|  |
| --- |
| CESI |
| Prosit : Petit détour |
| RECHERCHE OPÉRATIONNELLE |

|  |
| --- |
| Quentin Trappier – Hugo Durupt – Enzo Marcelli – Fabien Arrighi  27/05/2024 |

Contents

[I. Prosit aller 2](#_Toc167864950)

[Contexte 2](#_Toc167864951)

[Problématique 2](#_Toc167864952)

[II. Définition 3](#_Toc167864953)

[Graphe (boucle, sommets, arrête) 3](#_Toc167864954)

[Graphe simple / multigraphe 3](#_Toc167864955)

[Graphe connexe 4](#_Toc167864956)

[Graphe orienté 4](#_Toc167864957)

[Graphe pondéré 5](#_Toc167864958)

[Graphe cyclique 5](#_Toc167864959)

[Graphe biparti 6](#_Toc167864960)

[Densité d'un graphe 6](#_Toc167864961)

[Degré d'un nœud 6](#_Toc167864962)

[Arbre 7](#_Toc167864963)

[Liste d'adjacence 7](#_Toc167864964)

[Cycle hamiltonien 7](#_Toc167864965)

[Matrice d'adjacence 8](#_Toc167864966)

[Théorème d'Euler 8](#_Toc167864967)

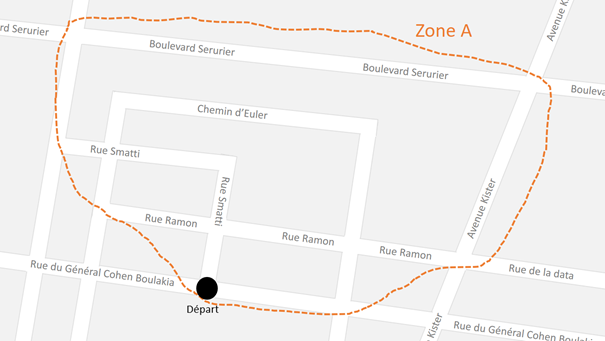
[Complexité 9](#_Toc167864968)

[III. Raisonnement 12](#_Toc167864969)

# Prosit aller

## Contexte

La création d’une ville intelligente nécessite la mise en place de boitiers sur les lampadaires classiques. Les équipes se contentent pour l’instant de suivre leur instinct concernant le chemin optimal. En tant qu’ingénieur, on nous demande de trouver une solution algorithmique au problème connu qui découle de cette situation : le problème du voyageur de commerce.



## Problématique

Comment faire en sorte d’optimiser le passage des techniciens dans les rues de la ville en évitant de passer plusieurs fois au même endroit ?

# Définition

## Graphe (boucle, sommets, arrête)

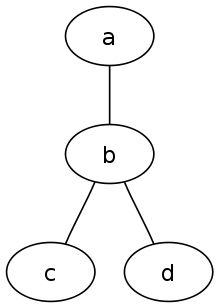
Un graphe est un ensemble de points (nommés nœuds ou sommets), dont certaines paires sont directement reliées par un ou plusieurs liens (nommés arêtes). Ces arêtes peuvent être orientées ou non orientées.

Dans un graphe orienté, la direction de l'arête est significative, indiquant un sens de déplacement de l'un des sommets vers l'autre.

Dans un graphe non orienté, les arêtes n'ont pas de direction spécifique.

## Graphe simple / multigraphe

Un graphe simple est un type de graphe où il existe au maximum une arête entre deux sommets donnés et il n'y a pas de boucles (arêtes qui relient un sommet à lui-même).

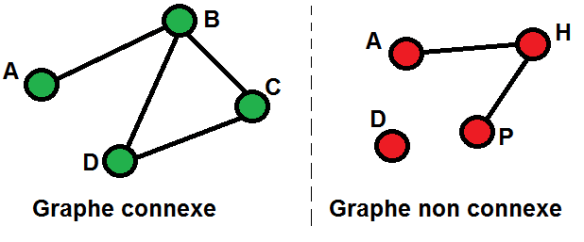


Un multigraphe peut contenir plusieurs arêtes entre les mêmes paires de sommets et permet également l'existence de boucles.

