

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université des Sciences et de la Technologie Houari Boumediene



Faculté d'Informatique

Departement : Systemes Informatiques

Spécialité: Big Data Analytics

Module : Etude de complexité

---

Mini projet à rendre

---

Présenté par:

- BENABED Anfel
- AMINE Ishak

Enseignant :

- Mr M.AIDER

2021-2022

# Table des matières

<b>1</b>	<b>Enoncé du projet</b>	<b>2</b>
<b>2</b>	<b>Environnement de travail</b>	<b>2</b>
2.1	Langage de programmation . . . . .	2
2.2	Outil de programmation . . . . .	3
<b>3</b>	<b>Réalisation du travail demandé</b>	<b>3</b>
3.1	Le graphe valué . . . . .	3
3.1.1	Définition . . . . .	3
3.1.2	Matrice de poids . . . . .	3
3.1.3	Construction d'un graphe valué . . . . .	4
3.2	Arbre couvrant de poids minimal . . . . .	6
3.2.1	Définition . . . . .	6
3.2.2	Propriétés . . . . .	6
3.3	Algorithme de Prim . . . . .	6
3.3.1	Problématique . . . . .	6
3.3.2	Domaine d'application . . . . .	7
3.3.3	Principe général de l'algorithme de Prim . . . . .	7
3.3.4	Theorème sur lequel repose l'algorithme de Prim . . . . .	7
3.3.5	Pseudo-code de l'algorithme de Prim . . . . .	7
3.3.6	Complexité de l'algorithme de Prim . . . . .	8
3.4	Algorithme de Kruskal . . . . .	9
3.4.1	Problématique . . . . .	9
3.4.2	Domaine d'application . . . . .	9
3.4.3	Principe général de l'algorithme de Kruskal . . . . .	9
3.4.4	Pseudo-code de l'algorithme de Kruskal . . . . .	10
3.4.5	Complexité de l'algorithme de Kruskal . . . . .	10
3.4.6	Différence entre Prim et Kruskal . . . . .	11
3.5	Implementation de l'application . . . . .	11
3.5.1	Graphe valué . . . . .	12
3.5.2	Algorithme de Prim . . . . .	14
3.5.3	Autres choix . . . . .	17

## List of Figures

1	Construction d'une matrice de poids. . . . .	4
2	Construction d'un graphe valué. . . . .	5
3	Resultat de la construction d'un graphe valué. . . . .	5
4	Algorithme de Prim. . . . .	8
5	Menu. . . . .	12
6	Graphe valué. . . . .	13
7	Autres choix - Graphe valué. . . . .	14
8	Implementation de Prim - Partie 01. . . . .	15
9	Implementation de Prim - Partie 02. . . . .	16
10	Autres choix - Prim. . . . .	17
11	Au revoir. . . . .	18
12	Message d'erreur. . . . .	18

# 1 Enoncé du projet

Codez dans le langage que vous voulez, le problème d'arbre couvrant de poids minimal. Soit l'algorithme de Kruskal soit l'algorithme de Prim tout en rajoutant le temps CPU.

De ce fait pour atteindre nos objectifs, nous avons divisé notre travail en trois parties :

- 1ere partie : Nous allons évoqué l'aspect du graphe valué.
- 2eme partie : Nous aborderons notre sujet principal qui est l'**Arbre couvrant de poids minimal** ainsi que son aspect algorithmique avec les deux algorithmes **Prim** et **Kruskal**.
- 3eme partie : Elle consiste à vous montrez les résultats de notre implémentation avec leurs explications.

## 2 Environement de travail

### 2.1 Langage de programmation

**Python :**

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet [7].



## 2.2 Outil de programmation



### Google Colaboratory :

Colaboratory est un produit de Google Research qui permet de partager des notebooks (projets) entre plusieurs utilisateurs, il est possible à partir de ces derniers d'écrire et d'exécuter du code Python à partir d'un navigateur sans avoir à télécharger ou installer quelconque logiciel au préalable [8].

