



# TUNISIA, THE DISCOVERY

# PRÉSENTÉ PAR

---



Fadi abdelkadir

# SOMMAIRE

---

1

INTRODUCTION

2

MODELISATION

3

ALGORITHMES

4

LA STRATEGIE OPTIMALE

5

MAPPING

6

CONTRAINTES

# CONTRIBUTION

---

1

THEORIE DES GRAPHERS  
LA MODELATION DE LA CARTE

2

COMPREHENSION ET  
IMPLEMENTATION

3

APPROCHE THEORIQUE

4

APPROCHE PRATIQUE

5

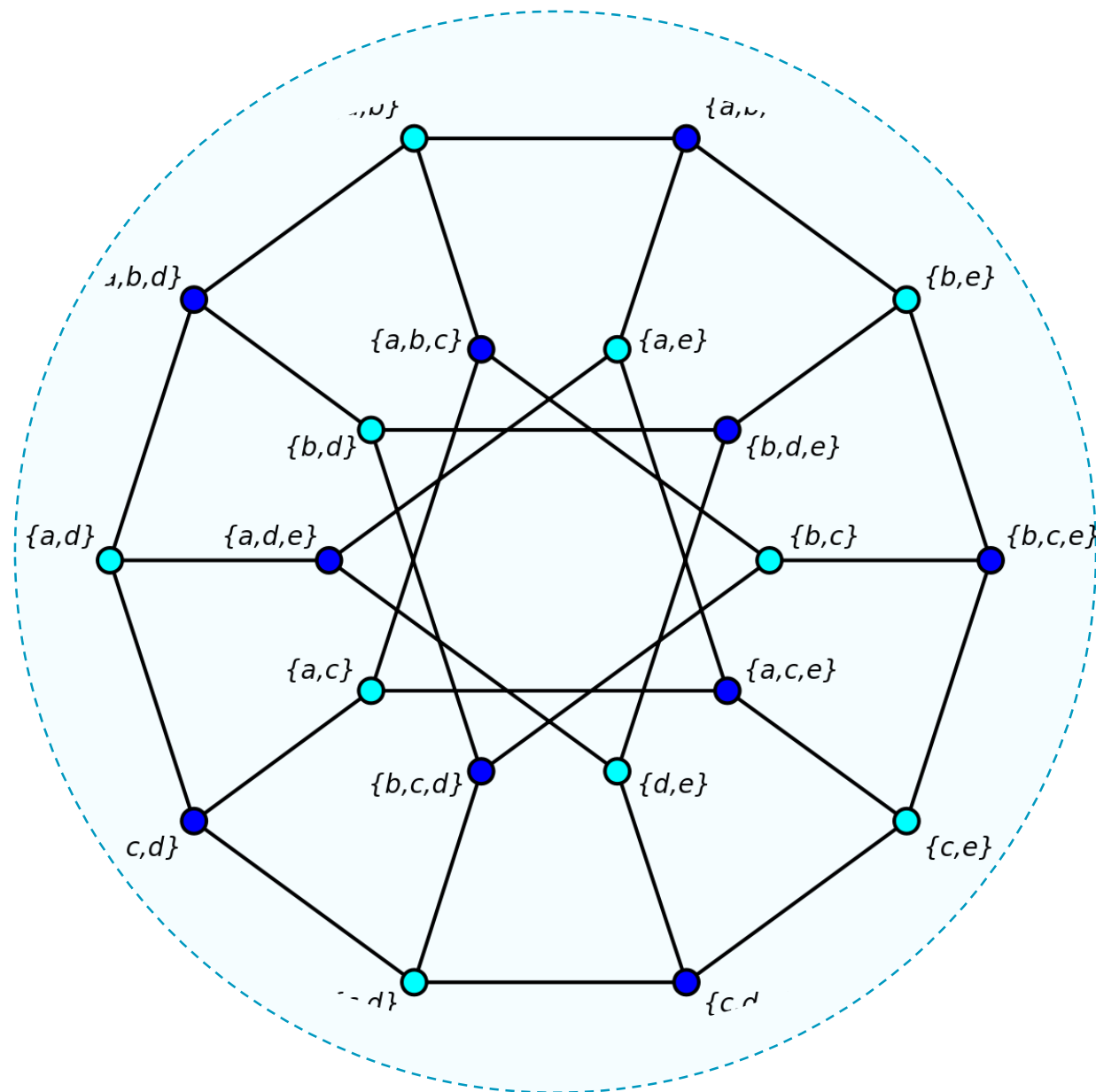
ETABLIR LE LIEN ENTRE  
A\* ET HAMILTON

6

ETUDE DE COUT



# Introduction



# INTRODUCTION

C'est un projet dont le but de promouvoir le tourisme de la Tunisie qui est l'un des secteurs importants du pays avec des circuits touristiques ,voyages organisés, excursions

On va vous faire découvrir la Tunisie dans toute sa splendeur  
Bienvenue dans un pays où vous ferez le plein du ciel bleu et de soleil, dont vous découvrirez le patrimoine exceptionnel et les traditions originales, les plus fameux monuments en Tunisie

# OBJECTIFS

## Les objectifs de Notre projet

Notre objectif : C'est de visiter les monuments de la Tunisie en passant une seule fois par chaque sommets







