Algorithmique
TD n° 8 : Hashing et bio-informatique

Comme vous l'avez vu lors des TDs précédents, il est possible de calculer la distance entre deux chaînes de caractères et de les aligner en fonction des similarités. Calculer ces alignements de manière exacte fait appel au remplissage d'un tableau de programmation dynamique de la taille du produit de la taille des mots.
Dans le contexte de la bio-informatique, on cherche souvent à comparer deux séquences ADN qui peuvent faire des milliards de lettres. À cause de la dimension des données, les matrices ne sont pas remplissables dans une espace ou temps raisonnable. Plutôt que de comparer les séquences, on préfère leur comparer leur contenus en k-mers (mots de taille k). On peut par exemple lister tous les k-mers des deux mots et compter le nombre de k-mers en commun pour avoir une approximation de la distance d'édition.

Exercice 1 : Codage

Un k-mer est un mot de taille k. En bio-informatique, les mots sont souvent des morceaux d'ADN composés uniquement des lettres A, C, G, T.

1. Comment transformer un k-mer en un entier de taille 2k bits ?
2. Écrire un algorithme qui, à partir d'une séquence ADN de taille n, écrit toutes les paires k-mer/valeur ($k \leqslant n$). Quelle est sa complexité ?

Exercice 2 : Tables de hachage

On veut maintenant comparer le contenu de deux séquences (et donc deux ensembles de k-mers). Pour cela, on peut commencer par énumérer tous les k-mers d'une des deux séquences et les stocker dans une table de hachage. Ensuite, on énumère tous les k-mers de la seconde séquence et on compte ceux qui entrent en collision avec ceux indexés dans la table.
Commençons par créer une table de hachage pour recevoir les k-mers de la première séquence. Pour simplifier, nous supposerons que tous les k-mers d'une séquence sont différents.

1. Si je veux créer une table de hachage dans laquelle je peux rentrer tous les k-mers possibles sans avoir de collision (2 k-mers à la même case), quelle taille doit faire ma table ?

   Pour réduire l'empreinte mémoire d'une telle structure de données, on créer une table t bien plus petite que la cardinalité de notre ensemble d'objets. On utilise ensuite une fonction de hachage qui transforme notre objet à insérer en un entier i ($0 \leqslant i \leqslant |t|$). Enfin, on insère l'objet à la position i.
2. Insérer les k-mers "ACT", "TGG" et "CAC" dans une table de taille 7 en utilisant comme fonction de hachage : $h(k\text{-}mer) = encode(k\text{-}mer)\%7$.
3. Que se passe-t-il lorsque l'on cherche à insérer le k-mer CAA ?

   Pour résoudre ce problème on peut choisir de ne pas avoir un seul élément par case mais une liste.
4. Quel impact cela a sur les complexités d'insertion et de requête de la table ?
5. Écrire la fonction de requête.

   La seconde façon de résoudre ce problème est d'envoyer l'élément dans une autre case quand la première est occupée. On peut par exemple l'envoyer à la première case disponible après la case d'origine.

6. Écrire une telle fonction d'insertion. Quelle est sa complexité ?
   Le problème de ce genre de solution est que lors d'un jeu de données biaisé qui envoie souvent vers la même case va avoir tendance à créer des grappes d'objets dans le tableau.
7. Proposez une méthode pour résoudre ce problème.
8. Proposer une fonction qui mette ensemble toutes les pièces précédentes et donne le pourcentage de k-mers en commun entre deux séquences.

Exercice 3 : Min-sketch

Quand l'ensemble de données est vraiment très grand, il n'est plus raisonnable de stocker tous les hash. En sous-échantillonnant notre ensemble de k-mers on peut tout de même obtenir une bonne estimation de la similarité entre nos deux séquences : On ne compare plus les hash de l'ensemble A à l'ensemble B mais un sous ensemble X de A et un sous ensemble Y de B.
Une méthode de sous-échantillonnage serait de ne pas gérer les collisions, mais en cas de conflit de juste remplacer la valeur précédente dans la table de hachage par la nouvelle valeur.

1. Calculez les sous-ensembles X et Y correspondant a l'ensemble A ({"ACT", "TGG", "CAC", "CAA"}) et l'ensemble B ({"CAA", "ACT", "TGG", "CAC"}) en utilisant la fonction de hash vue a l'exercice précédant. Est-ce que vous remarquez un problème ? Comment le résoudre ?
2. Grâce a cette méthode on peut calculer pour n'importe qu'elle séquence un Min-sketch, une table de hash d'une taille fixe qui contient un sous ensemble des k-mers présents dans une séquence. Écrivez une fonction qui réalise cette opération.
3. Écrivez une fonction qui estime la distance entre deux séquences grâce à un Min-sketch.
4. Comparez son résultat avec la fonction de l'exercice précédant. Est-ce qu'on obtient les même résultats ? D'où vient l'erreur ? Comment la réduire ? Est-ce qu'on peut estimer ce taux d'erreur ?