Algorithmes et structures de données 2
Laboratoire n°1 : Graphes non-orientés

17.09.2020

Introduction

Ce laboratoire est composé de trois parties distinctes.

1. Dans la première partie, vous utiliserez la représentation par matrice d'adjacence pour les parcours de graphes et afficherez leurs principales propriétés.
2. Dans la seconde partie, vous utiliserez un parcours en profondeur sur un graphe construit à partir des pixels d'une image, afin de colorier des régions.
3. Dans la troisième partie, vous utiliserez un parcours en largeur sur un graphe à symboles pour résoudre le jeu des « six degrés de séparation ».

Contraintes

- Veuillez prêter une attention particulière à la complexité (mémoire et processeur) de votre code, et justifier les choix que vous faites (par exemple pour privilégier l'un par rapport à l'autre).

Environnement de travail

- Pour expérimenter sur votre IDE, le plus simple est de télécharger les archives zip depuis Cyberlearn.
- Dans ce cas, faites attention à ce que les fichiers textes et d'images soient visibles par votre programme exécutable (dans CLion, ajouter le chemin du dossier de votre projet dans *Run -> Edit Configurations -> Working Directory*).

Travail demandé

Exercice 1 - Graphe à partir d'une image

- Le but de ce programme est d'explorer et compléter un exemple de bonne pratique pour le parcours de graphes non-orientés, en largeur (BFS) et en profondeur (DFS), avec calcul des composantes connexes (CC utilisant DFS).

- **Travail à faire :**
  - Analyser la structure des classes `BFS`, `DFS` et `CC`, notamment comprendre comment utiliser les méthodes publiques qu'elles offrent.
  - Compléter dans `GraphUsingAdjacencyMatrix.cpp` les trois fonctions membres `addEdge(int, int)`, `adjacent(int)` et `V()` mettant en œuvre un graphe avec représentation par matrice d'adjacence.

- Compléter dans `main.cpp` la fonction générique `CountEdges`, qui retourne le nombre d'arêtes du graphe.

Exercice 2 - Graphe à partir d'une image

- Le but de ce programme est de colorier en une couleur donnée les régions fermées d'une image en partant d'un point. Une image de test est fournie (`input.bmp`), ainsi que l'image que l'on souhaite obtenir (`goal.bmp`), qui sera comparée automatiquement à votre résultat (`output.bmp`).

- Vous utiliserez la librairie `bitmap_image.hpp` pour lire et écrire des images au format *bitmap*, principalement les méthodes `height()` et `width()` permettant d'obtenir la taille de l'image, ainsi que les méthodes `get_pixel(x,y,r,g,b)` et `set_pixel(x,y,r,g,b)` qui permettent d'obtenir ou de modifier les 3 composants entiers (*red*, *green*, *blue*) de la couleur du pixel à la position (*x*, *y*).

- Pour colorier une région à partir d'un point, chaque pixel de l'image sera le sommet d'un graphe. Le but sera de parcourir en profondeur le graphe recouvrant la région et changer les couleurs.

- Lorsqu'on lit une image, on doit la transformer en graphe avec la classe `GraphFromImage`. Cette classe est un wrapper permettant d'offrir les méthodes nécessaires d'un graphe.

- Chaque pixel sera un sommet, et ses sommets adjacents seront les pixels immédiatement au-dessus/au-dessous/à gauche/à droite ayant la même couleur. Attention aux bords de l'image !

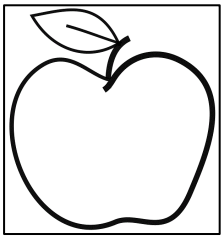
- Une *question bonus* est présente dans le `main()`. Vous pouvez y répondre directement sous forme de commentaire juste après la question.

- **Travail à faire :**

  - Implémenter dans le fichier `GraphFromImage.cpp` les six méthodes de la classe `GraphFromImage` selon les déclarations et les commentaires donnés dans l'entête `GraphFromImage.h`. Dans cet entête, vous ne devez pas modifier la partie `public:` mais vous pouvez ajouter des membres dans `private:` si vous le souhaitez.

  - Répondre à la question bonus dans le fichier `main.cpp`.

- **Résultat attendu :**

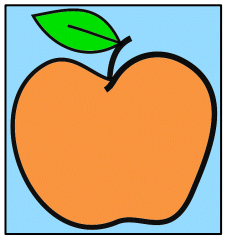


`input.bmp`

?

Sera comparé
à `goal.bmp`

`output.bmp`



`goal.bmp`

Exercice 3 - Graphe à symboles pour les « six degrés de séparation »

- Un graphe à symboles est une interface permettant de mapper des objets (dans notre cas des chaînes de caractères représentant des titres de films ou des noms d'acteurs) avec des entiers représentant les sommets dans nos implémentations habituelles de graphe (ici `GraphUsingAdjacencyLists`). Nous vous demandons d'implémenter cette interface. Les méthodes pour parcourir le graphe en largeur et trouver le plus court chemin sont données et ne doivent pas être changées.

- Le graphe à symboles vous permettra de résoudre le jeu des « six degrés de séparation », sous la forme des *Six Degrés de Kevin Bacon*[1].

