# Prosit 1 : Premier contrat

## Mots Clés :

* Recherche opérationnelle : Généralement un problème de Recherche Opérationnelle se formule de la façon suivante : « Trouver des valeurs aux inconnues x1, x2,…, xn , sachant que ces inconnues doivent respecter un ensemble de propriétés (contraintes) les liant entre elles
* Algorithme : description d'une suite d'étapes permettant d'obtenir un résultat à partir d'éléments fournis en entrée
* ADEME : Agence de l'Environnement et de la Maîtrise de l'Energie,
* Arbre couvrant minimal : l'arbre couvrant minimum est la façon de construire un tel réseau en minimisant un coût représenté par le poids des arêtes
* Graphes et théorie des graphes
* Complexité – Asymptotique : la limite de la complexité lorsque le problème à traiter croît en termes de dimension
* Chaine eulérienne : une chaine qui parcourt toutes les arêtes d'un graphe connexe une et une seule fois

Un graphe connexe admet un parcours eulérien si et seulement si ses sommets sont tous de degré pair sauf au plus deux.

* Occupation CP

## Contexte :

Nous avons été missionnés pour optimiser le trajet des agents se chargent de la pose d’éclairage intelligents dans la ville.

## Problématique :

Comment optimiser l’itinéraire en ayant un passage unique dans chaque rue ?

## Contraintes :

* Passage unique
* Partir du point de départ et y revenir.
* Si le passage unique est impossible, donner le nombre de passages minimal.
* Bonne performance de l’algorithme.

## Généralisation :

* Conception d’algorithme eulérien.
* Les bases de Python

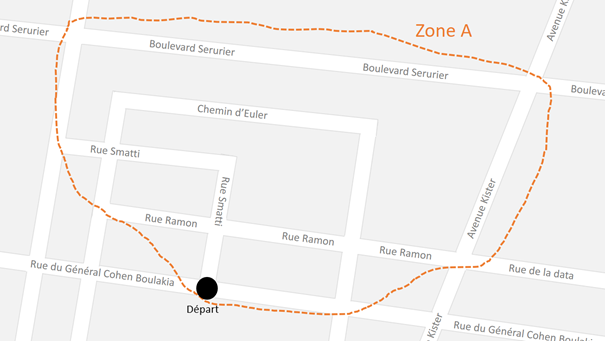
## Livrable :

* Modélisation du graphe
* Démonstration de l’algorithme et du programme. (Préciser le temps d’exécution du programme)
* Hypothèses / Pistes de solution :
* Conversion de l’itinéraire en matrice adjacente
* Chaque intersection serait un Sommet et chaque rue une arête

