Feuille de travaux dirigés n° 3
Algorithmes

Exercice 3.1

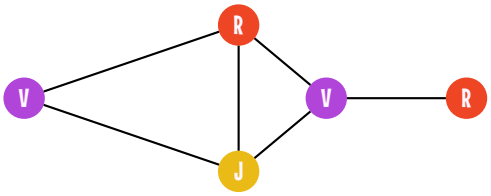
On considère l'ensemble de chaînes $\mathcal{T} = \{\text{CAG}, \text{TCC}, \text{TGC}, \text{AGG}, \text{GTC}, \text{GCA}, \text{ATG}\}$. On souhaite connaître la plus petite super-chaîne contenant tous les éléments de $\mathcal{T}$.

1. Représenter le graphe dont les nœuds sont les chaînes de $\mathcal{T}$ et où il existe une arête orientée de poids o entre chaque paire de nœuds (n_1, n_2) si n_2 est un suffixe de longueur o de n_1 ;
2. Trouver un chemin Hamiltonien dans le graphe. En déduire une super-chaîne pour $\mathcal{T}$;
3. Existe t-il un algorithme efficace pour trouver le chemin Hamiltonien de poids maximum ? Si oui, trouver la superchaîne pour $\mathcal{T}$ la plus courte ;

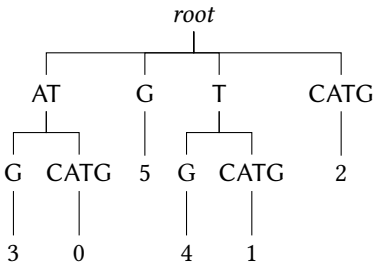
Exercice 3.2

On souhaite colorier les nœuds d'un graphe de façon à utiliser le minimum de couleurs en garantissant que deux nœuds reliés par un arc ne portent pas les mêmes couleurs (voir l'exemple figure 1a).

1. Proposer un *algorithme glouton* pour résoudre ce problème ;
2. Donner le code de la fonction `size_t greedy_coloring(graphp_t g)` prenant en entrée un graphe `g`, en coloriant les nœuds et retournant le nombre de couleurs nécessaire pour le colorier ;
3. L'algorithme précédent nous garantit-il de trouver le plus petit nombre de couleurs nécessaire ? Dans la négative, quel est son intérêt ?



(a) Coloriage de graphe.



(b) Arbre des suffixes de la chaîne `"ATCATG"`.

FIGURE 1 – Figures pour les exercices 3.2 et 3.3.

Exercice 3.3

On veut trouver toutes les occurrences d'une chaîne de bases (ex. : `"AT"`) dans un morceau d'ADN (ex. : `"ATCATG"`).

1. Écrire la fonction `void bf_matching(const string& strand, const string& pattern)` affichant à l'écran les positions de toutes les occurrences de `pattern` dans `strand` en utilisant un algorithme *brute force* ;
2. On considère l'*arbre des suffixes* pour la chaîne `"ATCATG"` présenté dans la figure 1b, où chaque feuille est étiquetée par la position de départ du suffixe dans la chaîne originale. Comment peut-on utiliser cet arbre pour trouver toutes les occurrences de la chaîne `"AT"` ?
3. Coder la fonction `void tree_matching(suffixtreep_t tree, const string& pattern)` affichant à l'écran les positions des occurrences de la chaîne `"AT"` en utilisant un arbre des suffixes.