SAÉ S1.02 : ARN et protéïnes

Ce document correspond au sujet de la SAÉ S1.02 de l'année universitaire 2021/2022. Ce travail est à faire par binôme en autonomie. La SAÉ est découpée en deux parties. **La première partie devra être rendue avant le 5 décembre et la deuxième partie avant le 16 janvier.**

Attention : L'évaluation de cette SAÉ se fera notamment durant les contrôles de R101. L'évaluation de la partie 1 de la SAÉ se fera en partie lors du troisième contrôle de la ressource R101 (prévu le 8 décembre 2021). L'évaluation de la partie 2 se fera en partie lors du dernier contrôle de R101 (prévu le 21 janvier 2022). Pour les contrôles, il faut connaître le sujet et le code (les structures de données utilisées et les diffférentes fonctions).

Sujet

Partie 1

Les exercices s'appuient sur la notion d'ADN (https://fr.wikipedia.org/wiki/Acide_d%C3%A9soxyribonucl%C3%A9ique)/ARN (https://fr.wikipedia.org/wiki/Acide_ribonucl%C3%A9ique). On supposera dans l'ensemble des exercices que l'ADN est codé sous la forme d'une chaîne de caractères contenant uniquement les lettres majuscules `A`, `T`, `G`, `C` correspondant aux différentes bases (adénine, cytosine, guanine ou thymine). L'ARN sera codé sous la forme d'une chaîne de caractères contenant uniquement les lettres majuscules `A`, `U`, `G`, `C`.

Question 1

Définir la fonction `est_base` prenant en paramètre un caractère et retournant `True` si ce caractère correspond à une base de l'ADN (est un des caractères `A`, `T`, `G`, `C`), et `False` sinon.

À titre d'exemple, l'appel de la fonction `est_base` avec la valeur `"A"` doit retourner `True` alors que l'appel avec la valeur `"z"` doit retourner `False`.

Question 2

Définir la fonction `est_adn` prenant en paramètre une chaîne de caractères et retournant `True` si la chaîne correspond à un ADN (est constituée uniquement des caractères `A`, `T`, `G`, `C`), et `False` sinon.

À titre d'exemple, l'appel de la fonction `est_adn` avec la valeur `"ATGTCAAA"` doit retourner `True` alors que l'appel avec la valeur `"ATBOAATG"` doit retourner `False`.

Question 3

L'ARN est construit à partir de l'ADN en remplaçant la thymine `T` par l'uracile codé par la lettre `U`. Ainsi la transcription de la séquence ADN `ATTGCA` en ARN donne `AUUGCA`.

Définir la fonction `arn` prenant en paramètre une séquence d'ADN et retournant la séquence ARN associée.

À titre d'exemple, l'appel de la fonction `arn` avec la valeur `"ATGTCAAA"` doit retourner `"AUGUCAAA"`.

Attention : si la chaîne passée en paramètre ne correspond pas à de l'ADN, la fonction `arn` doit retourner la valeur `None`.

Question 4

Un codon est une séquence de trois bases sur un ARN (messager) spécifiant l'un des 22 acides aminés protéinogènes dont la succession sur l'ARN détermine la structure primaire de la protéine à synthétiser *(définition issue de Wikipedia)*.

Définir la fonction `arn_to_codons` prenant en paramètre une chaîne de caractères correspondant à de l'ARN et découpant cet ARN en codons. La fonction doit retourner un tableau contenant la liste des codons. Par exemple, l'appel de la fonction `arn_to_codons` avec l'ARN `"CGUUAGGGG"` doit retourner le tableau `["CGU", "UAG", "GGG"]` .

Remarque : Si le nombre de nucléotides dans l'ARN n'est pas un multiple de 3, les derniers sont ignorés. Par exemple, les ARN `"CGUAAU"` et `"CGUAAUGC"` donnent tous les deux la séquence de codons `["CGU", "AAU"]` .

Question 5

Chaque codon correspond à un acide aminé. Par exemple, le codon `UUU` correspond à l'acide aminé Phenylalanine. Plusieurs codons correspondent à un même acide aminé. Certains codons, correspondant à la fin du séquençage des acides aminés, sont appelés *codons stop*. C'est le cas du codon `UAA` par exemple.

Le fichier `data/codons_aa.json` contient la correspondance entre codons et acides aminés au format JSON. Les codons qui ne sont pas dans le fichier sont les codons stop.

- Définir la fonction `load_dico_codons_aa` qui prend en paramètre un fichier au format JSON et retourne la structure de données chargée en mémoire à partir du JSON.
- Définir la fonction `codons_stop` prenant en paramètre un dictionnaire dont les clés sont les codons et les valeurs les acides aminés correspondants (chaînes de caractères). La fonction retournera un tableau contenant l'ensemble des codons stop, c'est-à-dire l'ensemble des codons possibles avec les caractères `AUGC` qui ne sont pas des clés du dictionnaire.

Remarque : Créer une variable initialisée avec les données contenues dans un fichier au format JSON se fait très facilement en python. Pour plus d'information, relire la fin du cours 9 du module M1102 (https://github.com/iutVilletaneuseDptInfo/M1102/blob/master/09_Dictionnnaires/cours9.ipynb).

Question 6

Définir la fonction `codons_to_aa` prenant en paramètre un tableau de codons (correspondant par exemple à une valeur retournée par la fonction `arn_to_codons`) et le dictionnaire de correspondance entre codons et acides aminés. La fonction devra retourner un tableau contenant les acides aminés correspondant aux codons.