# PROBLÉMATIQUE

---

Comment effectuer un passage touristique par tous les monuments une et une seule fois en passant par le plus court chemin



# MODULE ET PROGRAMME UTILISÉ

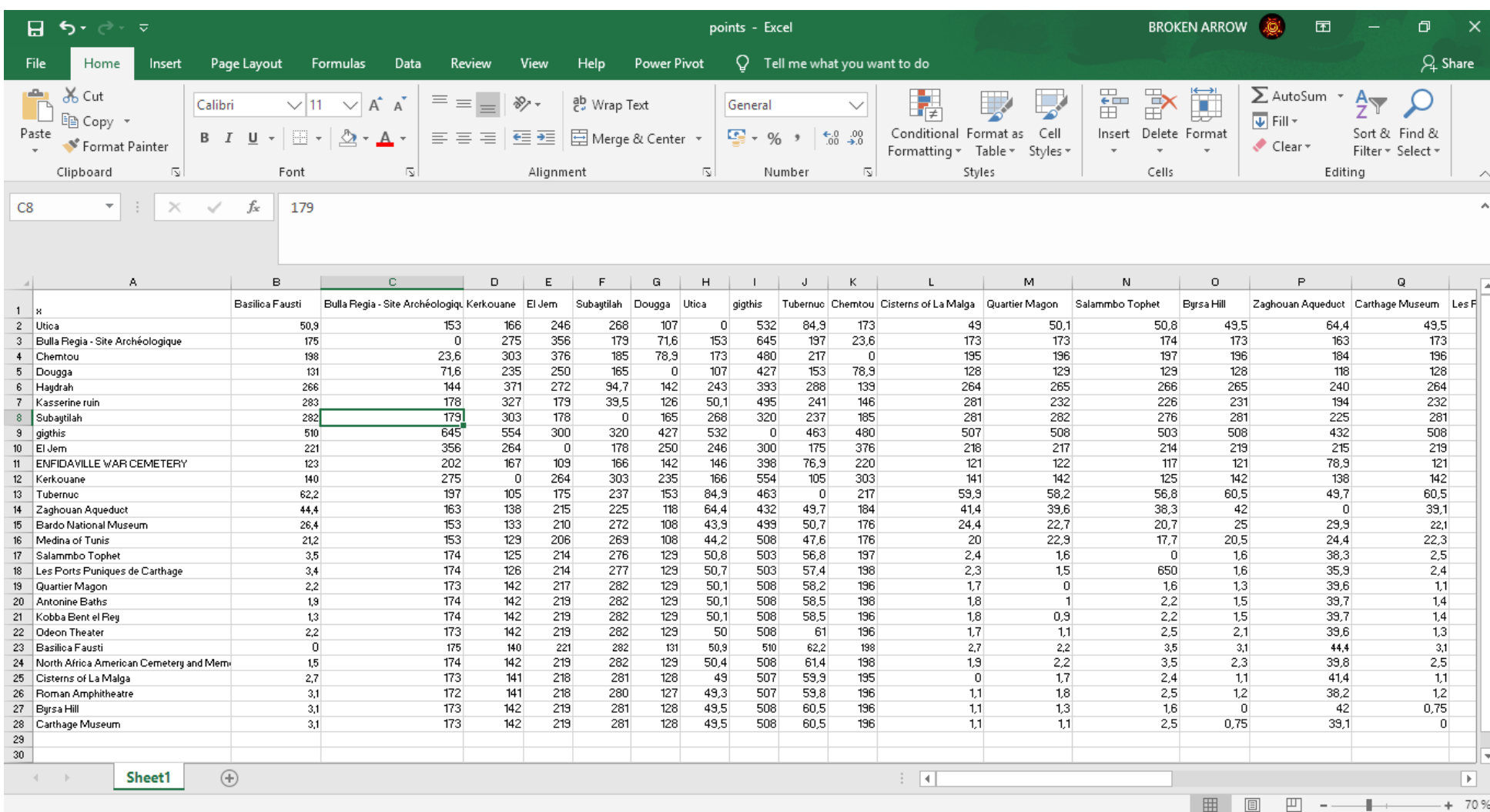
---

Numpy  
Pandas  
Matplotlib.[pyplot](#)  
Networkx  
Gmplot  
Folium

Python  
Google earth  
Ms Excel  
gephi

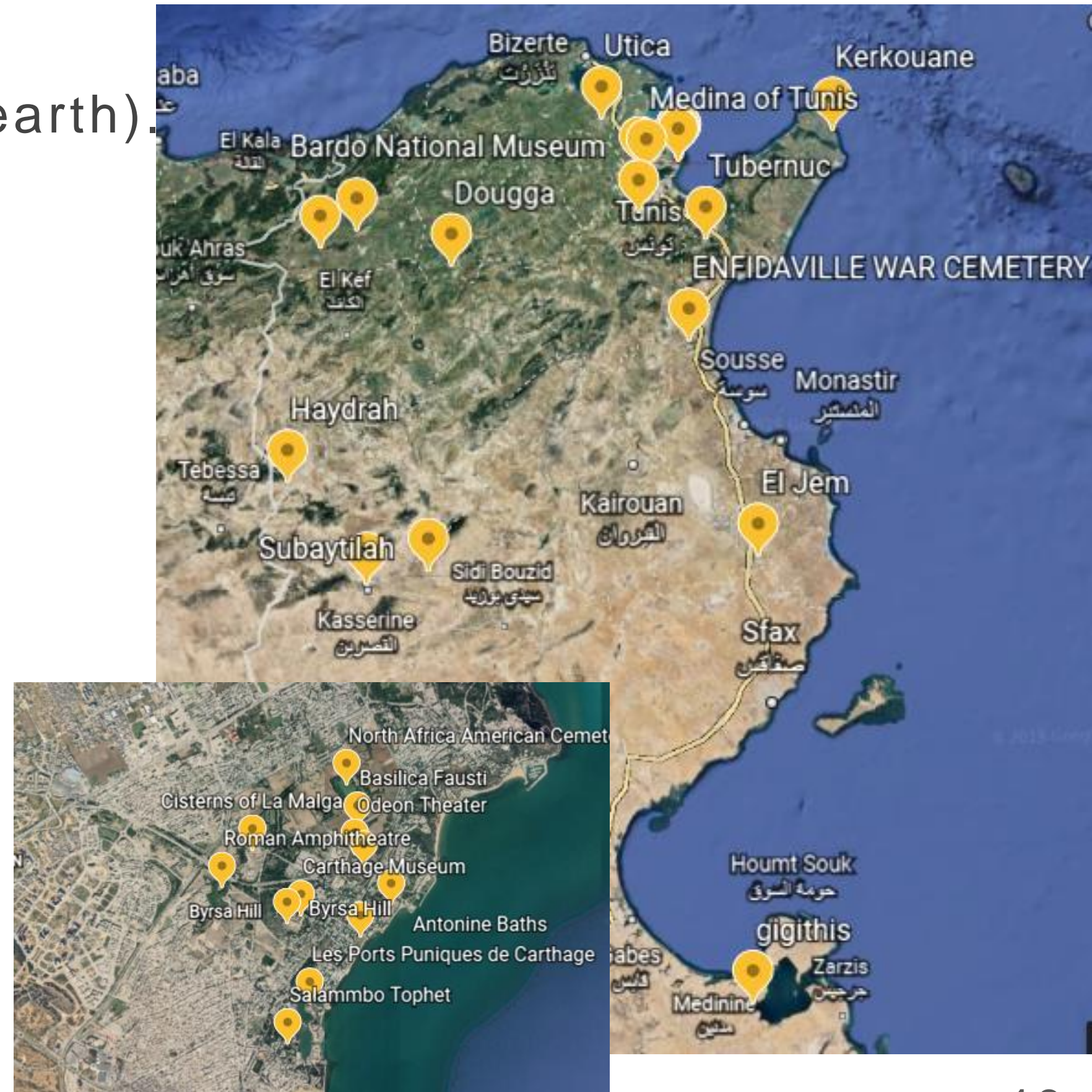
# Modélisation: carte

- Utilisation d'une application (google earth).
- Mesure de la latitude et la longitude
- Recherche des distances et l'implémentation dans un tableau (calcul à la main et utilisation de google maps)



The screenshot shows an Excel spreadsheet titled 'points - Excel'. The table contains distances between 28 locations. The locations are listed in the first column (A1 to A28) and the first row (B1 to Q1). The distance values are in the cells from B2 to Q28. The value 179 is entered in cell C8.

