

CALCULE DU NOMBRE DES POINTS  
D'ARTICULATION DANS UN GRAPHE NON  
ORIENTÉ

---

**RAPPORT DE TP2**

---

November 26, 2018

MAHAMDI Mohammed  
BOUKABENE Randa

# Table des Matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Approche de solution . . . . .	3
1.1.1	La solution triviale . . . . .	3
1.1.2	La solution optimale choisi . . . . .	4
<b>2</b>	<b>Implémentation de la solution</b>	<b>5</b>
2.1	Processus . . . . .	6
2.2	Calcul de la complexité . . . . .	8
2.3	Les outils utilisés . . . . .	8
<b>3</b>	<b>Conclusion</b>	<b>9</b>

# Table des Figures

2.1	Saisie du nombre neuds. . . . .	6
2.2	Saisie des données concernant le graphe . . . . .	7
2.3	Affichage du nombre des ponints d'articulation. . . . .	7
2.4	Affichage des points d'articulations dans un graphe (En couleur rouge). .	8

# Chapter 1

## Introduction

Dans un graphe non orienté, un sommet s'appelle un point d'articulation s'il est supprimé et que tous les arcs qui lui sont associés, le nombre de composantes connexes va augmenter.

Les points d'articulation représentent les vulnérabilités d'un réseau connecté . Ils sont utiles pour concevoir des réseaux fiables.

### 1.1 Approche de solution

#### 1.1.1 La solution triviale

Soit  $G$  un graphe à  $n$  sommets et  $m$  arêtes. Un algorithme trivial de complexité d'ordre  $O(nm)$  est le suivant :

$a =$  nombre de composantes connexes de  $G$  (déterminé à l'aide de DFS )

---

```
Pour chaque sommet  $v$  de  $V$  ayant des aretes incidentes
    retirer  $v$  de  $V$ 
     $b =$  nombre de composantes connexes de  $G$  une fois  $v$  elimine
    si  $b > a$ 
         $v$  est un PA de  $G$  # Afficher  $v$  ou incrementer un compteur
    remettre en place  $v$ 
```

---

Mais on ne veut pas une complexité de  $O(nm)$ .

### 1.1.2 La solution optimale choisi

L'idée est d'utiliser DFS (Depth First Search). Dans le parcours DFS, nous suivons les sommets sous forme d'arbre. un sommet  $u$  est le parent d'un autre sommet  $v$ , si  $v$  est découvert par  $u$  (évidemment,  $v$  est un adjacent de  $u$  dans le graphe). un sommet  $u$  est un point d'articulation si l'une des deux conditions suivantes est vraie:

1.  $u$  est la racine de DFS et il a au moins deux fils.
2.  $u$  n'est pas la racine de l'arbre DFS et  $u$  a un fils  $v$  tel qu'aucun sommet du sous-arbre qui a  $v$  comme racine avec  $v$  n'a pas d'arc de retour de l'un des parents de  $u$ .

# Chapter 2

## Implémentation de la solution

On a crée une classe Graph qui contient les attributs d'un graph (nombre des noeuds, le variable temps qu'on aura besoin au future et le nombre des points d'articulation) avec les deux fonctions suivantes :

---

```
find_Articulation_Points(self):
    DFS(self, node, visited, articulation_points, parent, low, discovery_time):
```

---

la première fonction **find\_Articulation\_Points()** sert à initialiser les paramètres , et qui appelle récursivement la deuxième fonction **DFS(self, node, visited, articulation\_points, parent, low, discovery\_time)**.

Dans la première fonction on initialise le nombre des points d'articulation et le temps de découvert à zéro, et on parcours l'ensemble des noeuds , si le noeud  $N$  n'est pas visité on exucute la fonction  $\text{DFS}(\dots, N)$  ( $N$  est un paramètre).

Pour la deuxième fonction est la fonction DFS mais elle est modifiée, pour garder la trace des parents, et savoirs'il existe un arc de retour (Back edge) ou non, pour vérifier les deux conditions cités Précédemment.