Une image contenant croquis, cercle, dessin, ligne

Description générée automatiquement

## Graphe connexe



Un graphe est dit connexe s'il existe un chemin entre chaque paire de sommets du graphe. En d'autres termes, pour chaque couple de sommets, il est possible de passer de l'un à l'autre en suivant les arêtes du graphe.

## Graphe orienté

**Une image contenant cercle, capture d’écran

Description générée automatiquement**

Un graphe orienté est un graphe dans lequel les arêtes ont une direction, c’est-à-dire qu’elles vont d’un sommet à un autre selon un sens précis. Un graffe dans lequel les arêtes n’ont pas de direction est un **graphe non orienté**.

## Graphe pondéré

**Une image contenant cercle, diagramme, ligne, dessin

Description générée automatiquement**

Un graphe pondéré est un graphe étiqueté dont toutes les étiquettes sont des nombres réels positifs ou nuls. Ces nombres sont les poids des liaisons (arêtes ou arcs) entre les sommets.

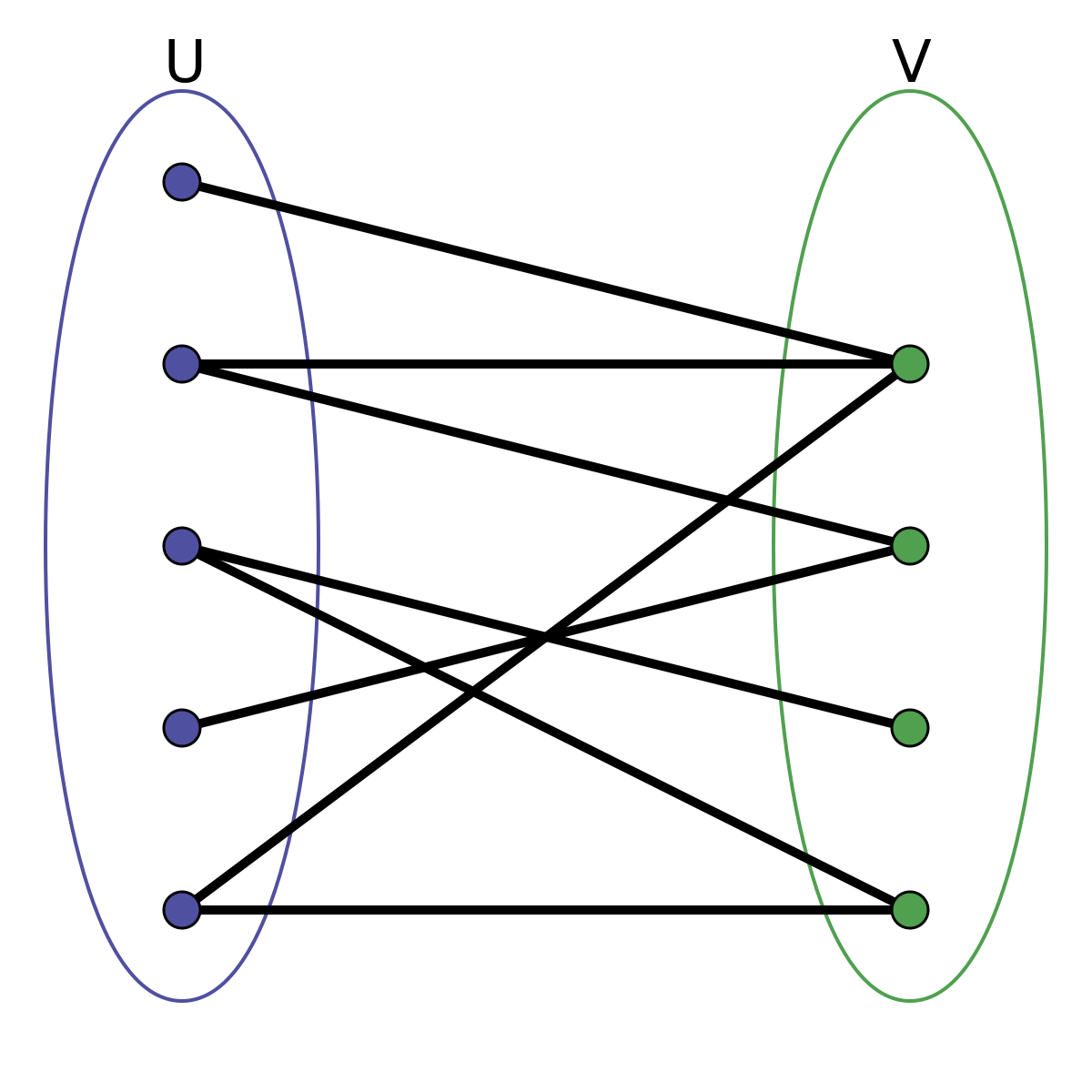
## Graphe cyclique

Une image contenant ligne, cercle, diagramme

Description générée automatiquement

Un graphe cyclique est un graphe qui contient au moins un cycle, c’est-à-dire une séquence de sommets et d’arêtes où l’on peut partir d’un sommet, suivre une série d’arêtes, et revenir au sommet initial sans répéter aucune arête.