## 3 Réalisation du travail demandé

### 3.1 Le graphe valué

#### 3.1.1 Définition

Un graphe valué  $G = (S, A, \cdot)$ , ou ce qu'on appelle aussi par graphe pondéré, est un graphe  $(S, A)$  qui peut être orienté ou non-orienté, muni d'une application :  $A \rightarrow \mathbb{R}$ . Cette application est appelée **valuation du graphe** [5].

#### 3.1.2 Matrice de poids

Par analogie avec la matrice d'adjacence, on peut définir la matrice des poids  $P(a_{i,j})$  du graphe dont les coefficients  $a_{i,j}$  correspondent aux poids des arêtes (ou des arcs dans le cas d'un graphe orienté) [6].

Le coefficient  $a_{ij}$  peut prendre les valeurs suivantes :

- 0 : S'il n'existe pas d'arêtes entre les sommets  $x_i$  et  $x_j$ .
- $P_{ij}$  : est le poids de l'arête entre les sommets  $x_i$  et  $x_j$ .

Vous trouverez ci-dessous le code de construction d'une matrice de poids :

```

G = nx.Graph()
m=[]

#Demander a l'utilisateur d'introduire le nombre de sommets (N) souhaités
N= int(input("Veuillez introduire le nombre de sommets du graphe: "))
print() #améliorer l'affichage des requetes

#Création de la matrice de taille NxN contenant les valeurs que l'utilisateur aura choisi
print("Veuillez introduire:") #améliorer l'affichage des requetes

for i in range(0, N):
    P=[] # P: Poids
    print() #améliorer l'affichage des requetes

    #Remplir les poids entre le sommet actuel et tous les sommets du graphe
    for j in range(0, N):
        val=int(input("    Le poids entre le sommet "+ str(i+1)+" et "+str(j+1)+" : "));
        P.append(val)
        if(not (val==0)): #Si le poids n'est pas nul, on l'ajoute à notre graphe
            G.add_edge(i+1, j+1,weight=val)

    m.append(P)

```

Figure 1: Construction d'une matrice de poids.

### 3.1.3 Construction d'un graphe valué

Vous trouverez ci-dessous le code de construction du graphe valué :

```

#pos:position
pos = nx.spring_layout(G)

options = {
    "font_size": 15,
    "node_size": 1500,
    "node_color": '#D7AFAF',
    "edgecolors": '#D7AFAF',
    "linewidths": 1,
    "width": 1,
}

labels = nx.get_edge_attributes(G,'weight')
nx.draw_networkx_edge_labels(G,pos,edge_labels=labels)

nx.draw_networkx(G,pos, **options)

ax = plt.gca()
ax.margins(0.10)
plt.axis("off")
ax.set_title('Le graphe valué G :')
C=0

plt.show()

```

Figure 2: Construction d'un graphe valué.

Nous obtiendrons alors le résultat suivant :

Veuillez introduire le nombre de sommets du graphe: 3

Veuillez introduire:

```

Le poids entre le sommet 1 et 1: 1
Le poids entre le sommet 1 et 2: 5
Le poids entre le sommet 1 et 3: 14

Le poids entre le sommet 2 et 1: 6
Le poids entre le sommet 2 et 2: 8
Le poids entre le sommet 2 et 3: 13

Le poids entre le sommet 3 et 1: 22
Le poids entre le sommet 3 et 2: 30
Le poids entre le sommet 3 et 3: 19

```

Figure 3: Resultat de la construction d'un graphe valué.

**Remarque :** Le résultat final du graphe valué se trouve dans la partie implementation de l'application.

## 3.2 Arbre couvrant de poids minimal

### 3.2.1 Définition

Etant donné un graphe  $G$  non orienté connexe valué, **un arbre couvrant de poids minimal (ACM) de ce graphe** est un arbre dont la somme des poids de ces arêtes est minimale. D'une autre manière, c'est le minimum entre toutes les sommes des poids des arêtes des autres arbres de ce graphe. L'arbre couvrant de poids minimal est aussi connu sous certains autres noms, tel qu'arbre couvrant minimum ou encore arbre sous-tendant minimum.

