Introduction

(intro sur la partie bio)

Un brin d’ARN peut adopter un certain repliement dans l’espace. Sa structure secondaire est alors caractérisée par l’appariement de certaines des bases qui le composent. Cet appariement peut être décrit avec un système de parenthésage dans lequel une parenthèse ouvrante et sa parenthèse fermante représentent deux bases appariées et les tirets représentent les bases non appariées, qui appartiennent à des boucles. Ces systèmes de parenthésage peuvent se traduire en arbres enracinés où chaque tiret est une feuille et chaque paire de parenthèse est un nœud interne.

Diagramme UML/d’organisation du projet…

Maquette interface utilisateur

Fonctionnalités développées

Le programme est capable de stocker l’information concernant un brin d’ARN. Il y a deux manières de générer un brin :

* Soit en donnant directement à la fonction getDataRfam le nom d’une famille Rfam, ce qui retourne un objet de type Brin avec sa séquence et son système de parenthésage, qu’on appelle Dash Bracket format.
* Soit en utilisant la fonction RandomRfam qui retourne le nom d’une famille au hasard dans la base de données. Il est alors possible de passer ce nom à getDataRfam pour obtenir le brin correspondant.

Une fois qu’un brin a été créé, on peut représenter son système de parenthésage sous forme d’arbre avec la fonction convertToTree. Elle prend en paramètre le parenthésage et retourne en sortie la racine de l’arbre correspondant, qui est de type Nœud et qui contient un ArrayList de nœuds fils. L’opération inverse est également possible grâce à la fonction convertToDashBracket, qui prend quant à elle la racine d’un arbre pour retourner le système de parenthésage associé. Elle utilise elle-même la fonction récursive convertToDashBracketInter.

Une fonction appelée nombreNoeuds prend en entrée la racine d’un arbre pour retourner le nombre total de nœuds qui le composent. Elle est utile pour les fonctions suivantes :

* Tout d’abord dans egalite, qui à partir de la racine de deux arbres pattern et target renvoie un booléen qui est vrai si les deux arbres sont égaux (c’est-à-dire que les deux arbres sont vides ou bien le nombre de fils est le même pour chaque sous-arbre) et faux sinon.
* Ensuite, dans comparaison, qui prend elle aussi deux racines en paramètre pour comparer l’arbre pattern avec l’arbre target. La racine du plus grand sous-arbre commun est alors retournée en sortie.

Une interface graphique a été développée (…)

Amorce d’analyse des résultats

Organisation humaine

Environnement technique

Pour ce qui est de l’environnement technique, nous avons d’abord tenté d’utiliser l’IDE IntelliJ pour le projet. Ayant rencontré quelques problèmes avec le logiciel, nous avons préféré nous tourner vers Eclipse, qui nous a semblé plus simple d’utilisation. Nous nous sommes servis de GitHub pour conserver et se partager facilement tous les fichiers en lien avec le projet. Nous avons également installé un module appelé CodeTogether qui nous a permis de coder ensemble en temps réel. Nous avons ainsi pu nous entraider et réfléchir ensemble tout en ne modifiant le code que d’un côté, nous évitant dans le même temps les problèmes de merge qui auraient pu survenir avec GitHub. Enfin, nous avons communiqué grâce à des appels sur Discord ou en salle Collaborate pour partager nos idées sur le code.

Choix techniques

Perspectives

Retour d’expérience