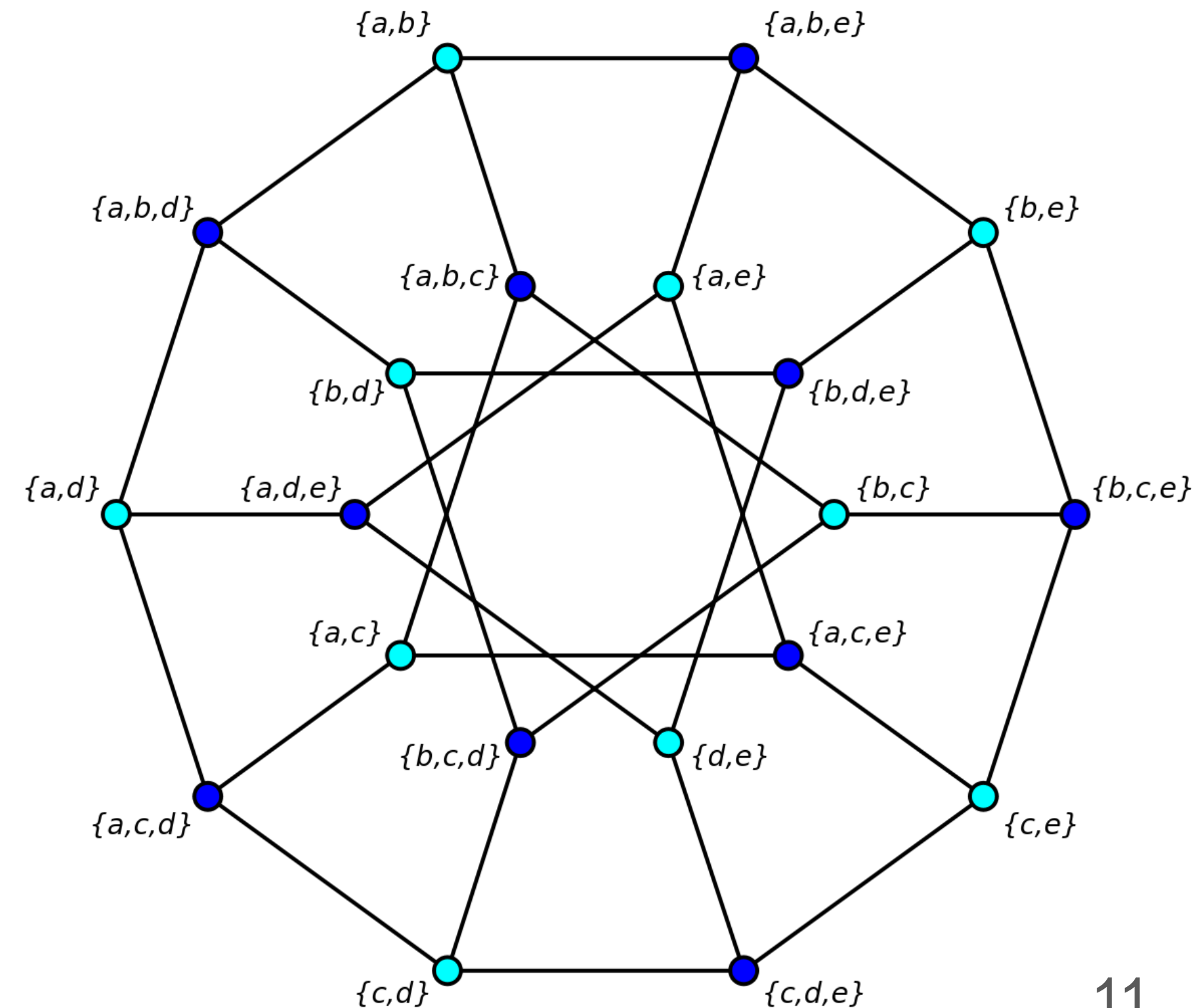
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	x	Basilica Fausti	Bulla Regia - Site Archéologique	Kerkouane	El Jem	Subaytilah	Dougga	Utica	gigithis	Tubernuc	Chemtou	Cisterns of La Malga	Quartier Magon	Salamambo Tophet	Byrsa Hill	Zaghoun Aqueduct	Carthage Museum	Les F
2	Utica	50,9	153	166	246	268	107	0	532	84,9	173	49	50,1	50,8	49,5	64,4	49,5	
3	Bulla Regia - Site Archéologique	175	0	275	356	179	71,6	153	645	197	23,6	173	173	174	173	163	173	
4	Chemtou	198	23,6	303	376	185	78,9	173	480	217	0	195	196	197	196	184	196	
5	Dougga	131	71,6	235	250	165	0	107	427	153	78,9	128	129	129	128	118	128	
6	Haydrah	266	144	371	272	94,7	142	243	393	288	139	264	265	266	265	240	264	
7	Kasserine ruin	283	178	327	179	39,5	126	50,1	495	241	146	281	232	226	231	194	232	
8	Subaytilah	282	179	303	178	0	165	268	320	237	185	281	282	276	281	225	281	
9	gigithis	510	645	554	300	320	427	532	0	463	480	507	508	503	508	432	508	
10	El Jem	221	356	264	0	178	250	246	300	175	376	218	217	214	219	215	219	
11	ENFIDAVILLE WAR CEMETERY	123	202	167	109	166	142	146	398	76,9	220	121	122	117	121	78,9	121	
12	Kerkouane	140	275	0	264	303	235	166	554	105	303	141	142	125	142	138	142	
13	Tubernuc	62,2	197	105	175	237	153	84,9	463	0	217	59,9	58,2	58,8	60,5	49,7	60,5	
14	Zaghoun Aqueduct	44,4	163	138	215	225	118	64,4	432	49,7	184	41,4	39,6	38,3	42	0	39,1	
15	Bardo National Museum	26,4	153	133	210	272	108	43,9	499	50,7	176	24,4	22,7	20,7	25	29,9	22,1	
16	Medina of Tunis	212	153	129	206	269	108	44,2	508	47,6	176	20	22,9	17,7	20,5	24,4	22,3	
17	Salamambo Tophet	3,5	174	125	214	276	129	50,8	503	58,8	197	2,4	1,6	0	1,6	38,3	2,5	
18	Les Ports Puniques de Carthage	3,4	174	126	214	277	129	50,7	503	57,4	198	2,3	1,5	650	1,6	35,9	2,4	
19	Quartier Magon	2,2	173	142	217	282	129	50,1	508	58,2	196	1,7	0	1,6	1,3	39,6	1,1	
20	Antonine Baths	1,9	174	142	219	282	129	50,1	508	58,5	198	1,8	1	2,2	1,5	39,7	1,4	
21	Kobba Bent el Rieg	1,3	174	142	219	282	129	50,1	508	58,5	196	1,8	0,9	2,2	1,5	39,7	1,4	
22	Odeon Theater	2,2	173	142	219	282	129	50	508	61	196	1,7	1,1	2,5	2,1	39,6	1,3	
23	Basilica Fausti	0	175	140	221	282	131	60,9	510	62,2	198	2,7	2,2	3,5	3,1	44,4	3,1	
24	North Africa American Cemetery and Mem	1,5	174	142	219	282	129	50,4	508	61,4	198	1,9	2,2	3,5	2,3	39,8	2,5	
25	Cisterns of La Malga	2,7	173	141	218	281	128	49	507	59,9	195	0	1,7	2,4	1,1	41,4	1,1	
26	Roman Amphitheatre	3,1	172	141	218	280	127	49,3	507	59,8	196	1,1	1,8	2,5	1,2	38,2	1,2	
27	Byrsa Hill	3,1	173	142	219	281	128	49,5	508	60,5	196	1,1	1,3	1,6	0	42	0,75	
28	Carthage Museum	3,1	173	142	219	281	128	49,5	508	60,5	196	1,1	1,1	2,5	0,75	39,1	0	