### Remarque:

Pour bien comprendre fonctionnement du programme , le code source est bien documenté.

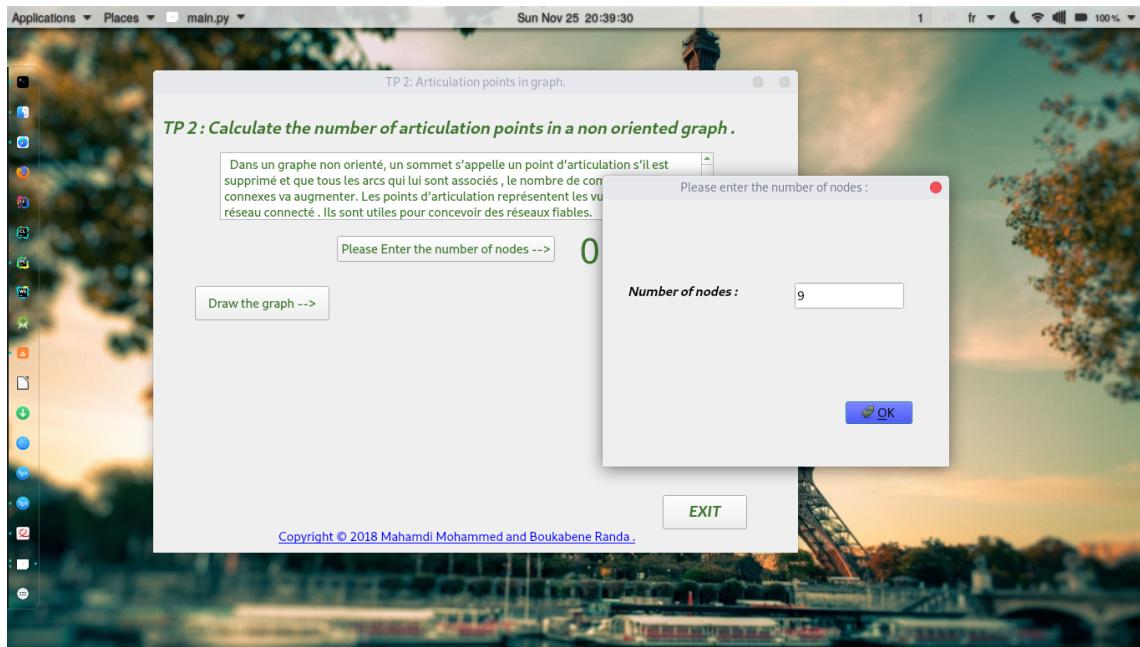
### Code source du projet:

Voici le lien vers le code source :

