26/04/2024

Projet de Algorithmique de graphes:

Tables de routage

Introduction :

Ce projet a été réalisé en binôme et consiste en la simulation du transfert de données vers le réseau. Il est séparé en 2 partie: la première nous permettait de visualiser le fonctionnement des sources vers le buffer et la 2ième nous demandait d'ordonné l'envoie des paquets en choisissant les bonnes source, on choisissait soit une source aléatoire, soit la source avec le plus grand nombre de paquet dedans, soit on choisissait les sources dans l'ordre et en boucle.

Initialisation du projet :

Lors de l'exécution du projet il y 3 variables principales qui sont :

- **Nb_noeud***(cf ligne 375)*, qui est la valeur qui précise le nombre de noeud de la table de routage;
  Mat_f*(cf ligne 398)*, matrice qui pour chaque noeud le chemin minimale et le poids pour aller vers les autres noeuds;
- **V_liste_arete***(cf ligne 389)*, liste qui aura comme élément toute les liens entre les nœuds qui sont des objets de la classe nœud.

Les variables secondaire sont taille_buffer et taille_paquet :

- **Liste_noeud***(cf ligne 376)*, une liste qui aura comme éléments tous les noeuds de la table de routage;
- **Mat***(cf ligne 397)*, matrice initialisée avant de trouver tous les plus courts chemins.

Architecture utilisée :

Pour l'architecture du projet nous avons créé 3 classes à savoir la classe **Noeud**, la classe **Arete** et la classe **Schema**.

- La classe **Noeud***(cf ligne 22)*, représente chaque noeud du graphe lors de l'affichage de celui-ci et elle est composé d'un constructeur et de 2 assesseur :

Le constructeur prends en argument un élément qui va être le **nom** du noeud et d'un élément qui va préciser de quel **tier** est le noeud;
- Les accesseurs sont pour récupérer le nom et le tier de chaque objet de la classe
- Noeud.

- La classe **Arete***(cf ligne 32)*, permet de représenter les arete qui vont relié different noeud, elle comporte un constructeur et 2 accesseurs :
  - Le constructeur prend en argument 2 **nom de noeud** et le **poids**;
  - Les accesseurs permettent de récupérer le tuple de nœud que l'arête relie et le poids.

- La classe **Schema***(cf ligne 42)*, permet de faire l'interface avec le graphe et les bouton qui le modifie, elle comporte un constructeur, une fonction **main**, **reset**, **affichage**, **affiche_chemin**, **entree** et **run** :
  - Le constructeur prend en paramètre la **matrice** de routage et la **liste des arêtes**;
  - La fonction **main***(cf ligne 55)* est la fonction principale de la classe schéma elle permet de placer les 100 noeuds du graphe puis vérifie si il sont superpose avec l'appel de la fonction verif_position qu'on expliquera dans la prochaine partie;
  - La fonction **reset***(cf ligne 87)* permet de réinitialiser le graphe, le remettre comme quand il était juste après l'appel de la fonction main au-dessus;
  - La fonction **affichage***(cf ligne 98)* permet d'afficher les arêtes dans le graphe;
  - La fonction **affiche_chemin***(cf ligne 129)* met en gras le chemin le plus court entre 2 noeuds;
  - La fonction **entree***(cf ligne 158)* créait les boutons relié aux fonctions reset et affiche_chemin;
  - La fonction **run***(cf ligne 175)* permet de lancer les fonctions main, affichage et run puis lance la page tkinter.

Toutes les fonctions vu au dessus prend comme paramètre uniquement leur objet d'origine avec le self.

Création des arêtes:

Il y a 3 fonctions de création d'arête, une pour chaque tier :

- La fonction **arete_t1***(cf ligne 184)* trouve tous les noeuds de tier 1 et entre chaque duo de noeud met avec 75% de réussite une arête entre les 2 noeuds et le poids de cette arête sera entre 5 et 10 inclus;
- La fonction **arete_t2***(cf ligne 204)* trouve tous les nœuds de tier 2 et ajoute 1 ou 2 arête avec les nœuds du tier 1. La fonction **compte***(cf ligne 216)* permet de savoir avec combien de nœud de tier 2 un nœud aussi de tier 2 est relié. Ceux ci ne doivent avoir une arête avec des noeuds de tier 2 que 2 ou 3 fois;
- Et enfin la fonction arete_t3 relie chaque nœud de tier 3 à 2 noeud de tier 2 aléatoire.

Autre fonction :

Enfin il reste les fonctions connexe, poids, d'initiation de la table de routage, routage, chemin, et de vérification de position :

- La fonction **connexe***(cf ligne 256)* vérifie si en partant du noeud 0 on peut rejoindre tous les autres noeud si oui alors le graphe est connexe sinon renvoie false et la boucle *(cf ligne 388)* recrée toute les arrête et vérifie si le nouveau graphe est connexe;
- La fonction **poids***(cf ligne 281)* initialise la matrice de routage à 0 et modifie toute dont on a déjà les arrête en assignant le poids dans la matrice. Elle renvoie alors cette matrice;
- La fonction **init_routage***(cf ligne 291)* initialise les matrices de routage de poids et de chemin grâce à la matrice renvoyée par la fonction poids. Elle renvoie la matrice de poids et la matrice du chemin;
- La fonction **routage***(cf ligne 313)* prend en arguments les deux matrices renvoyées par la fonction init_routage et cherche le plus court chemin entre chaque noeud puis met la valeur du poids dans la matrice de poids et le noeud suivant dans le chemin pour aller à un autre noeud dans la matrice de noeud. Puis renvoyer ces 2 matrices avec tous les poids et les chemins;
- La fonction **verif_position***(cf ligne 354)* vérifie si des nœuds sont trop proches les uns des autres et si oui les éloigne. Au début de celle-ci, il y a une boucle qui tourne 3 fois pour éviter qu' après les modifications de la première boucle il y ait encore des nœuds superposés.