M23 Graphes - épisode 3

1 - Coloriage glouton

Dans le fichier `algos.c`, écrire le code de la fonction suivante, qui réalise le coloriage glouton du graphe reçu en argument :

```
void coloriage_glouton(graphe const *g, uint8_t *couleur_sommet);
```

Le deuxième argument reçu par cette fonction est un tableau (préalablement alloué) qui contiendra à l'issue de l'appel la couleur de chaque sommet du graphe, représentée ici par un entier sur 8 bits. On utilisera les entiers successifs à partir de 0 en tant que couleurs.

Pour démarrer l'algorithme, il faudra initialiser la couleur de tous les sommets à une valeur particulière indiquant qu'il ne sont pas encore coloriés. On pourra utiliser la valeur `255` pour cela.

Les sommets seront traités dans l'ordre naturel des sommets dans le graphe (du sommet `0` au sommet `n-1`).

Rappel de l'algorithme de coloriage glouton *: pour chaque sommet `s` du graphe on affecte à `s` la plus petite couleur non-utilisée parmi les sommets adjacent à `s` déjà coloriés.*

Quelques pistes de réflexion :

- il pourra être utile de maintenir la plus grande couleur actuellement utilisée `couleur_max`
- pour déterminer la plus petite couleur disponible parmi les voisins, on pourra :
  - se servir d'un tableau de taille `couleur_max + 1` (1 case par couleur possible)
  - compter le nombre d'occurences de chaque couleur parmi les voisins
  - trouver le numéro de la première couleur dont le nombre d'occurences est nul
  - si toutes les couleurs sont utilisées, on augmente `couleur_max` de 1 et on affecte cette nouvelle couleur au sommet

---

2 - Permuter les sommets d'un graphe

Le résultat de l'algorithme de coloriage glouton dépend de l'ordre dans lequel les sommets ont été considérés.

Afin de pouvoir tester cet algorithme en prenant en compte les sommets dans des ordres différents, écrire la fonction suivante dans le fichier `algos.c` :

```
void appliquer_permutation(graphe const *src, graphe *dst,
                           size_t const *permutation);
```

Cette fonction reçoit en argument un graphe `src`, un graphe `dst` et un tableau d'entiers (`size_t`) `permutation`. On suppose que `permutation` a une taille `n` égale à l'ordre de `g` et contient une permutation des entiers compris entre `0` et `n-1`. Ainsi, pour un graphe d'ordre `5`, `permutation` pourra contenir par exemple les entiers suivants : `[3, 4, 1, 0, 2]`.

La fonction `appliquer_permutation` insère dans le graphe `dst` autant de sommets qu'il y a de sommets dans `src`. Enfin, si les sommets `i` et `j` du graphe `src` sont connectés par une arête, alors `permutation[i]` et `permutation[j]` sont connectés par une arête dans le graphe `dst`.

Afin de pouvoir appeler cette fonction, il sera utile d'écrire tout d'abord la fonction suivante dans le fichier `utils.c` :

```
void generer_permutation(size_t *permutation, size_t n);
```

Cette fonction reçoit un tableau d'entiers (`size_t`) `permutation` (préalablement alloué) et sa taille `n`, puis remplit le tableau avec une permutation *aléatoire* des entiers compris entre `0` et `n-1`.

---

3 - Nombre chromatique

On peut tenter d'approcher le nombre chromatique d'un graphe en réalisant un grand nombre de coloriages gloutons sur des permutations différentes d'un graphe et en gardant trace du nombre minimum de couleurs requises.

Dans le fichier `algos.c`, écrire la fonction suivante qui retourne le plus petit nombre de couleurs nécessaires au coloriage glouton du graphe `g` obtenu sur `n` permutations aléatoires de `g` :

```c
uint32_t estimation_nb_chromatique(graphe const *g, uint32_t n);
```

---

4 - Graphes aléatoires

Dans le fichier `algos.c`, écrire la fonction suivante qui génère un graphe aléatoire :

```c
void generer_aleatoire(graphe *g, size_t ordre, uint32_t k);
```

Pour générer un tel graphe, c'est l'existence de chaque arête potentielle qui va être décidée de manière aléatoire. Ainsi, pour chaque couple de sommet, on va tirer un entier à l'aide de la fonction standard `rand` et ajouter l'arête correspondante dans le graphe `g` uniquement si ce nombre est un multiple de `k`.

Indication: il n'est permis de faire un appel à `srand` que dans votre fonction `main`.

Tester la fonction d'estimation du nombre chromatique avec des graphes générés aléatoirement.