Attention : Si l'un des codons est un codon stop, alors la synthèse (traduction des codons en acides aminés) s'arrête. Par exemple, si le tableau de codons passé en paramètre est `["CGU", "AAU", "UAA", "GGG", "CGU"]` , alors le tableau retourné doit être `["Arginine", "Asparagine"]` car le codon `CGU` correspond à l'Arginine, le codon `AAU` correspond à l'asparagine et `UAA` est un codon stop.

Partie 2

Le séquençage des acides aminés ne commence que lorsque le *codon start* est rencontré. Ce codon est le codon `AUG` . On appelle séquence codante l'ensemble des codons compris entre le codon start et un codon stop. Dans une séquence codante, le codon `AUG` correspond à l'acide aminé *Methionine*. Une séquence de codons peut contenir zéro, une ou plusieurs séquences codantes. Par exemple, la

séquence `"CGU", "UUU", "AUG", "CGU", "AUG", "AAU", "UAA", "AUG", "GGG", "CCC", "CGU", "UAG", "GGG"` contient deux séquences codantes : `"CGU", "AUG"` et `"GGG", "CCC", "CGU"` puisque `"AAU"` et `"UAG"` sont des codons stop.

Remarque : Une séquence codante est délimitée par le codon start et un codon stop. Si la séquence de codons ne contient aucun codon stop, alors il n'y a pas de séquence codante.

Question 1

Définir la fonction `nextIndice` prenant en paramètre un tableau `tab`, un indice `ind` de `tab`, et un deuxième tableau `elements`. La fonction recherche dans le tableau `tab` à partir de l'indice `ind` et retourne l'indice de la première case du tableau `tab` contenant une valeur de `elements`.

Par exemple, l'appel de la fonction `nextIndice` avec le tableau `tab` égal à `["bonjour", "hello", "buongiorno", "ciao", "bye"]`, et le tableau `elements` égal à `["hello", "bye"]` doit retourner 1 si `ind` est inférieur ou égal à 1 lors de l'appel, et 4 sinon.

Remarque : si aucune valeur de `elements` n'apparaît dans `tab` à partir de l'indice `ind`, la fonction doit renvoyer la valeur `len(tab)`.

Question 2

Définir la fonction `decoupe_sequence` prenant en paramètre trois tableaux `seq`, `start` et `stop`. La fonction doit découper le tableau `seq` en séquences et retourner un tableau contenant les différents morceaux. Un morceau dans le tableau `seq` est une partie non vide de `seq` comprise entre une valeur de `start` et une valeur de `stop`.

L'appel de la fonction `decoupe_sequence` avec les tableaux

```
seq = ["val1", "début", "val2", "val3", "end", "val4", "fin", "begin", "val
5", "fin", "val6"]
start = ["début", "begin"]
stop = ["fin", "end"]
```

doit retourner

```
[
    ["val2", "val3"],
    ["val5"]
]
```

Question 3

Définir la fonction `codons_to_seq_codantes` qui prend en paramètre une séquence de codons et le dictionnaire de correspondance entre codons et acides aminés, et découpe la séquence de codons en séquences codantes. Les différentes séquences sont stockées dans un tableau. L'apel de la fonction `codons_to_seq` avec la séquence `["CGU", "UUU", "AUG", "CGU", "AUG", "AAU", "UAA", "AUG", "GGG", "CCC", "CGU", "UAG", "GGG"]` doit retourner le tableau :

```
[
    ["CGU", "AUG", "AAU"],
    ["GGG", "CCC", "CGU"]
]
```

Remarque : la fonction ne doit pas retourner de séquence codante vide.

Question 4

Définir la fonction `seq_codantes_to_seq_aas` prenant en paramètre un tableau de séquences codantes (même type que les valeurs retournées par la fonction précédente) et le dictionnaire de correspondance entre codons et acides aminés, et retournant un tableau contenant les différentes séquences d'acides aminés codées par les différentes séquences codantes. L'appel de la fonction `seq_codantes_to_seq_aas` avec le tableau retourné dans l'exemple de la fonction `codons_to_seq` doit retourner le tableau :

```
[
    ["Arginine", "Methionine", "Asparagine"],
    ["Glycine", "Proline", "Arginine"]
]
```

Question 5

Définir la fonction `adn_encode_molecule` prenant en parmètre un brin d'ADN, le dictionnaire de correspondance entre codons et acides aminés et une molécule (séquence d'acide aminées). La fonction doit retourner `True` si l'ARN obtenu à partir de l'ADN puis découpé en codons contient une séquence codante correspondant à la molécule, c'est-à-dire si la séquence d'acide aminée correspondant à une séquence codante est la même que la molécule. Par exemple, l'appel de la fonction `adn_encode_molecule` avec l'ADN `"CGTTTTATGCGTATGAATTAAATGGGGCCCCGTTAGGGG"` et la molécule `["Glycine", "Proline", "Arginine"]` doit retourner `True` .

Développement et rendu

La SAÉ devra être faite en binôme. Les fonctions doivent être implémentées dans un module appelé `biology` et les tests unitaires des fonctions dans le module `test_biology` . L'utilisation des fonctions du module devra être présentée dans un notebook appelé `using_biology.ipynb` . Le répertoire du projet devra également contenir un fichier `etudiants.txt` contenant les codes INE de deux étudiants du projet (un par ligne). Le répertoire devra aussi contenir un sous-répertoire `data` contenant le fichier `codons_aa.json` .

Un squelette de rendu de projet se trouve sur le dépôt gitlab de l'université.

Évaluation

La SAÉ sera notée de la manière suivante :

- Questions relatives à la SAÉ dans les contrôles : 12 points
- Vérification automatique du code via des tests unitaires : 3 points
- Qualité du code : 2 points
- Qualité des tests unitaires : 2 points
- Notebook de présentation : 1 point

Entrée [ ]:
```
1
```