<https://github.com/MahamdiAmine/Articulation-points/>

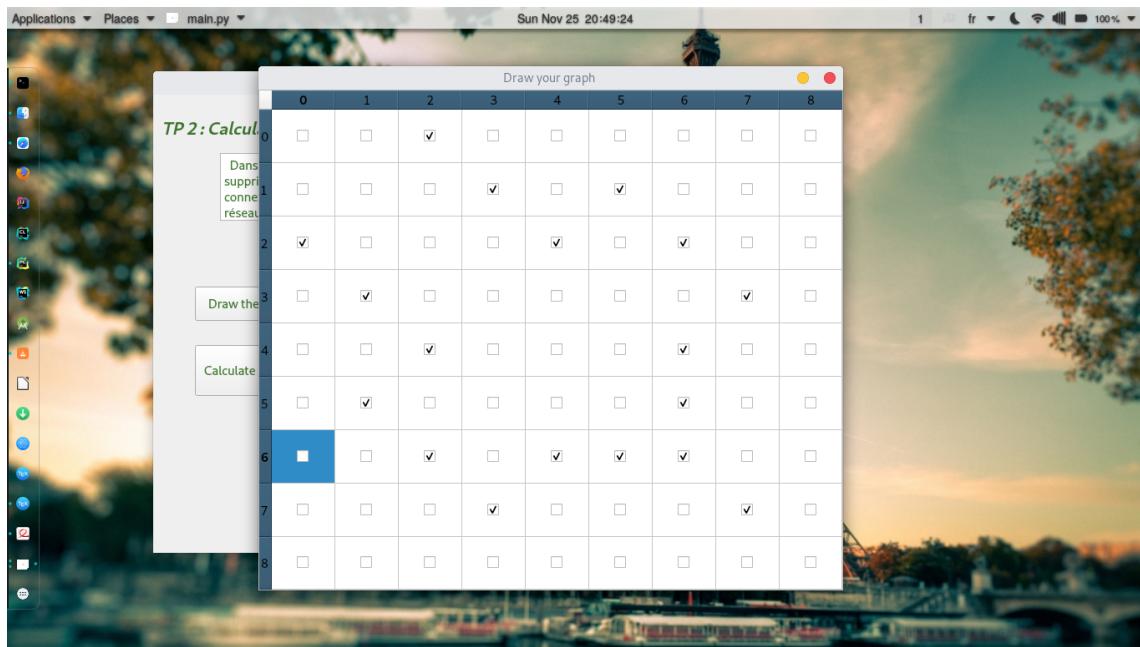
## 2.1 Processus

1. On lit le nombre du neuds :



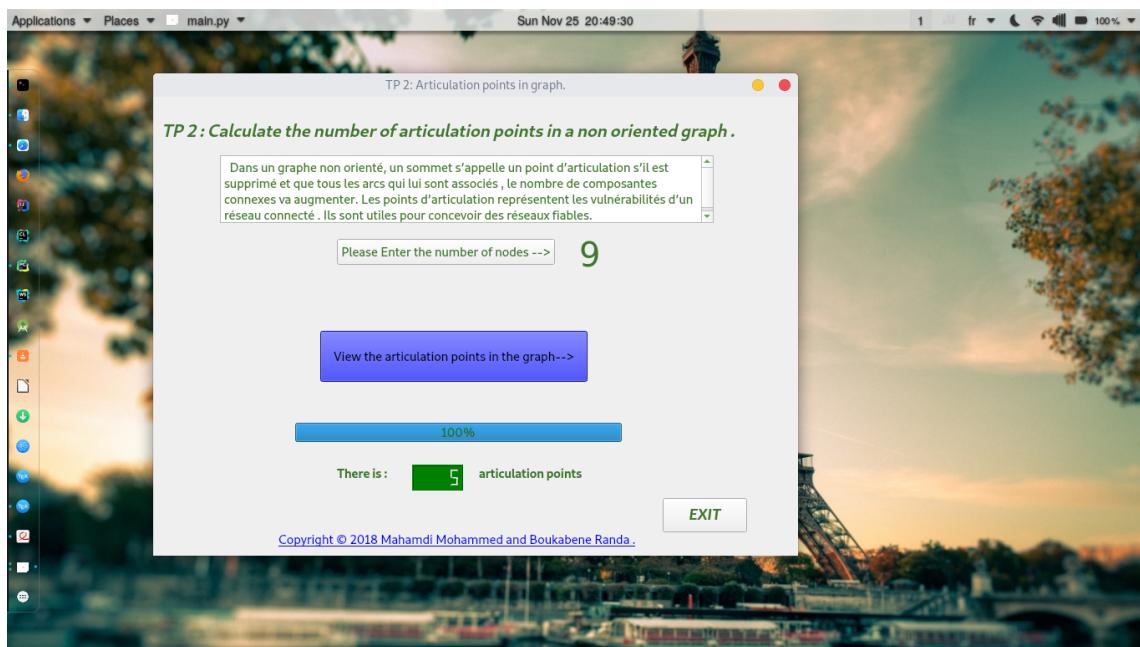
**Figure 2.1:** Saisie du nombre neuds.