### 3.2.2 Propriétés

L'arbre couvrant de poids minimal repose sur trois propriétés que nous citons:[1]

- **Multiplicité ou unicité:** un graphe connexe peut comporter plusieurs arbres couvrants minimum différents, mais si tous les poids sont différents, alors il est unique.
- **Propriétés des cycles:** Lorsqu'on fait face à un cycle, si une arête est de poids strictement plus grand que les autres, alors cette arête n'est pas dans l'arbre couvrant de poids minimal.
- **Poids de l'arbre d'un ensemble de points:** si une arête est de poids strictement plus grand que les autres, alors cette arête n'est pas dans l'arbre couvrant de poids minimal dans  $\mathbb{R}^d$  avec la norme euclidienne, quel est le poids de l'arbre couvrant minimal. Il est de l'ordre de  $n^{1/d}$  en moyenne et avec probabilité 1.

## 3.3 Algorithme de Prim

### 3.3.1 Problématique

Le problème consiste à trouver un arbre couvrant de poids minimal à partir d'un graphe connexe non orienté valué [2].



### 3.3.2 Domaine d'application

Les arbres de portée minimale ont des applications directes dans la conception de réseaux, notamment les réseaux informatiques, les réseaux de télécommunications, les réseaux de transport, les réseaux d'adduction d'eau et les réseaux électriques (pour lesquels ils ont été inventés).

Ils sont utilisés comme des sous-programmes dans des algorithmes pour d'autres problèmes, notamment l'algorithme de Christofridès pour l'approximation du problème du voyageur de commerce.

Hors du contexte des réseaux, ils sont également utiles par exemple pour le partitionnement de données et le traitement d'image.

### 3.3.3 Principe général de l'algorithme de Prim

L'algorithme de Prim est un algorithme glouton qui calcule un arbre couvrant minimal dans un graphe connexe valué et non orienté. Il part du principe qu'à partir d'un arbre initial réduit à un premier et seul sommet, puis il augmente à chaque itération la taille de l'arbre en le connectant au plus proche voisin libre.

**Remarque:** Un algorithme glouton est un algorithme qui effectue à chaque instant, le meilleur choix possible, sans retour en arrière et sans sauter les étapes suivantes, afin d'atteindre le résultat le plus optimal possible.

### 3.3.4 Théorème sur lequel repose l'algorithme de Prim

Considérons un graphe non orienté pondéré  $G=(X, E, V)$  ainsi que l'ensemble des arbres de recouvrement notés  $(X, F, V)$  de  $G$ . (il est possible pour un graphe d'avoir plusieurs arbres couvrants différents).

Notons  $(X, F, V) \rightarrow (X', F', V)$  l'ensemble des arbres de recouvrement du graphe  $G$  qui contiennent l'arbre  $A=(X', F', V)$ . Tel que  $X' \subseteq X$  et  $F' \subseteq F$ .

Parmi les arbres de recouvrement minimaux du graphe  $G=(X, E, V)$  pour lesquels le sous-arbre  $A=(X', F', V)$  est imposé, il en existe au moins un qui contient l'arête de valeur minimale ayant une de ses extrémités dans  $X'$  et son autre extrémité dans  $X-X'$  [2].

### 3.3.5 Pseudo-code de l'algorithme de Prim

Les étapes les plus importantes de l'algorithme de Prim sont:

- Il faut commencer par choisir un sommet du graphe  $G$  qu'on nommera 'Sommet source'.
- Trouver l'arête de poids minimal connectée à ce sommet là et la rajouter à l'arbre.
- Refaire la 2eme étape jusqu'à ce que tous les sommets de  $G$  soient ajouter à l'arbre.

#### Algorithme de Prim

```

d(s1) = 0; ouvrir(s1);
Pour tout k de 2 à n faire d(sk) = + ∞ ;
FinPour
Pour k de 1 à n faire
    Soit x un sommet ouvert tel que d(x) est minimum
    Examiner(x);
FinPour.