## Graphe biparti

****

Un graphe biparti est un graphe dont l’ensemble des sommets peut être divisé en deux sous-ensembles disjoint U et V tels que chaque arête relie un sommet de U à un sommet de V. Autrement dit, il n’y a pas d’arêtes entre les sommets au sein d’un même sous ensemble.

## Densité d'un graphe

La densité d'un graphe indique à quel point les nœuds du graphe sont interconnectés. C'est une mesure de la proportion des connexions actuelles par rapport au nombre total de connexions possibles.

* Si la densité est proche de 1, le graphe est presque complet, c'est-à-dire que presque tous les nœuds sont connectés entre eux.
* Si la densité est proche de 0, il y a très peu de connexions entre les nœuds.

## Degré d'un nœud

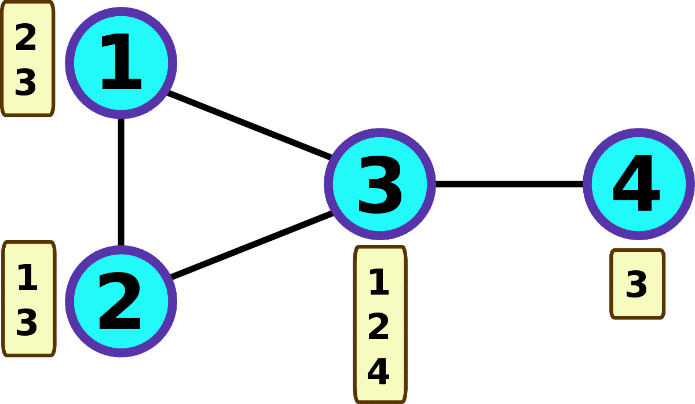
Le degré d’un nœud est le nombre d’arêtes qui sortent d’un nœud. Si le graphe et orienté, on peut différencier le degré sortant du degré entrant.

(Ordonnancement de tâches : le tri topologique)

Arbre

* Acyclique, non orienté, et connexe
* Une seule et unique racine
* Minimiser l'usage de ressources avec les arbres couvrants minimaux

## Liste d'adjacence



Une liste d'adjacence est une manière de représenter un graphe. Pour chaque nœud, on garde une liste de tous les nœuds auxquels il est connecté.

## Matrice d'adjacence

Une matrice d'adjacence est un tableau utilisé pour représenter un graphe. Dans ce tableau, chaque ligne et chaque colonne correspondent à un nœud (ou sommet) du graphe.

