par exemple :

```
(rouge, vert, bleu) = img1.getpixel((0,0))
img3.putpixel((0,0), (rouge, vert, bleu))
```

Ce code permet de récupérer les quantités de rouge, vert, bleu du pixel de coordonnées (0,0) de la première image et de colorer le pixel correspondant de la variable `img3`.

Enfin, l'instruction `img3.show()` vous permet à tout moment d'afficher l'image correspondante à la variable `img3`.

Principe général

Pour dissimuler la signature, nous allons créer une nouvelle image à partir de l'image de départ et de l'image de la signature. Dans cette nouvelle image, chaque pixel de coordonnées (i, j) sera :

- identique à celui de coordonnées (i, j) dans l'image de départ si le pixel (i, j) dans l'image de la signature est noir ;
- légèrement différent de celui de l'image de départ sinon, de façon à pouvoir intégrer également les informations du pixel de la signature.

Pour cela voici le principe choisi sur un exemple d'un pixel où la couleur de la signature n'est pas noire, par exemple (124, 124, 124), avec une couleur du pixel correspondant du tableau à (74, 107, 83). Comme la signature est en niveaux de gris, pour chaque pixel, il n'y a donc qu'un entier à masquer, ici 124. Son écriture binaire sur un octet est `0111 1100`. Nous allons répartir chacun de ces bits dans les trois composantes du pixel de l'image originale.

La couleur (74, 107, 83) s'écrit en binaire (`0100 1010`, `0110 1011`, `0101 0011`), nous gardons uniquement certains bits de poids forts (5 pour le rouge et le vert, 6 pour le bleu, pour "laisser" 8 emplacements libres) :

(`0100 1...`, `0110 1...`, `0101 00..`)

Nous remplaçons ces bits par ceux de `0111 1100` séparés en trois groupes tels que nous les lisons dans l'ordre : `011`, `111` et `00`.

Ainsi nous obtenons

$$(0100\ 1011, 0110\ 1111, 0101\ 0000)$$

soit la couleur (75, 111, 80), très proche de la couleur d'origine.

Détails de la réalisation

Votre code devra contenir les fonctions suivantes :

- Une fonction `dec_vers_bin` prenant en paramètre un entier compris entre 0 et 255 et renvoyant son écriture binaire sous la forme d'une liste de 8 bits.
  Exemple :

```
>>> dec_vers_bin(80)
[0, 1, 0, 1, 0, 0, 0, 0]
```

- Une fonction `bin_vers_dec` prenant en paramètre une liste de 8 bits et renvoyant l'écriture décimale correspondante.
  Exemple :

```
>>> bin_vers_dec([0, 1, 0, 1, 0, 0, 0, 0])
80
```

- Une fonction `combinaison1` prenant en paramètres deux nombres compris entre 0 et 255 et renvoyant un troisième nombre entier. Ce dernier aura une écriture binaire constituée des 5 bits de poids fort du premier entier et des 3 bits de poids forts du second entier.
  Exemple :

```
>>> combinaison1(74, 124)
75
```

- Une fonction `combinaison2` prenant en paramètres deux nombres compris entre 0 et 255 et renvoyant un troisième nombre entier. Ce dernier aura une écriture binaire constituée des 5 bits de poids fort du premier entier et des 3 bits suivants du second entier.
  Exemple :

```
>>> combinaison2(107, 124)
111
```

- Une fonction `combinaison3` prenant en paramètres deux nombres compris entre 0 et 255 et renvoyant un troisième nombre entier. Ce dernier aura une écriture binaire constituée des 6 bits de poids fort du premier entier et des 2 bits de poids faible du second entier.
  Exemple :

```
>>> combinaison3(83, 124)
80
```

- Une fonction permettant de compléter la variable `img3` avant de sauvegarder l'image correspondante sous le nom de `vermeer1_avec_signature.png`
- Une fonction permettant, à partir des fichiers `vermeer2_avec_signature.png` et `vermeer2.jpg` fournis, de reconstituer et d'afficher l'image d'une autre signature de Vermeer, qui a été intégrée au fichier `vermeer2_avec_signature.png` avec la méthode précédente.