```



Robert C. Prim

#### Examiner(x)

```

    Pour tout voisin non fermé y de x faire
        Si d(y) > c(x,y) alors
            d(y) = c(x,y); ouvrir(y);
        FinSi
    FinPour
    fermer(x);

```

Figure 4: Algorithme de Prim.

### 3.3.6 Compléxité de l'algorithme de Prim

Pour un graphe  $G$  ayant  $n$  sommets, nous devons :

- **Initialisation des variables:** Elle aura une complexité de  $O(1)$ .
- **Marquage d'un nouveau sommet:** Cette étape se répétera  $n$  fois (elle se repétera autant de fois que le nombre de sommet de  $G$  car il faut tous les marquer) avec la boucle qui est à l'intérieur qui sert à parcourir tous les poids entre les sommets. Ça fera alors une complexité de  $O(n^2)$ .

On conclue alors que Prim est un algorithme polynomiale de complexité  $O(n^2)$ .

## 3.4 Algorithme de Kruskal

### 3.4.1 Problématique

Supposons que nous travaillons sur un graphe connexe, certains problèmes obligent à transformer ce graphe en un arbre (graphe sans cycle) qui contient tous les sommets du graphe et quelques arêtes. On dit alors qu'on a un arbre couvrant du graphe.

Parfois, lorsque le graphe est valué ( chaque arête possède un poids qui est un nombre qui représente le coût de cette arête) , il s'agit de chercher un arbre recouvrant de poids minimum ( expliqué pour l'algorithme de prime ) [3].

### 3.4.2 Domaine d'application

On peut retrouver l'application de cette algorithme dans plusieurs domaines tels que :

- **Réseau maritime:** Supprimer les liaisons maritimes les moins rentables en préservant l'accessibilité aux différents ports.
- **Réseau électrique:** Jeter des réseaux électriques de façon à minimiser le coût total du câblage en liant toutes les maisons.
- **Réseau LAN:** Si vous avez un grand LAN avec plusieurs commutateurs, c'est essentiel de s'assurer que seulement un nombre minimum de paquets seront transmis à travers le réseau.
- **Divers:** Générer des labyrinthes.

Il est aussi utilisé dans d'autres algorithmes de résolution tel que des algorithmes de résolution pour le problème du voyageur de commerce.

### 3.4.3 Principe général de l'algorithme de Kruskal

L'algorithme construit un arbre couvrant minimum en sélectionnant des arêtes par poids croissant (en pratique, on trie d'abord les arêtes du graphe par poids croissant), on test pour chaque une d'elle si elle nous forme un cycle. Si oui, on ne la prend pas car on veut construire un arbre.

A la fin de l'exécution, nous obtenons un arbre couvrant minimum (Le graphe obtenu est aussi connexe).

#### 3.4.4 Pseudo-code de l'algorithme de Kruskal

Les étapes les plus importantes de l'algorithme de Kruskal sont:

- Trier les  $m$  arêtes du graphe par valeurs croissantes.
- Soit  $a_0$  une des plus petites arêtes :  $F = a_0$ .
- Soit  $p$  le nombre d'arêtes déjà placées dans le graphe partiel :  $p = 1$ .
- Soit  $n$  le nombre de points du graphe  $(X, E, V)$ .
- Tant que  $p$  inférieur ou égal à  $n-2$  et que toutes les arêtes n'ont pas été examinées  
Faire  
Soit  $a_0$  la plus petite arête non encore examinée. Si l'ajout de  $a_0$  à  $F$  ne crée pas de cycle dans le graphe partiel  
alors  
ajouter  $a_0$  à  $F$ ;  
 $p = p + 1$  ;  
fsi  
Fait
- Si  $p = n-1$  alors Ecrire("F contient les arêtes d'un arbre de recouvrement minimal");  
Sinon Ecrire("le graphe initial n'était pas connexe et il n'est pas possible de trouver un arbre de recouvrement minimal");  
finsi