* Si le graphe est non orienté, la matrice est symétrique.
* Si le graphe est orienté, la matrice n'est pas nécessairement symétrique.

L'élément à la position (𝑖,𝑗) de la matrice est :

* 1 s’il y a une arête (ou arc) entre le nœud 𝑖 et le nœud 𝑗.
* 0 s'il n'y a pas d'arête entre ces deux nœuds.

Par exemple, pour un graphe avec 3 nœuds où le nœud 1 est connecté au nœud 2, et le nœud 2 est connecté au nœud 3, la matrice d'adjacence serait :

A number in a circle

Description automatically generated with medium confidence

## Cycle hamiltonien

Un cycle hamiltonien passe par chaque sommet d'un graphe une seule fois et revient au point de départ. On dit qu'un graphe possède un cycle hamiltonien s'il existe un cycle passant une, et une seule fois, par chaque sommet.

Une image contenant cercle, capture d’écran

Description générée automatiquement

Théorème d'Euler

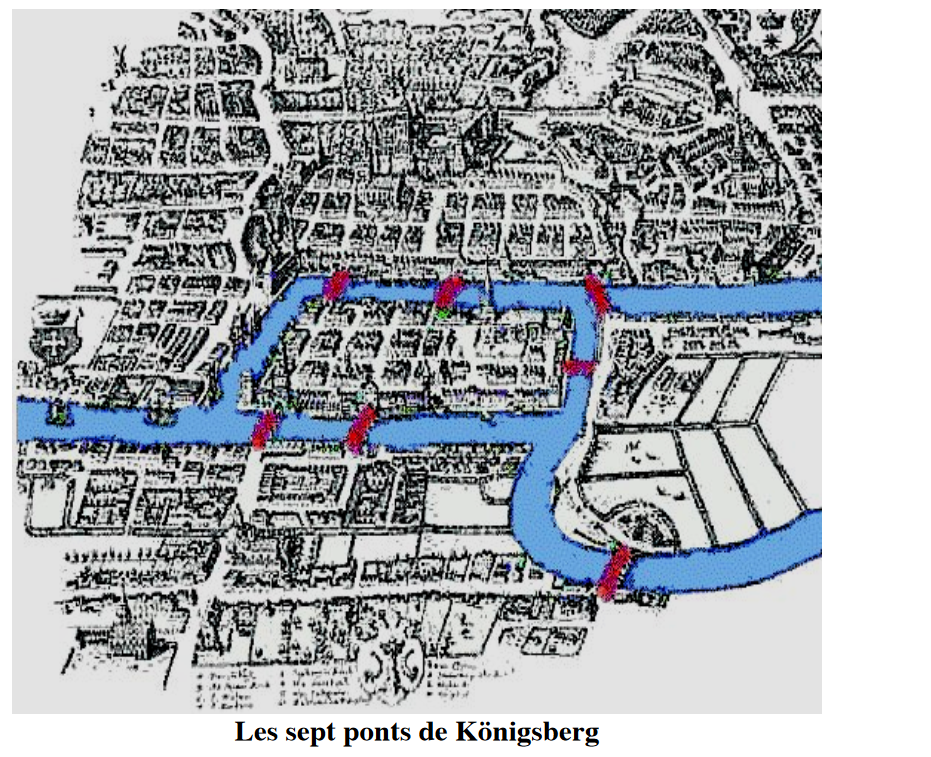
Le théorème d’Euler introduit les chaines eulériens et les cycles eulériens dans les graphes connexes non orientés.

Une chaine eulérienne est un chemin qui passe une seule fois par toutes les arrêtes. Le cycle eulérien est une chaine eulérienne qui finit sur le nœud de départ. Si un graphe possède un cycle eulérien, on dit que ce graphe est eulérien.

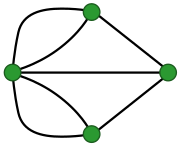
Pour savoir si un graphe possède une chaine eulérienne, il faut n’avoir qu’au minimum 2 nœud de degré impair. S'il y a plus de 2 nœuds de degré impair, aucune chaine eulérienne existe dans ce graphe.