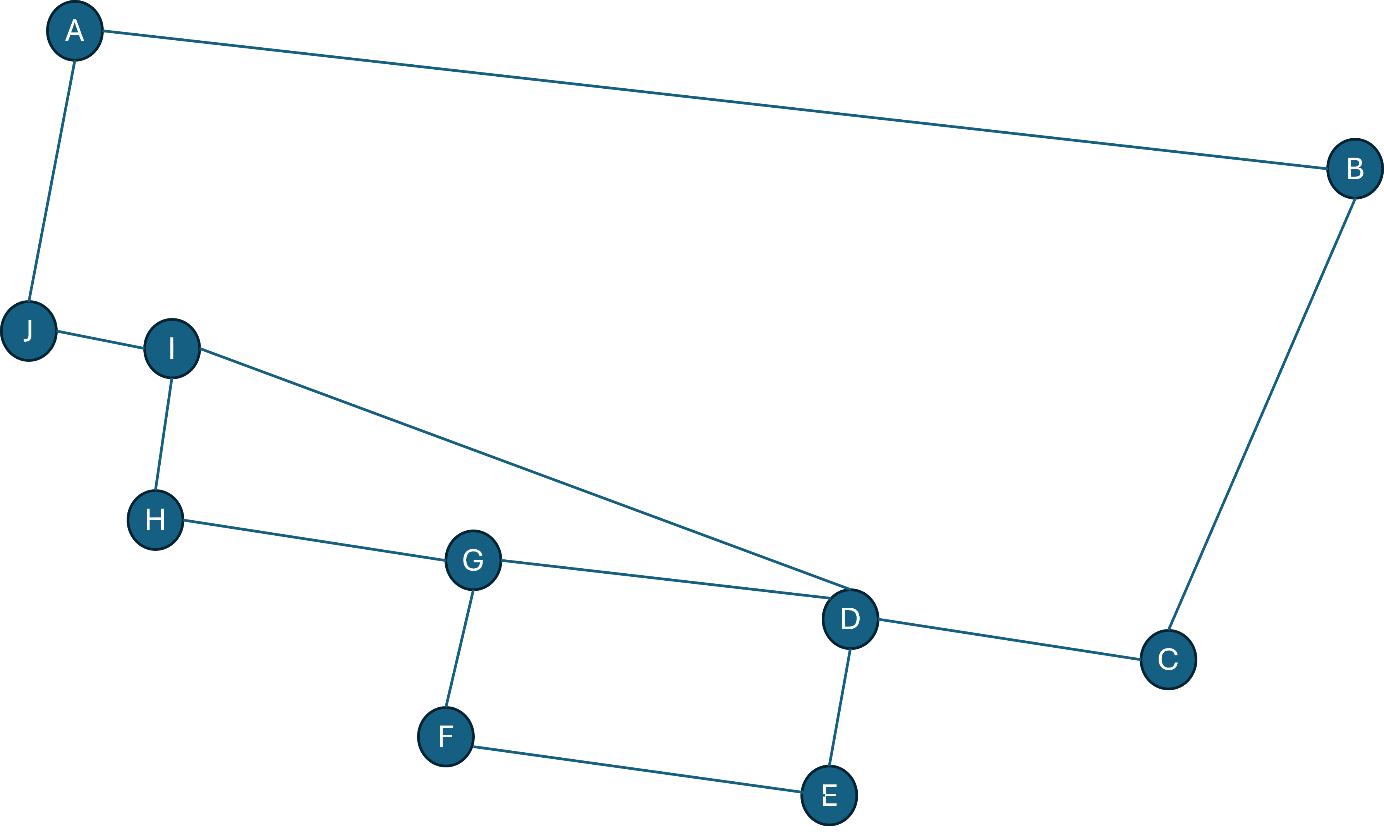
## Plan d’actions :

* Explorer les ressources
* Remodéliser le problème sous forme de graphe
* Etudier les différentes manières de modéliser un graphe sur python
* Faire un choix de modélisation
* Faire une condition de passage unique sur le programme
* Avoir une progression de l’itineraire sur le programme pour suivre l’ordre de passage sur les aretes
* Un algo pour passage unique, dans le cas contraire: un autre pour l’algo optimisé
* Implémenter l’algorithme (structure de données, analyse de performance)

## Réalisation :



On représente en graphe :



En regroupant les infos on a :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | J |
| Degré | 2 | 2 | 2 | 4 | 2 | 2 | 4 | 2 | 4 | 2 |

Si on en suit la propriété :

« Un graphe connexe admet un parcours eulérien si et seulement si ses sommets sont tous de degré pair sauf au plus deux. »

Ici tous les sommets sont reliés, chaque sommet est de degrés paire, nous somme donc en présence d’une chaine eulérienne

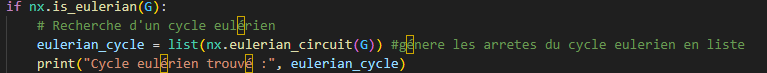
Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Tabdegre étant un tableau comportant les degrés de chaque sommets,

En connaissant le nombre de nombre paire, on peut déterminer s’il existe une chaine eulérienne ou non.

Via la librairie NetworkX nous pouvons connaitre la chaine eulerienne du graphe :



Ou G est le graphe et nx la librairie Networkx

Une image contenant ligne, cercle, diagramme, conception

Description générée automatiquement