#### 3.4.5 Complexité de l'algorithme de Kruskal

- L'algorithme est basé sur un tri des poids ( étape 1 ) de complexité  $O(|E| \log |E|)$  tel que  $E$  est le nombre d'arête du graphe.
- L'étape 5 comporte au plus  $E$  itérations et pour chaque itération, l'algorithme vérifie si l'arête que l'on souhaite ajouter n'induit pas à un cycle et opère des affectations, ajouts, additions... Ce qui donne  $O(|E|)$  opérations.
- Nous pouvons conclure que la complexité de l'algorithme est en  $O(|E| \log |E|)$  qui est la complexité du tri des poids.

**Remarque 01:** Le nombre d'arêtes  $m$  d'un graphe connexe peut varier de  $n-1$  à  $((n-1)n)/2$  donc, l'algorithme de Kruskal est d'autant plus rapide que le graphe connexe est pauvre en arêtes.

**Remarque 02:** Il existe d'autres manières de procéder pour réduire la complexité de l'étape 5 comme utiliser une structure de données de type anti-arborescence qui fait passer la complexité de l'étape à  $O(\log n)$  mais cela ne va pas impacter la complexité globale de l'algorithme.

### 3.4.6 Différence entre Prim et Kruskal

Les algorithmes de Prim et Kruskal sont des algorithmes gloutons qui trouvent un arbre couvrant minimum pour un graphe non orienté pondéré connecté (connexe).

Nous remarquons que le résultat obtenu est le même mais le principe de fonctionnement diffère :[9]

- L'algorithme de Prim s'initialise avec un nœud, tandis que l'algorithme de Kruskal démarre avec une arête.
- Les algorithmes de Prim s'étendent d'un nœud à un autre tandis que l'algorithme de Kruskal sélectionne les arêtes de manière à ce que la position de l'arête ne soit pas basée sur celle qui la précède.
- Dans l'algorithme de prim, le graphe doit être un graphe connecté tandis que celui de Kruskal peut également fonctionner sur des graphes déconnectés.
- il existe différents types de complexités et si on s'intéresse à la complexité temporelle, nous remarquons que pour un graphe  $G = (V, E)$ , l'algorithme de Prim a une complexité temporelle de  $O(V^2)$ , et la complexité temporelle de Kruskal est  $O(\log V)$ .

## 3.5 Implementation de l'application

Pour notre application, nous avons décidé d'implémenter l'algorithme de Prim.

Lors du lancement de cette dernière, un menu est proposé à l'utilisateur avec une liste de choix comme suit :

Bienvenue dans le projet du module Etuc qui a été élaboré par BENABED Anfel et AMINE Ishak.

Nous vous proposons les choix suivants :

1. Graphes valués
2. Algorithme de Prim
3. Sortir

Veuillez saisir votre choix s'il vous plait:

Figure 5: Menu.

### 3.5.1 Graphe valué

Dans le cas où l'utilisateur introduit la premiere option, il lui sera demandé d'introduire le nombre de sommets ainsi que le poids entre chaque sommet et à l'introduction du dernier poids, le graphe lui sera affiché.

Veillez introduire le nombre de sommets du graphe: 3

Veillez introduire:

```
Le poids entre le sommet 1 et 1: 0
Le poids entre le sommet 1 et 2: 12
Le poids entre le sommet 1 et 3: 3

Le poids entre le sommet 2 et 1: 15
Le poids entre le sommet 2 et 2: 1
Le poids entre le sommet 2 et 3: 22

Le poids entre le sommet 3 et 1: 13
Le poids entre le sommet 3 et 2: 23
Le poids entre le sommet 3 et 3: 2
```

Le graphe valué G :

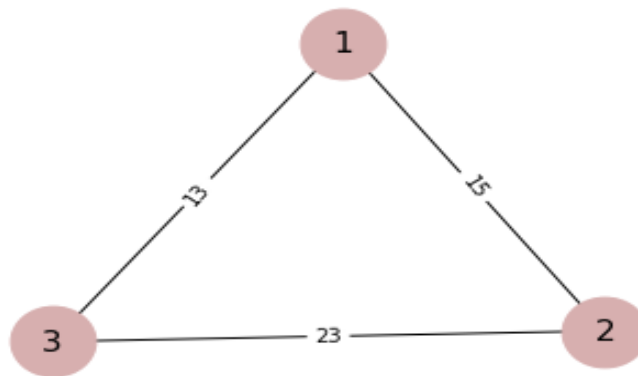


Figure 6: Graphe valué.