Automates

Présentation générale

Alphabet et Langage

En informatique, on utilise de nombreux *langages*. Parmi ceux-ci, on distingue une famille particulière, celle des *langages rationnels*.

Un langage rationnel est notamment défini par un ensemble de *mots* constitués avec les lettres d'un *alphabet*. On peut par exemple définir l'alphabet constitué des 3 lettres a, b et c :

$$\mathcal{A} = \{a, b, c\}$$

.

Il est ensuite possible de définir un langage sur cet alphabet. Voici par exemple le langage constitué des 3 mots a, $baba$ et bac :

$$\mathcal{L}_1 = \{a, baba, bac\}$$

Un langage n'est pas forcément un ensemble fini. On peut ainsi définir sur l'alphabet $\mathcal{A}$ le langage suivant :

$$\mathcal{L}_2 = \left\{ \begin{array}{l} \text{mots constitués de lettres de A contenant au moins un } a \\ \text{et finissant par un } b \end{array} \right\}$$

On trouvera par exemple dans $\mathcal{L}_2$ des mots comme ab, aab, abb, acb, $aaab$, ou encore $cabab$.

Automate

Pour savoir si un mot appartient à un langage, on peut utiliser des *automates*. Un automate associé à un langage peut être représenté ainsi :

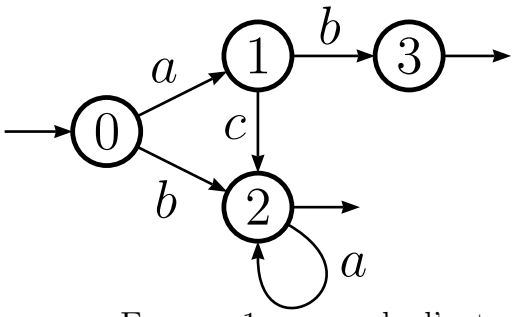


FIGURE 1 – exemple d'automate

Les cercles contenant des chiffres sont appelés des *états*, les flèches surmontées d'une lettre sont appelées des *transitions*.

Sur un automate, on distingue un *état initial* unique, cible d'une flèche ne partant d'aucun autre état – l'état 0 sur la figure 1 – et des états finaux, d'où part une flèche ne ciblant aucun autre état – les états 2 et 3 sur la figure 1.

Pour déterminer si un mot appartient à un langage dont on connaît l'automate, on peut voir le mot comme la suite des embranchements à prendre dans l'automate en partant de l'état initial, chaque lettre indiquant un embranchement. Si à chaque étape, il existe un embranchement correspondant à la lettre courante, et si après avoir parcouru toutes les lettres du mot, on se retrouve dans un état final, alors le mot est un mot du langage associé à l'automate. Sinon, le mot n'appartient pas au langage.

Présentation de quelques exemples

Exemple 1. Le mot `acaa` appartient-il au langage associé à l'automate de la figure 1 ?

état courant	lettres restant à lire	lettre courante	état suivant
0	acaa	a	1
1	caa	c	2
2	aa	a	2
2	a	a	2
2	$\emptyset$	fin sur un état final	

On a parcouru toutes les lettres du mot et on est arrivé sur un état final ; le mot *acaa* appartient bien au langage associé à l'automate.

Exemple 2. Le mot `a` appartient-il au langage associé à l'automate de la figure 1 ?

état courant	lettres restant à lire	lettre courante	état suivant
0	a	a	1
1	$\emptyset$	fin sur un état non final	

On a parcouru toutes les lettres du mot et on est arrivé sur un état non final ; le mot *a* n'appartient donc pas au langage associé à l'automate.

Exemple 3. Le mot `abca` appartient-il au langage associé à l'automate de la figure 1 ?

état courant	lettres restant à lire	lettre courante	état suivant
0	abca	a	1
1	bca	b	3
3	ca	c	pas de transition

On arrive sur un état dont ne part aucune transition associée à la prochaine lettre du mot ; le mot *abca* n'appartient donc pas au langage associé à l'automate.

