Rapport ero1

Problème 1

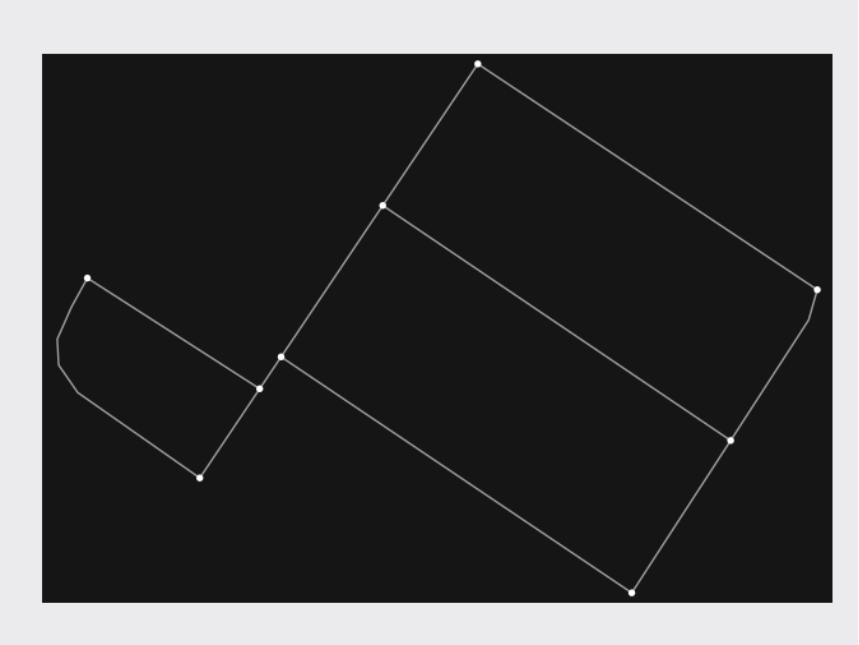
Après une brève analyse du sujet de recherche nous avons pris connaissance des problèmes sur lesquels nous allions travailler. Nous avons commencé par faire un brainstorming en groupe autour de la première question tout en la reformulant plus précisément : Quel algorithme nous permettrait de trouver un plus court chemin en prenant un graphe en entrée ? Comment pourrions-nous convertir une carte de Montréal en un graphe exploitable ? Après avoir fait des recherches sur les problèmes du type plus court chemin ou postier chinois et s'être souvenu des cours de théorie des graphes, nous avons considéré l'algorithme d'euler comme étant l'algorithme le plus approprié pour le problème du drone sachant que ce dernier doit obligatoirement passer par les routes. En effet, il permet de trouver le plus court chemin d'un graphe en passant par toutes ses extrémités et en revenant à son point de départ. Par analogie avec la ville de Montréal, les arêtes du graphe seraient les carrefour de la ville et les rues empruntés par le drone seraient les chemins parcourus par l'algorithme.
Nous allons mettre en place deux modèles : un théorique basé sur toute la ville de Montréal et un pratique basé sur un seul quartier de Montréal, le quartier Ghetto McGill. Pour pouvoir appliquer notre algorithme, nous avons besoin de données en input à exploiter et plus particulièrement d'un graphe représentant tous les chemins par lesquels nous devons passer.

Pour se faire nous avons utilisé la librairie osmnx qui va nous permettre, à partir d'une carte de la ville, de récupérer une matrice avec les coordonnées des noeuds (ou le noeud correspond a l'intersection de plusieurs routes entre différentes rues ainsi que la distance qui séparent deux noeuds).
Étant que le calcul de la distance par l'algo est exponentiel en fonction du nombre d'intersections, nous avons réduit notre cas pratique à une dizaine d'intersections.

Le premier algorithme utilisé est l'algorithme de Dijkstra. L'objectif de cet algorithme est de trouver l'itinéraire le plus court possible, donc aussi la distance, entre 2 nœuds donnés dans un graphique. Il est de la catégorie d'un algorithme gourmand, qui tente de trouver le chemin optimal en recherchant les voisins les plus proches et en s'ajustant.
Nous savons que si chaque sommet du graphe est pair (un sommet relié à un nombre pair d'autre sommets) le graphe est eulérien. Or dans la réalité il peut y avoir des sommets impairs. Nous allons donc les récupérer dans une liste.
Nous allons ensuite générer tous les appariements possible entre sommets impairs afin de les combiner pour en faire des sommets pairs.
Chaque distance calculée par Dijkstra grâce aux sommets sera stockée dans une liste.
Nous allons utiliser des boucles imbriquées simples pour obtenir la somme des pondérations de tous les tronçons.

Graphe des intersections de Montréal généré à l'aide de la bibliothèque OSMnx.

Mais nous avons décidé de nous attaquer à un seul quartier : Ghetto McGill.



Problème 2

Après premier passage de l'algorithme, correspondant au premier passage du drone, l'analyse qui en résulte nous donne une liste de rues enneigées ou non.

L’étape d’après est donc de modifier les données d’entrée du programme pour recalculer un chemin avec seulement les rues nécessitant le passage de la déneigeuse et donc avoir l’itinéraire final que nos déneigeuses auront à parcourir. Les routes empruntées par la déneigeuse seront les routes identifiées par le drone comme obstruées par la neige.

Problème 3

Avec cette distance nous pouvons aussi calculer les coûts d’une telle opération avec une simple fonction prenant en entrée la distance, le coût de l’essence à l’instant T, le salaire des déneigeurs et les possibles frais d’entretiens des engins. Nous avons pris la consommation d’essence d’un poid lourd comme référence soit 3km pour 1L, 0.8 cents le litre en moyenne et une distance de 40 km/h pour une déneigeuse.

Les problèmes de conceptions rencontrées ont été:
-Une difficulté pour récupérer les données du graph (transformer les coordonnées en une matrice utilisable pour notre algorithme)
-Des problèmes liées à la doc qui était compliqué à comprendre par moment
-L’algorithme utilisé nous donnait uniquement la distance minimal pour parcourir toute les rues sans nous donner le chemin à prendre
- La complexité de l’algorithme était aussi trop grande ce qui nous empêchait de faire des test à grande échelle (à partir de 40 rues c'était déjà trop) car cela prenait beaucoup trop de temps