Chapitre 2

Semaine 2 : Exploration du graphe d'interactions protéine-protéine

Dans les questions suivantes, vous aurez besoin de lire, voire d'écrire, des graphes d'interactions entre protéines. Il ne vous sera jamais précisé quelle structure de données utiliser. Vous devrez trouver par vous-même, en fonction de la question posée, laquelle sera la plus adaptée.

2.1 Exploration du graphe global

2.1.1 Question exploration

Écrire une fonction `count_vertices(file)` qui compte le nombre de sommets d'un graphe.

2.1.2 Question exploration

Écrire une fonction `count_edges(file)` qui compte le nombre d'arêtes d'un graphe.

2.1.3 Question nettoyage

Écrire une fonction `clean_interactome(file)` qui lit un fichier contenant un graphe d'interactions protéine-protéine et y enlève (i) toutes les interactions redondantes, et (ii) tous les homo-dimères. Le graphe obtenu sera écrit dans un nouveau fichier au même format que le format de départ.

2.1.4 Question test

Il est fondamental de tester le bon comportement de vos fonctions avant de poursuivre plus avant dans l'étude des réseaux d'interactions protéine-protéine. Vos fonctions ne sont pas utilisables si nous n'avons pas moyen de nous assurer qu'elles font effectivement ce pour quoi elles ont été concues. Préparez toute une série de tests pour vérifier que vos fonctions ont le comportement que nous attendons d'elles.

2.2 Travail autour du degré des sommets

On nomme degré d'un sommet le nombre d'arêtes incidentes au sommet. Par exemple, dans le graphe de départ, si A est connecté à B et F, le degré du sommet A est 2.

2.2.1 Question degré

Écrire une fonction `get_degree(file, prot)` qui prend en argument un fichier contenant un graphe d'interactions protéine-protéine et le nom d'une proteine, et qui renvoie le degré de cette protéine dans le graphe.

2.2.2 Question degré

Écrire une fonction `get_max_degree(file)` qui renvoie le nom de la protéine de degré maximal ainsi que le degré de cette protéine.

2.2.3 Question degré

Écrire une fonction `get_ave_degree(file)` qui calcule le degré moyen des protéines du graphe.

2.2.4 Question degré

Écrire une fonction `count_degree(file, deg)` qui calcule le nombre de protéines du graphe dont le degré est exactement égal à `deg`.

2.2.5 Question degré

Écrire une fonction `histogram_degree(file, dmin, dmax)` qui calcule, pour tous les degrés `d` compris entre `dmin` et `dmax`, le nombre de protéines ayant un degré `d`. Si vous êtes courageux, essayer d'afficher le résultat sous la forme d'un histogramme en comme par exemple :

```
1 **
2 **
3 **
```

Que constatez-vous ? Quelle analyse pouvez-vous faire au vu de cette distribution ?

2.3 Semaine 2 : Structurez, commentez et déposez votre travail

2.3.1 Charge de travail semaine 2

Pour cette semaine de projet, nous estimons la charge de travail à entre 2h et 4h de programmation (selon votre aisance), et 2h de nettoyage de code et commentaires. La

compréhension des notions nouvelles dans le projet (sommets, arêtes, degrés) pourrait vous predre une heure de lecture en plus (maximum).

2.3.2 Objectifs de la semaine 2

1. Vos fonctions devront être écrites propremment.
2. Vos noms de variables doivent être cohénrents entre eux (pas de mélange de langue français - anglais, et comme le code est commencé en anglais, tous les noms de variables sont en anglais)
3. Vos noms de fonctions et de variables ont la même syntaxe (ici tout en minuscule avec des tirets du bas pour séparer les mots).
4. Vos noms de variables doivent contenir un suffixe indiquant leur type.
5. Vos noms de variables doivent faire moins de 20 caractères de long.
6. Toutes vos fonctions doivent être préfixées avec des commentaires qui indiquent le but de la fonction et sa signature.
7. Auncune fonction ne doit faire plus de 30 lignes.
8. Votre code doit être déposé sous git.
9. Vos enseignants doivent avoir accès à votre projet sous git.
10. Votre mise à jour du code doit être visible sur l'architecture du projet git.