Il est à noter que l'utilisateur peut effectuer un autre choix s'il le souhaite.

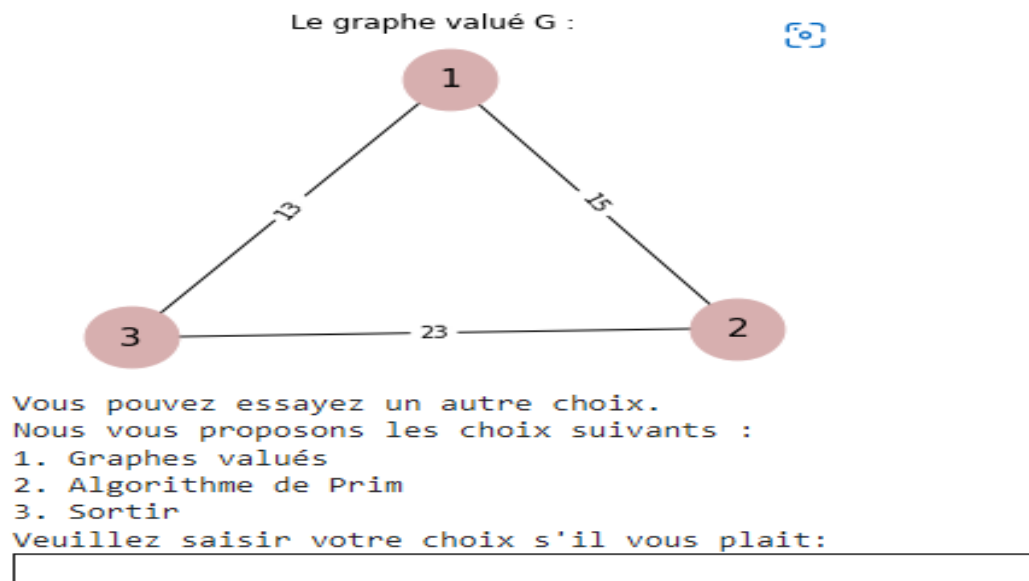


Figure 7: Autres choix - Graphe valué.

### 3.5.2 Algorithme de Prim

Dans le cas où l'utilisateur introduit la deuxième option, la même demande lui sera ainsi faite (introduire le nombre de sommets et le poids des arêtes), sauf que cette fois-ci, à la fermeture de la fenêtre du graphe, une nouvelle fenêtre contenant votre arbre couvrant de poids minimum vous sera affichée.



Veillez saisir votre choix s'il vous plait:

2

Veillez introduire le nombre de sommets du graphe: 3

Veillez introduire:

Le poids entre le sommet 1 et 1: 0  
Le poids entre le sommet 1 et 2: 12  
Le poids entre le sommet 1 et 3: 3

Le poids entre le sommet 2 et 1: 15  
Le poids entre le sommet 2 et 2: 1  
Le poids entre le sommet 2 et 3: 22

Le poids entre le sommet 3 et 1: 13  
Le poids entre le sommet 3 et 2: 23  
Le poids entre le sommet 3 et 3: 2

Le graphe initial G :

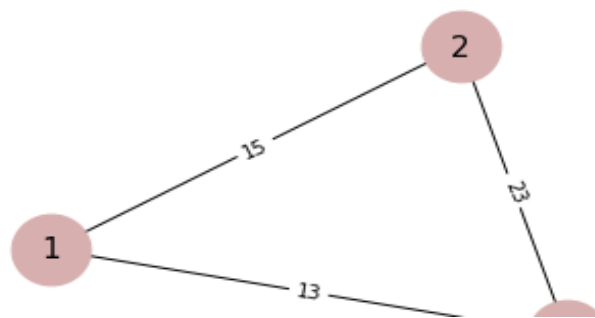


Figure 8: Implementation de Prim - Partie 01.

