TP3 - Union Find

Ce qu'on voit lors de ce TP

- L'implémentation d'Union Find avec des forêts
- L'utilité d'Union Find dans certains problèmes (ici simples) d'algorithmique des graphes

Téléchargez les fichiers `union_find.h`, `verification_fonctions_union_find.c` et `verification_fonctions_union_find.h`. Déplacez-les dans un répertoire personnel, ainsi que vos deux fichiers `liste.c` et `liste.h`.

I) Forêts Union Find

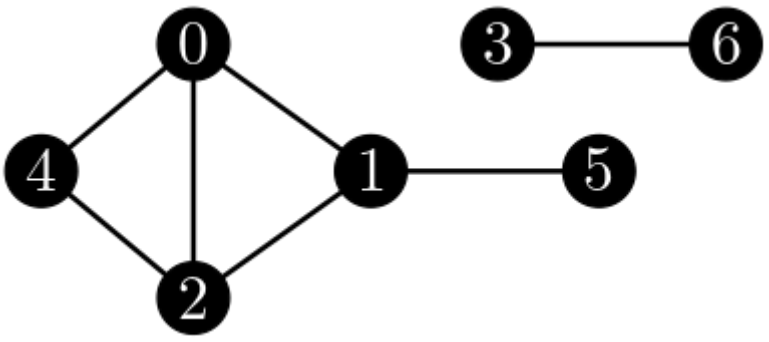
On va coder la structure avec les forêts que nous avons vue en cours. On fera les unions selon la taille des arbres et on utilisera la compression des chemins.

Créez un fichier `union_find.c` et codez toutes les fonctions dont le prototype se trouve dans `union_find.h`. N'oubliez pas de créer un fichier `main.c` pour tester vos fonctions, ainsi qu'un `makefile` afin de pouvoir compiler facilement votre TP.

II) Application : quelques problèmes de graphes

On va utiliser cette structure pour résoudre quelques problèmes de graphes, à savoir le **nombre de composantes connexes** et la **détection de cycles**.

Pour ces deux problèmes, on supposera que les sommets du graphe sont indexés de 0 jusqu'à $n-1$, où n est le nombre de sommets. Les graphes seront alors donnés sous la forme d'une liste d'arêtes + un nombre de sommets. Par exemple, pour le graphe suivant :



on pourra le coder avec la liste `[4,0,2,1,2,4,1,5,3,6,1,0,0,2]` (les arêtes sont regroupées par 2) et le nombre $n = 7$. On imaginera que les arêtes sont ajoutés au fur et à mesure : d'abord l'arête $(4, 0)$ puis $(2, 1)$ puis $(2, 4)$, etc.

Problème 1 : Nombre de composantes connexes.

Écrire une fonction qui prend en paramètres une liste comme au dessus et un nombre de sommets et qui renvoie le nombre de composantes connexes dans votre graphe.

Par exemple, pour le graphe du dessus, votre fonction devra renvoyer `2`.

Quelle est la complexité de votre fonction ? Est-ce optimal ?

Pour tester l'efficacité (et le bon fonctionnement !) de votre fonction, vous pouvez utiliser la fonction `test_nombre_composantes` se trouvant dans `verif_fonctions_union_find.c`. Par exemple, vous pourrez écrire `test_nombre_composantes(ma_super_fonction_qui_marche,1000)`.

Problème 2 : Détection de cycle

Écrire une fonction qui prend en paramètres une liste comme au dessus et un nombre de sommets et qui renvoie la position de l'arête qui provoque pour la première fois un cycle. Si le graphe n'a pas de cycle, on renverra -1. On considère que la première arête est indexée par 0, puis la deuxième par 1, etc.

Par exemple, pour le graphe du dessus, votre fonction devra renvoyer `6`, car l'arête $(1, 0)$ est la septième arête de la liste et elle provoque l'apparition d'un cycle avec $(4, 0)$, $(2, 1)$ et $(2, 4)$. L'arête $(0, 2)$ produit aussi un cycle mais elle se situe après l'arête $(1, 0)$.

Quelle est la complexité de votre fonction ?

Pour tester cette fonction, vous pouvez utiliser la fonction `test_apparition_premier_cycle`. Par exemple, vous pourrez écrire `test_apparition_premier_cycle(ma_fonction_qui_chie_dans_la_colle,1000)`.