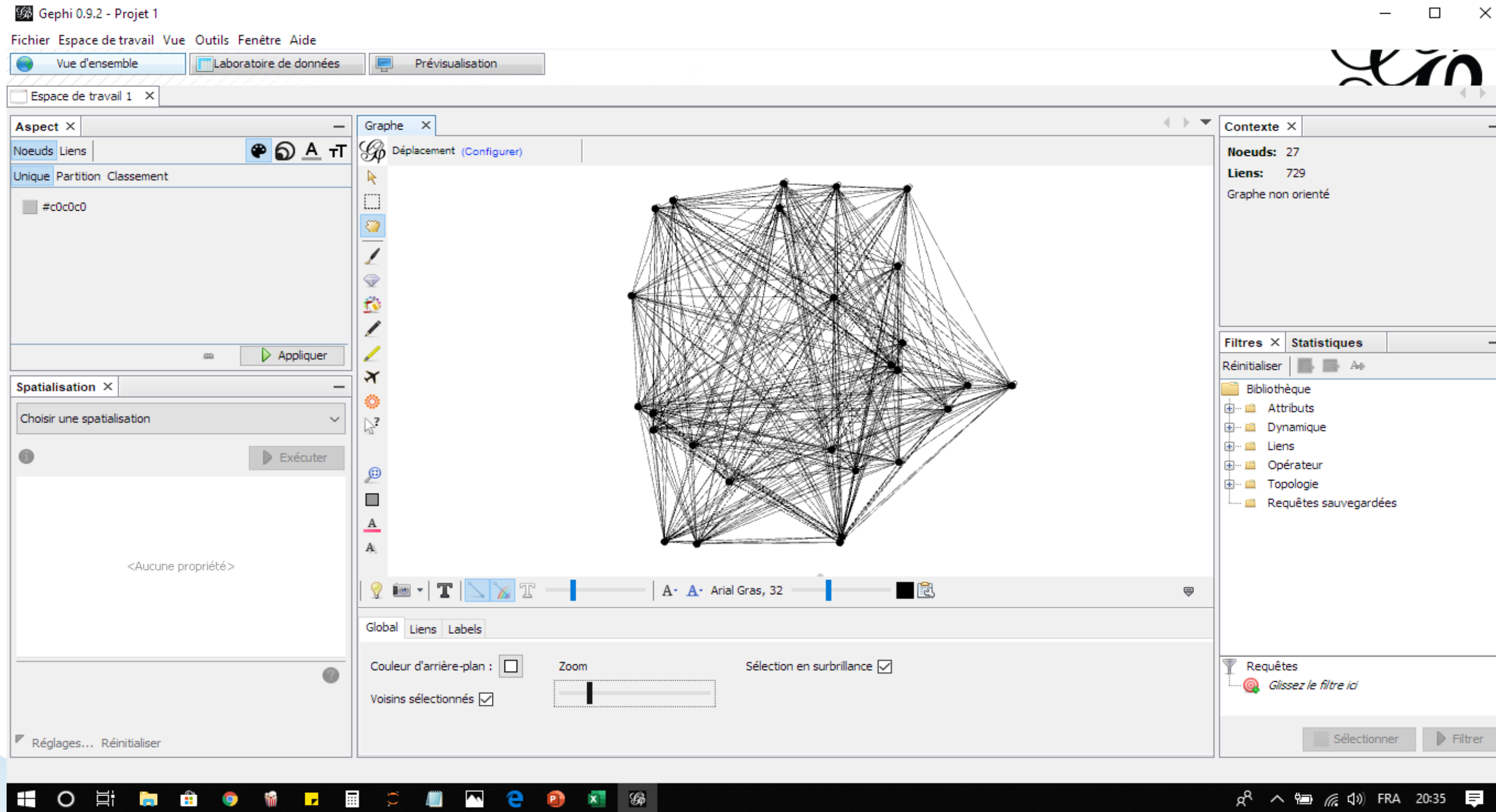


# THEORIE DE GRAPHE

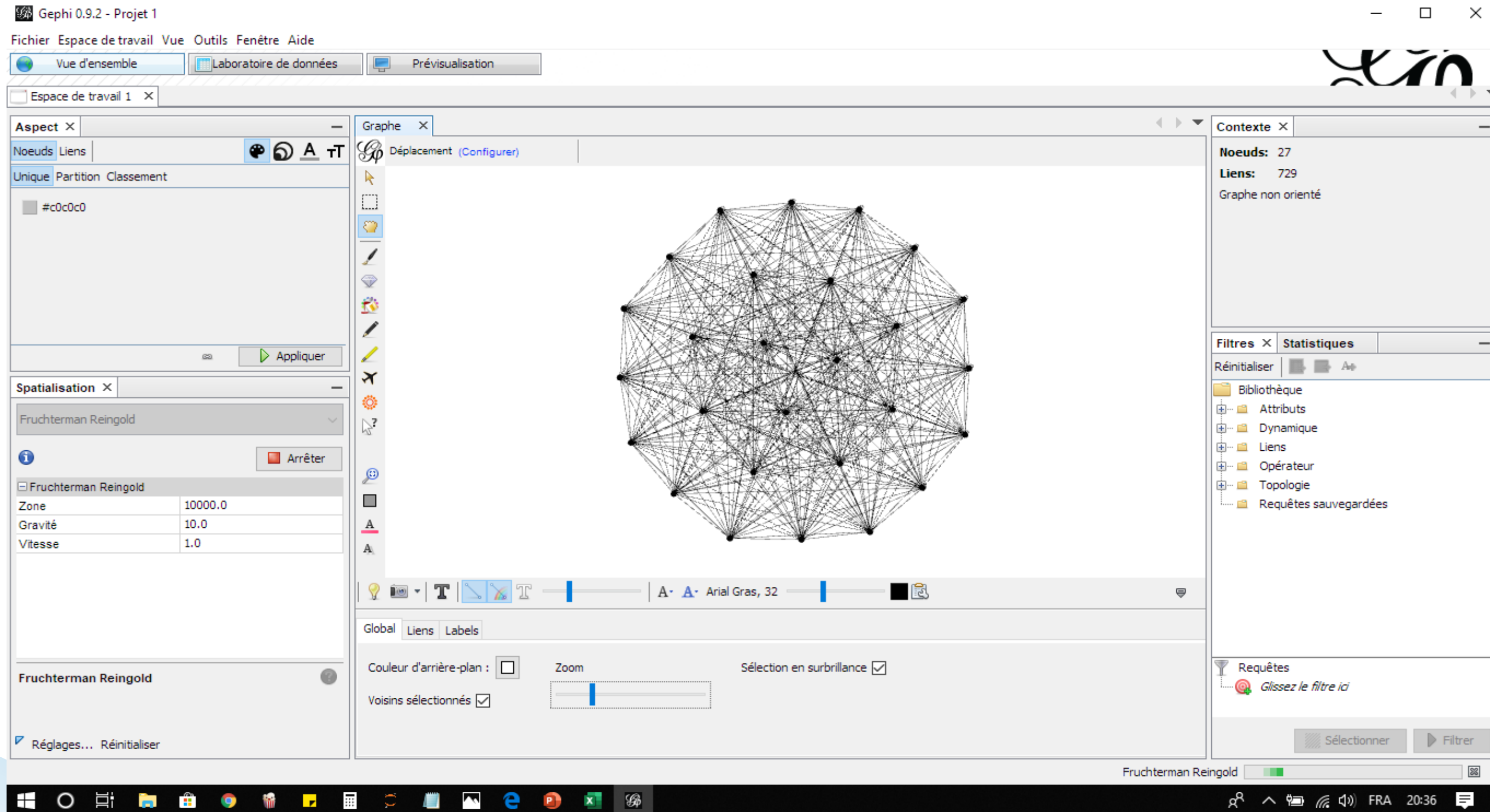
Un graphe est un modèle qui étudie un ensemble d'objets liés par des relations: un ensemble de sommets et d'arrêtes. Pour représenter des relations entre les éléments dont il peut être appliqué dans l'informatique mathématique même pour les réseaux