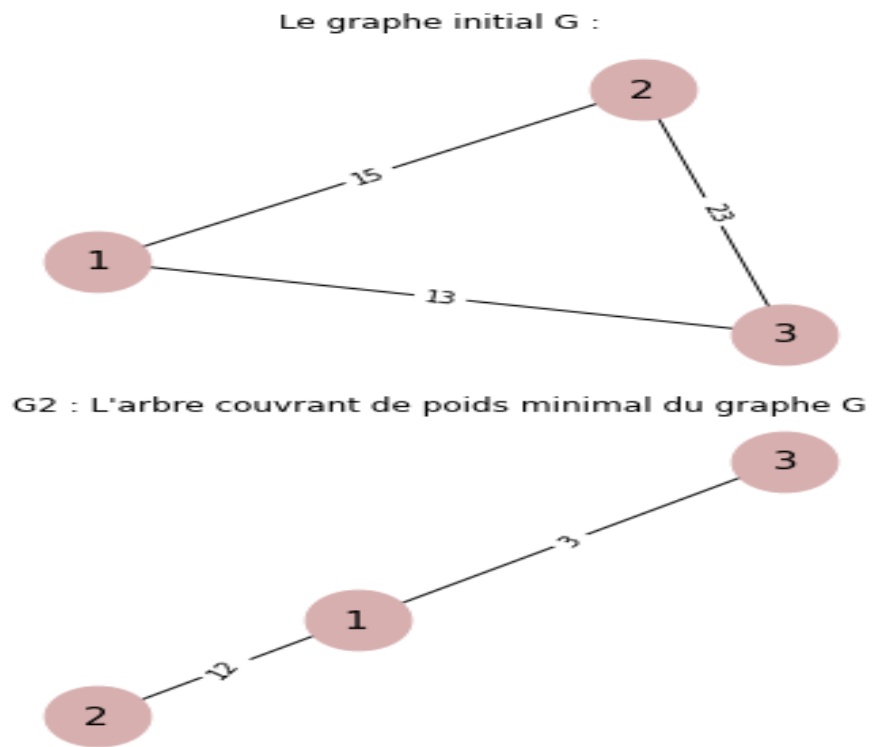

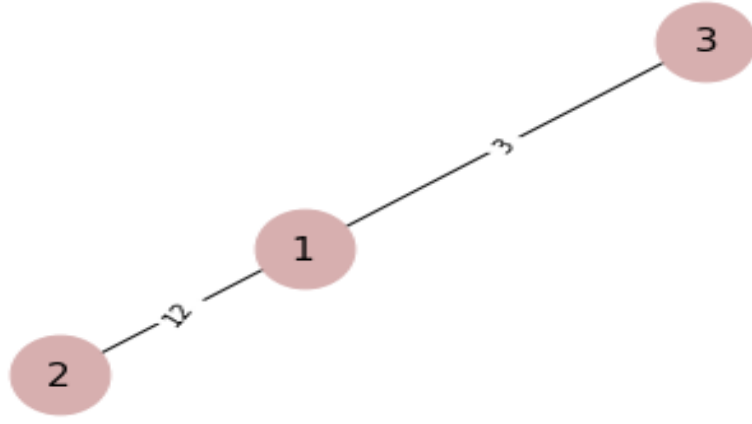


Figure 9: Implementation de Prim - Partie 02.

Il est à noter que l'utilisateur peut effectuer un autre choix s'il le souhaite.

G2 : L'arbre couvrant de poids minimal du graphe 



Vous pouvez essayer un autre choix.  
Nous vous proposons les choix suivants :

1. Graphes valués
2. Algorithme de Prim
3. Sortir

Veillez saisir votre choix s'il vous plait:

Figure 10: Autres choix - Prim.

