|  |  |
| --- | --- |
| LogoPoly_Genie1Classe.png | École Polytechnique de Montréal  Département de Génie Informatique et Génie Logiciel |

**LOG2810**

**STRUCTURES DISCRETES**

**Hiver 2017**

**TP1 : Graphes**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Remis par :**

|  |  |
| --- | --- |
| **Matricule** | **Prénom & Nom** |
| **1827808** | **Audrey Labrie** |
| **1804702** | **Sébastien Chagnon (groupe 2)** |
| **1818055** | **Sida Eric Li** |

**À :**

**David Johannès**

**Le 1er mars 2017**

**Introduction**

L’objectif de ce travail pratique est d’appliquer nos connaissances de la théorie des graphes. Pokémon Go est un jeu dont le but est d’amasser le plus de points en attrapant des Pokémons et en affrontant des adversaires. Le but de ce TP est donc de faciliter la vie de l’utilisateur en lui proposant des parcours qui optimiserait son gain de points. Pour ce faire, nous avons implémenté deux méthodes, une qui propose à l’utilisateur le plus court chemin pour atteindre un gain minimal donnée et le plus grand gain que l’utilisateur peut obtenir en parcourant au maximum une distance donnée.

Pour ce faire, nous avons créé une interface graphique permettant de visualiser les chemins proposés à l’utilisateur.

**Difficultés et solutions**

Nous avons rencontré trois difficultés lors de ce TP.

Premièrement, nous avons eu de la difficulté à modéliser le graphique. Il y avait trop d’arcs pour qu’il soit visuellement organisé. Nous avons donc décidé de seulement connecter les arcs des chemins les plus courts entre les différents sommets. Pour faire cela, nous avons classé les arcs en ordre croissant de leurs distances entres les sommets. Par après, nous itérons à travers cette liste d’arcs ordonnée, puis nous connectons les sommets seulement si la distance de l’arc est plus petit que la distance minimale entre les deux sommets déjà connectés par d’autres arcs passant par d’autres chemins. Par exemple, si sommet A et sommet B sont connectés par un arc de distance 10 et sommet B et sommet C sont connectés par un arc de distance 10 et qu’on rencontre un arc qui connecte sommet A et sommet C avec une distance de 25, on sait que le chemin le plus court va de sommet A à sommet B à sommet C qui est de 20, donc on n’ajoute pas un arc connectant sommet A à sommet C directement (distance 25 plus grand que distance 20).

Deuxièmement, il était difficile de trouver une librairie qui pouvais nous aider à représenter graphiquement notre solution. Donc, nous avons décidé de représenter le graphique en cercle à l’aide librairies Java. En fait, tous les sommets forment un cercle et au centre de celui-ci, on peut y observer des arcs qui sont des chemins entre les différents sommets. Le chemin choisit par la méthode plusCourtChemin() ou plusGrandChemin() devient coloré, ce qui permet à l’utilisateur de bien visualiser la solution.

Finalement, au tout début, nous passions des pairs et des tuples en paramètre de certaines méthodes. On a réalisé que c’était inutile de faire ceci, car on pouvait juste passer les paramètres simples.

**Conclusion**

Ce laboratoire a été utile car nous avons appris à adapter un algorithme de graphe à une application concrète. Nous avons étudié l’algorithme Dijkstra afin de créer le nôtre.

Pour le prochain laboratoire, nous aimerions devoir passer moins de temps sur l’interface graphique et plus d’accent sur les algorithmes.