Pour savoir si un graphe possède un cycle eulérien, il faut que tous les nœuds du graphe soit pairs.

Ce théorème est utilisé pour résoudre le problèmedes 7 ponts de Königsberg. Ce problème comporte plusieurs ponts qui relient plusieurs zones d’une ville entre elles. Le but de ce problème est de savoir s’il est possible de ne passer qu’une seule fois par chaque pont et de revenir à notre zone d’origine.



On peut la représenter sous forme de graphe :



Grâce à ce graphe, nous voyons qu’il y a plus d’un nœud de degré impair, un cycle eulérien est donc impossible.

Complexité asymptotique, notation de Landau

## Algorithme de Kruskal

L’algorithme de Kruskal permet de trouver un arbre couvrant minimum dans un graphe connexe non-orienté et pondéré.

Pour trouver l’arbre couvrant minimum, l’algorithme de Kruskal prévoit les étapes ci-dessous :

* Trier les arêtes (a) du graphique (G) par poids croissant. Ce faisant, les arêtes les plus légères sont examinées en premières.
* Partir d’un arbre (T) vide. C’est-à-dire un arbre qui ne contient aucune arête ; son poids est égal à 0. Sa construction est progressive en suivant les étapes suivantes.
* Sélectionner les arêtes par poids croissant. Il faut donc commencer par les arêtes les plus légères. Si l’ajout d’une arête n’implique pas la création d’un cycle dans le graphe, il convient de l’ajouter à l’arbre.
* Répétez l’opération jusqu’à ce que tous les sommets (S) soient reliés. Et ce, sans créer de cycle dans l’ARPM.

A green and red lines on a black background

Description automatically generated

## Complexité

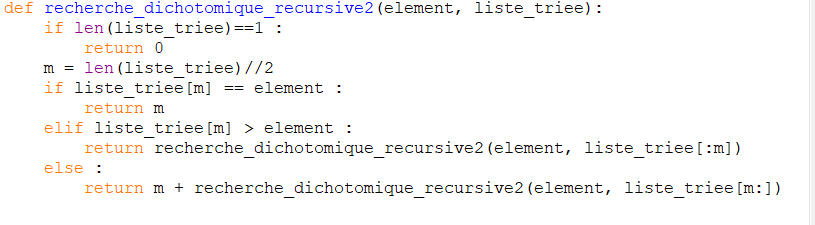
* O(1) : complexité constante

Exemple : Accès à un élément du tableau



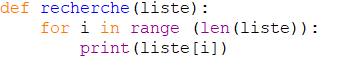
* O(log(n)) : complexité logarithmique

Exemple : Recherche dichotomique d’un tableau ([lien](https://fr.wikipedia.org/wiki/Recherche_dichotomique))



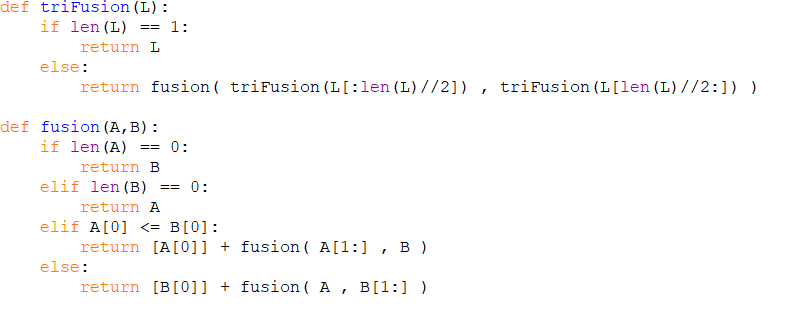
* O(n) : complexité linéaire

Exemple : Parcours classique d’une liste



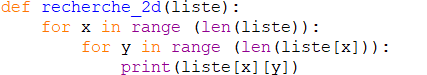
* O(n\*log(n)) : complexité linéarithmique