### 3.5.3 Autres choix

Certes, il ne reste qu'un seul de choix visible sur le menu. Cependant, il reste en vérité un choix et un message d'erreur.

Dans le cas où l'utilisateur introduit le chiffre 03, un message de d'au-revoir lui sera affiché et le programme se terminera :

```
Nous vous proposons les choix suivants :  
1. Graphes valués  
2. Algorithme de Prim  
3. Sortir  
Veuillez saisir votre choix s'il vous plait:  
3  
Nous vous remercions pour votre visite. A bientôt!
```

Figure 11: Au revoir.

Et dans le cas où l'utilisateur saisie un chiffre qui n'est pas dans le menu (chiffre>3) alors un message d'erreur lui sera affiché ainsi que le menu qui lui permettra d'accéder à notre application :

```

.
```

---

```
Nous vous proposons les choix suivants :  
1. Graphes valués  
2. Algorithme de Prim  
3. Sortir  
Veuillez saisir votre choix s'il vous plait:  
5  
Nous n'avons pas de solution pour votre choix.  
Nous vous proposons les choix suivants :  
1. Graphes valués  
2. Algorithme de Prim  
3. Sortir  
Veuillez saisir votre choix s'il vous plait:  

```

Figure 12: Message d'erreur.

## References

- [1] [https://fr.wikipedia.org/wiki/Arbre\\_couvrant\\_de\\_poids\\_minimal](https://fr.wikipedia.org/wiki/Arbre_couvrant_de_poids_minimal)
- [2] [https://fr.wikipedia.org/wiki/Algorithme\\_de\\_Prim](https://fr.wikipedia.org/wiki/Algorithme_de_Prim)
- [3] [https://fr.wikipedia.org/wiki/Algorithme\\_de\\_Kruskal](https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal)
- [4] [http://ressources.unit.eu/cours/EnsR0tice/module\\_avance\\_thg\\_voo6/co/algoprime.html](http://ressources.unit.eu/cours/EnsR0tice/module_avance_thg_voo6/co/algoprime.html)
- [5] [https://perso.liris.cnrs.fr/samba\protect\discretionary{\char\hyphenchar\font}{\char\hyphenchar\font}{\char\hyphenchar\font}\ndojh.ndiaye/fichiers/App\\_Graphes.pdf](https://perso.liris.cnrs.fr/samba\protect\discretionary{\char\hyphenchar\font}{\char\hyphenchar\font}{\char\hyphenchar\font}\ndojh.ndiaye/fichiers/App_Graphes.pdf)
- [6] [http://yallouz.arie.free.fr/terminale\\_cours/graphes/graphes.php?page=g3](http://yallouz.arie.free.fr/terminale_cours/graphes/graphes.php?page=g3)
- [7] [https://fr.wikipedia.org/wiki/Python\\_\(langage\)](https://fr.wikipedia.org/wiki/Python_(langage))
- [8] [https://colab.research.google.com/notebooks/basic\\_features\\_overview.ipynb](https://colab.research.google.com/notebooks/basic_features_overview.ipynb)
- [9] <https://fr.esdifferent.com/difference\protect\discretionary{\char\hyphenchar\font}{\char\hyphenchar\font}{\char\hyphenchar\font}between\protect\discretionary{\char\hyphenchar\font}{\char\hyphenchar\font}{\char\hyphenchar\font}kruskal\protect\discretionary{\char\hyphenchar\font}{\char\hyphenchar\font}{\char\hyphenchar\font}and\protect\discretionary{\char\hyphenchar\font}{\char\hyphenchar\font}{\char\hyphenchar\font}prim#:~:text=Quelle%20est%20la%20diff%C3%A9rence%20entre%20l%27algorithme%20de%20Kruskal,ne%20soit%20pas%20bas%C3%A9%20sur%20la%20derni%C3%A8re%20%C3%A9tape>