    - Nous partons d'une liste de films avec leurs acteurs dans le fichier `movies.txt`, et le but est de trouver le lien entre deux acteurs – en particulier lorsque l'un d'eux est Kevin Bacon.

    - Un acteur est en lien avec Kevin Bacon s'ils ont joué dans le même film (degré 1), ou si l'acteur a joué dans le même film qu'un acteur ayant joué dans un même film que Kevin Bacon (degré 2) et ainsi de suite. (Le degré 6 est une hypothèse sur le degré maximal.)

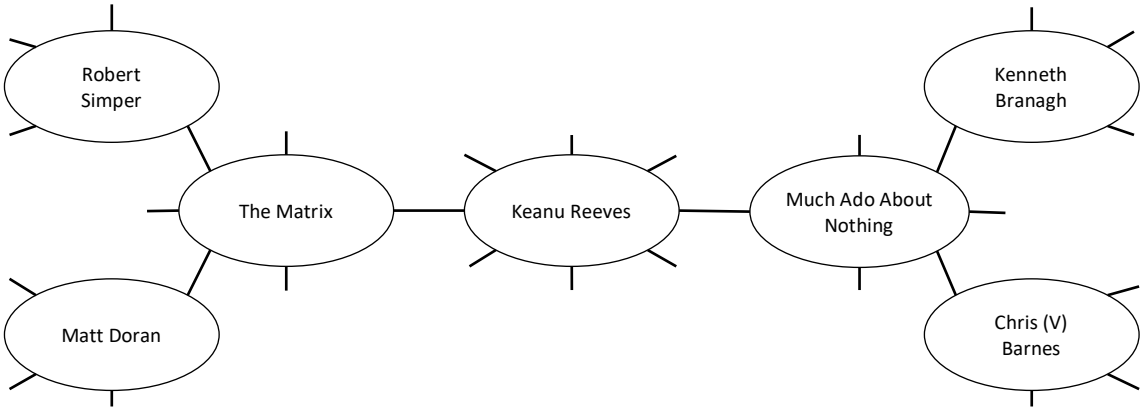
    - Par exemple, l'acteur Eric Beaver a joué dans *Bis ans Ende der Welt (1991)* où a joué également Max von Sydow (I). Celui-ci a joué aussi dans *Minority Report (2002)* en compagnie de Tom Cruise, qui a joué dans un même film (*A Few Good Men (1992)*) que Kevin Bacon ! Donc Eric Beaver est à trois degrés de séparation de Kevin Bacon.

    - Lorsqu'on indique dans le `main()` la source comme « Eric Beaver » et la destination comme « Bacon, Kevin », le programme devrait afficher ceci :

```
Eric Beaver
Bis ans Ende der Welt (1991)
Max von Sydow (I)
Minority Report (2002)
Tom Cruise
A Few Good Men (1992)
Kevin Bacon
```

- Données d'entrée : le fichier `movies.txt` contient sur chaque ligne un titre de film (il y en a 4'188), suivi des noms des acteurs qui ont joué dans le film, séparés par des '|'. L'extrait suivant permet de générer le fragment de graphe ci-après.

```
The Matrix (1999)|Keanu Reeves|Robert Simper|...
...
Much Ado About Nothing (1993)|Kenneth Branagh|Chris Barnes (V)|Keanu Reeves|...
```



[1] Wikipédia : http://fr.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon – voir aussi le site web suivant et ses expérimentations avec une base de données de ca. 412'000 acteurs : http://oracleofbacon.org/how.php.

- **Travail à faire :**
  - Compléter dans `GraphUsingAdjacencyLists.cpp` les trois fonctions membres `addEdge(int, int)`, `adjacent(int)` et `V()` mettant en œuvre un graphe avec représentation par listes d'adjacence.
  - Implémenter les 5 méthodes incomplètes ou manquantes dans `SymbolGraph.h` qui permettent de construire un graphe à symboles (ici par listes d'adjacence) à partir d'un fichier.
    - S'agissant d'une classe générique, on convient *d'écrire les définitions dans le fichier .h*.
    - Le constructeur contient déjà des indications utiles sur la lecture du fichier `movies.txt`.
    - Pour créer un graphe de symboles utilisant le type `GraphUsingAdjencyLists` (selon l'appel de `SymbolGraph` dans le `main()`), on doit connaître le nombre total de sommets, ce qui n'arrive que lorsqu'on a fini de lire le fichier. On acceptera ici les solutions qui lisent deux fois le fichier, mais la solution correcte est de stocker les arêtes dans une variable temporaire, puis les ajouter au graphe après la lecture complète du fichier.
    - Justifier vos choix de structures de données utilisées en fonction de la complexité (mémoire et processeur) de votre code.
  - Compléter le fichier main.cpp afin d'obtenir le résultat attendu, à savoir le chemin entre Eric Beaver et Kevin Bacon ainsi que tous les films avec Keanu Reeves.

Modalités pratiques

- À rendre sur Cyberlearn au plus tard le dimanche **29.09.2020** à **23h59.**
- Pas de rapport à rendre pour ce laboratoire, mais soignez vos commentaires et entêtes.
- Travail à faire par groupes de trois étudiant-e-s. Durée : 4 périodes encadrées. Sources sur Cyberlearn.
- Respectez les consignes indiquées sur Cyberlean.

Bon travail !