2. On lit le graphe sous forme d'une matrice comme le montre la figure suivante :



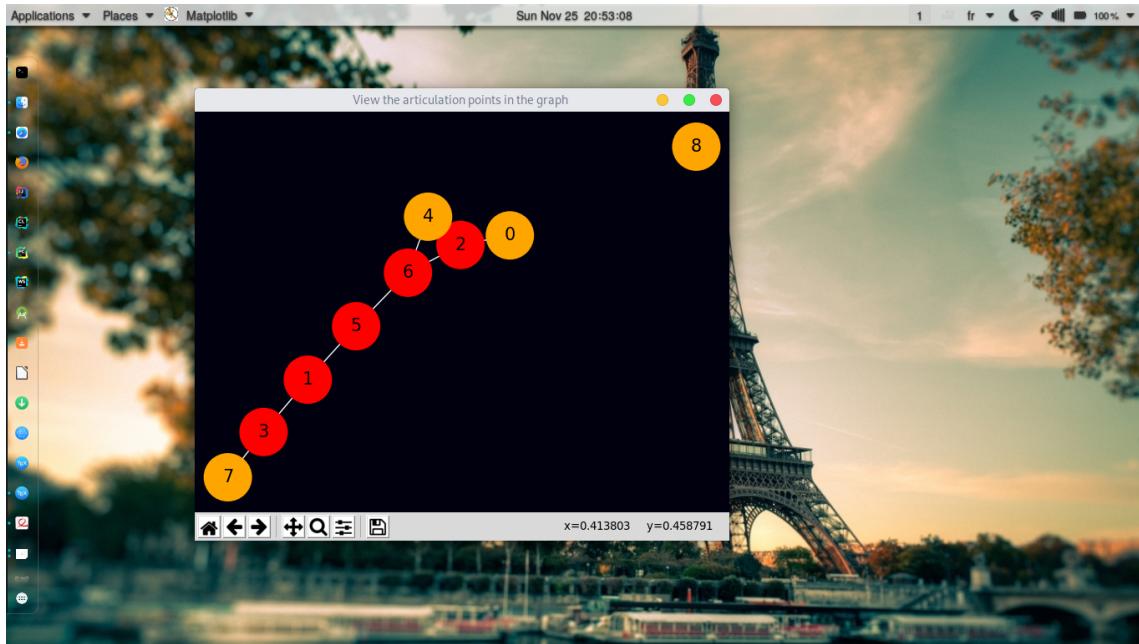
**Figure 2.2:** Saisie des données concernant le graphe

### 3. Calcule du nombre des points d'articulation:



**Figure 2.3:** Affichage du nombre des points d'articulation.

### 4. Affichage des points d'articulations dans un graphe (En couleur rouge) (pour la vérification ):



**Figure 2.4:** Affichage des points d'articulations dans un graphe (En couleur rouge).

## 2.2 Calcul de la complexité

La complexité de l'algorithme :  $O(n+m)$ .

**preuve** : On a Implémenté le graphe à l'aide d'une liste d'adjacence, tout les neuds sont visités une seule fois et chaque arc apparaîtra deux fois. Une fois dans la liste d'adjacence de chaque extrémité . Donc, la complexité finale sera :  
 $O(n) + O(2m) \approx O(n+m)$ .

## 2.3 Les outils utilisés

1. **Python3** : langage de programmation.
2. **PyQt 5** : L'interface graphique.
3. **networkx et matplotlib.pyplot** : Afficher le graphe et visualiser les points d'articulation avec les options(zoomer, enregistrer l'image, modifiée les couleurs du graphe, ou des arcs ...).
4. **Latex** : pour générer ce rapport.

# **Chapter 3**

## **Conclusion**

La théorie des graphes est la discipline mathématique et informatique qui étudie les graphes, lesquels sont des modèles abstraits de dessins de réseaux reliant des objets.

Les algorithmes élaborés pour résoudre des problèmes concernant les objets de cette théorie ont de nombreuses applications dans tous les domaines liés à la notion de réseau (réseau social, réseau informatique, télécommunications, etc.)