Exemple : Tri fusion, permettant de trier une liste en la divisant



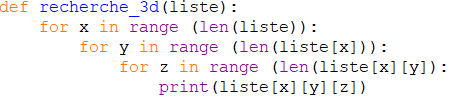
* O(n²) : complexité quadrique (polynomiale)

Exemple : Parcours d’une liste à 2 dimensions



* O(n3) : complexité cubique (polynomiale)

Exemple : Parcours d’une liste a 3 dimensions



* O(en) : Complexité exponentielle

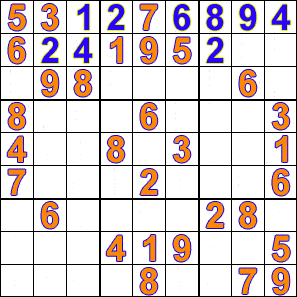
Exemple : Problème du sac à dos par brute force ([lien](https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos))

* O(n !) : complexité factorielle

Exemple : Problème du voyageur de commerce avec une approche naïve ([lien](https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_voyageur_de_commerce))

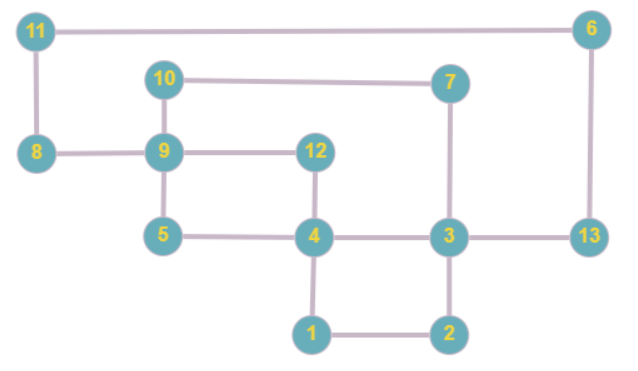
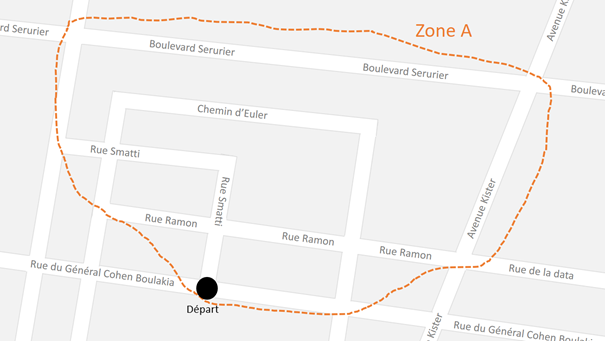
## Backtracking

En algorithmique, le backtracking est une famille d'algorithmes pour trouver des solutions à des problèmes algorithmiques, notamment de satisfaction de contraintes. Contrairement à une recherche exhaustive, un algorithme de backtracking construit incrémentalement des solutions candidates. Il abandonne la construction lorsqu'il ne peut compléter le candidat courant en solution valide. Il revient alors sur ses choix (d'où le nom) et en prend d'autres pour construire d'autres solutions candidates.



# Raisonnement

La première étape est de modéliser la situation sous forme de graphe. Ici nous avons représenté les intersections par des sommets et les rues par des arêtes :



Le graphe est disponible sur le lien suivant :

<https://graphonline.ru/fr/?graph=MjCHxMNpEyWNCQty>

Ce qui nous donne donc :

A map with blue circles and dots

Description automatically generated

Après ça, nous avons déterminé quel algorithme utiliser pour répondre à notre besoin. L’objectif ici est de trouver le meilleur chemin permettant d’intervenir sur tous les lampadaires. Cela doit donc se faire sans repasser pas les mêmes rues. Une méthode existe pour cela qui est le cycle Eulérien.

([Voir code](https://github.com/arrighi-fabien/Projet-Recherche-Operationnelle/tree/prosit-1))