Les figures 2 et 3 donnent des exemples d'automates associés aux langages $\mathcal{L}_1$ et $\mathcal{L}_2$ définis au début du sujet.

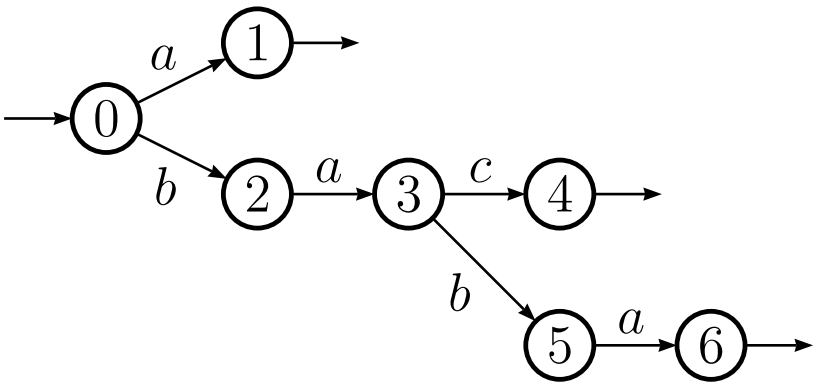


FIGURE 2 – Automate permettant de reconnaître les mots du langage $\mathcal{L}_1$

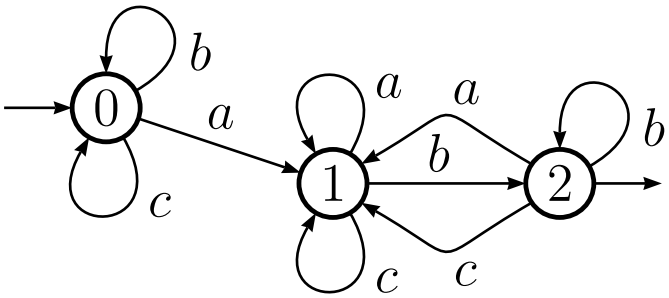


FIGURE 3 – Automate permettant de reconnaître les mots du langage $\mathcal{L}_2$

Travail à réaliser

Question de compréhension

En reproduisant le principe montré dans les exemples précédents, complétez le tableau suivant et déterminer si le mot `babacab` est reconnu par l'automate de la figure 3.

état courant	lettres restant à lire	lettre courante	état suivant
0	babacab	b	
	abacab	a	
	bacab	b	
	acab	a	
	cab	c	
	ab	a	
	b	b	
	∅		

Les réponses sont à indiquer dans votre fichier Python en commentaires, vous pouvez simplement donner la liste des états courants, la liste des états suivants et la réponse à la question posée.

Implémentation d'un automate

En utilisant vos connaissances sur les structures de données en Python, proposez une méthode pour représenter un automate par une ou plusieurs variables.
Afin de répondre à cette question, proposez simplement l'implémentation de l'automate de la figure 1 dans votre fichier Python.

Programme à écrire

En respectant votre implémentation, vous devez réaliser un programme qui permet :
— d'initialiser un automate sans transition (qui dépend uniquement du nombre d'états de l'automate, on précisera lesquels sont des états finaux)
— d'ajouter une transition entre un état et un autre en précisant le caractère.
— de déterminer si un mot est reconnu ou non par l'automate

Il est attendu de tester le programme au moins avec cet automate :

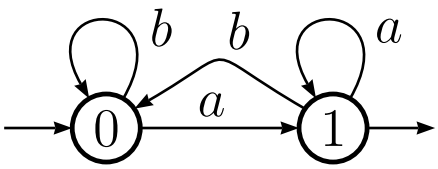


FIGURE 4 – Automate permettant de reconnaître les mots du langage $\mathcal{L}_3$

qui permet de reconnaitre les mots définis sur l'alphabet $\mathcal{A} = \{a, b\}$ se terminant par un a.