Algorithmique et structures de données
Projet

Ensimag - 1A

Année 2020-2021

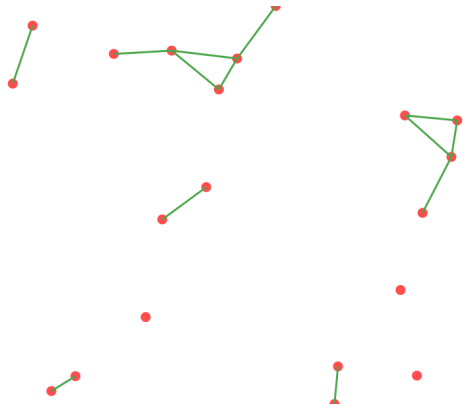

Figure 1: Exemple de graphe, les arètes sont vertes

1 Problématique

On s'intéresse dans ce projet à l'identification de structures dans un nuage de points. On considère en entrée un ensemble de points $p_1, p_2, \ldots, p_n$ du plan (tous distincts) ainsi qu'une distance seuil s. Deux points p_i et p_j sont *proches* si leur distance $d(p_i, p_j)$ est au plus s.

À partir de cette information, il est possible d'extraire un *graphe* $G = (V, E)$ tel que:

- l'ensemble V de sommets est l'ensemble de points : $V = \{p_1, p_2, \ldots, p_n\}$
- deux sommets p_i, p_j sont rélies par une arête s'ils sonst proches :

$$E = \{\{p_i, p_j\} \mid 1 \leq i < j \leq n,\ d(p_i, p_j) \leq s\}$$

Dans ce projet, nous cherchons à calculer la distribution des tailles des *composantes connexes* de G. Une composante connexe forme un sous-graphe

où tous les sommets sont reliés par transitivité (si on imagine les arètes comme des fils, il est possible de soulever toute la composante en tirant sur un seul de ses sommets).

Par exemple, sur le graphe de la Figure 1 on peut distinguer 9 composantes connexes différentes. En particulier, on distingue (au centre, en haut) une grosse composante de 5 sommets. Notons que *dans cette composante* tous les sommets ne sont pas reliés directement entre eux. Il n'existe pas d'arête entre le sommet de gauche et le sommet de droite. On voit également trois sommets isolés, chacun d'entre eux formant sa propre composante connexe.

Pour tout nuage de point (et seuil), on va chercher à dénombrer le nombre de points dans chaque composante. Votre algorithme devra afficher sur la sortie standard (*print*) le nombre de point de chaque composante (des plus grandes aux plus petites). Il n'est pas forcément nécessaire de calculer les composantes elles-mêmes, leurs tailles respectives pouvant suffire. Sur notre exemple, la solution est :

[5, 4, 2, 2, 2, 2, 1, 1, 1]

Attention: On vous demande de respecter exactement le même format d'affichage (incluant les caractères '[' et ']') ; le plus simple étant d'afficher un vecteur d'entiers.

2 Fichiers Fournis

On vous fournit un mini-module de géométrie "*geo*". Le module *geo* fournit une classe *Point* et une classe *Segment*. La bibliothèque contient également une fonction *tycat* qui permet d'afficher des itérables de points et segments dans *terminology*. Un petit exemple de son utilisation est donnée dans le fichier *hello.py*.

Le répertoire de travail contient également un ensemble de fichiers de tests "exemple_*.pts" qui contiennent chacun un distance et un ensemble de points.

On vous demande de modifier le fichier *main.py* en évitant de changer la fonction main. La fonction *load_instance* permet de charger un fichier de test. Le main lance donc les calculs sur tous les fichiers donnés en argument. Vous pouvez bien sûr créer d'autres fichiers python ou fonctions mais les tests automatiques seront réalisés en lançant *main.py* sur mes jeux de tests.

3 Résultats Attendus

Pour réaliser ce projet on vous demande d'écrire au moins un algorithme répondant au problème donné. De plus, il vous est demandé de réaliser un petit rapport de quelques pages incluant au évaluation expérimentale de

votre/vos algorithme(s). En particulier on vous demande de tracer au moins une courbe de performance. Nous vous demandons d'expliciter sous quelles conditions sur les entrées votre/vos algorithme(s) fonctionne(nt).

Le code et le rapport sont à uploader sur TEIDE pour le 18 avril 2021.

Attention ! Vous pouvez échanger entre vous sur les idées et méthodes de travail mais vous ne devez pas échanger de code. Chacun est en charge de la sécurisation de ses fichiers.

4 Gitlab

Cette année, j'ai créé un répertoire par équipe sur gitlab. Afin de pimenter un peu le projet je propose de réaliser un high-score.

Je dispose pour cela de trois jeux de tests.

- un jeu facile, que tout le monde devrait réussir à passer ;
- un jeu plus difficile ;
- un jeu très difficile.

Il y aura donc trois high-score, un par jeu de test.

Chaque nuit je vais lancer sur mon pc de bureau un script qui va pour chaque groupe:

- récupérer la dernière version du code sur gitlab avec un git pull ;
- lancer le test 1 (avec timeout) ;
- si le test 1 passe, lancer le test 2 (timeout plus élevé) ;
- si le test 2 passe, lancer le test 3 (timeout encore plus élevé).

Il n'y aura pas d'accès au réseau, pas d'accès au disque et pas d'accès à des modules python.

Je garde un historique des tous les temps obtenus et génère les high score utilisant pour chaque équipe le meilleur temps obtenu à chaque test (potentiellement des algos différents).

Vous pourrez retrouver sur chamilo un lien vers les pages de high-score.

Quelques informations supplémentaires :

- Le high score est là pour rigoler un peu mais ne sera pas pris en compte dans l'évaluation du projet qui évalue les algorithmes développés et vos expériences.
- C'est compliqué d'écrire des tests. Surtout pour faire passer tous les projets en une nuit. Je n'ai donc aucune idée que cela peut donner. Trop facile ? Trop difficile ?
- Si tout ne se passe pas comme prévu je rajouterai de nouveaux jeux de tests.