# Modélisation : graphe



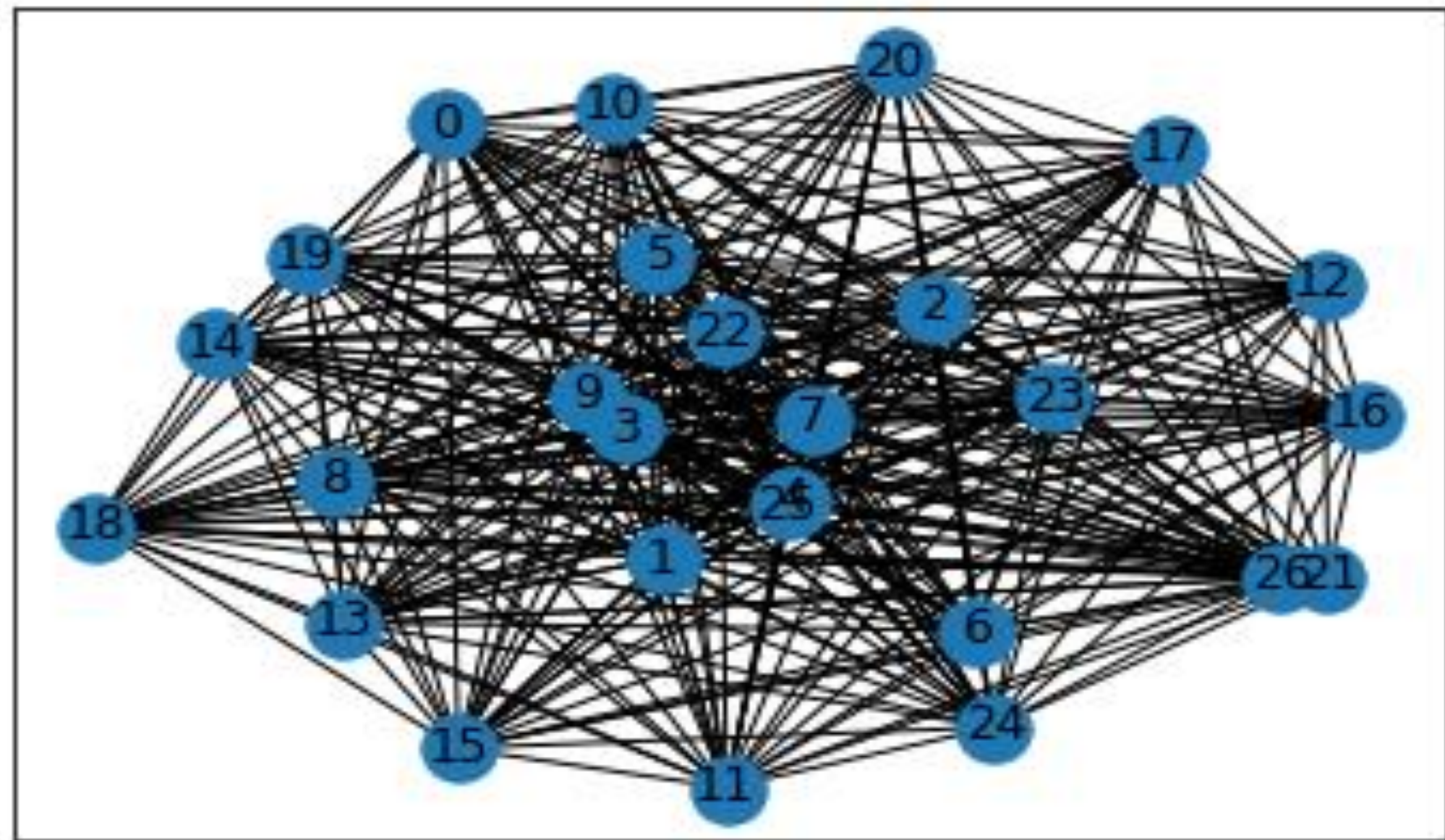
# Modélisation : graphe





# Modélisation : graphe

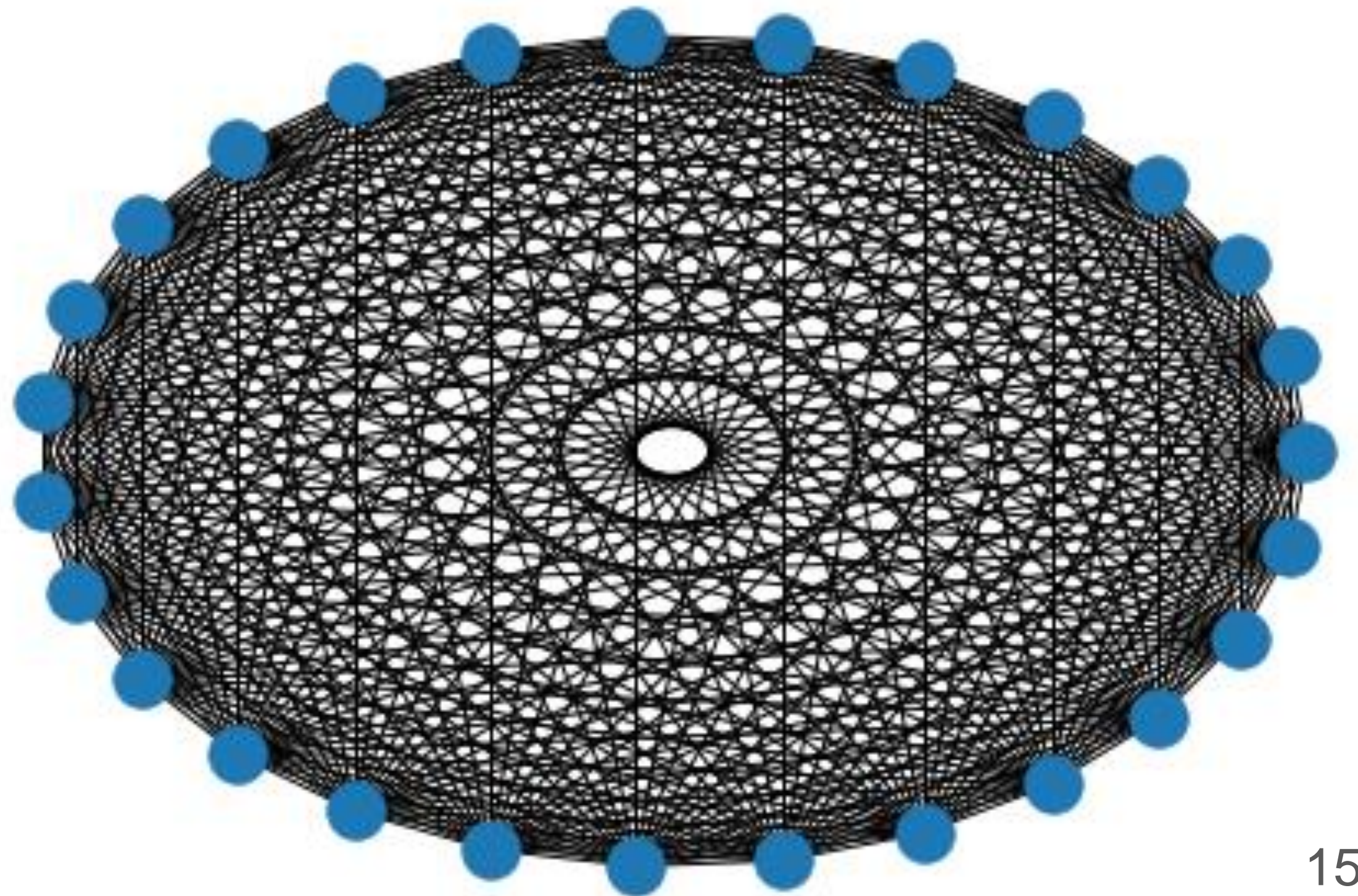
`nx.draw_networkx(g)`





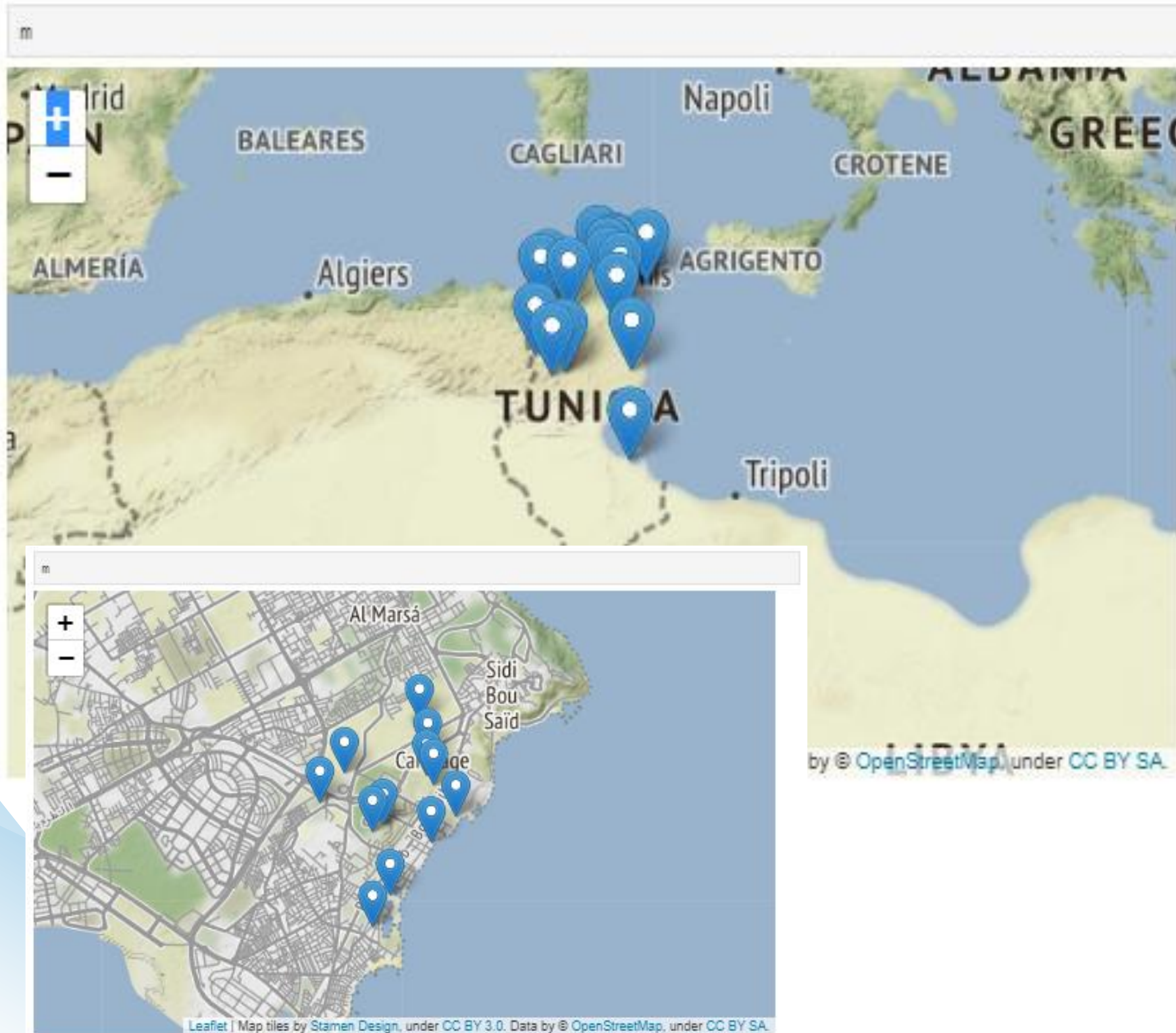
# Modélisation : graphe

`nx.draw_circular(g)`





# MAPPING



```
import folium
```

```
m = folium.Map(location=[33.8439408, 9.400138],tiles='stamen terrain ')
```

```
folium.Marker( location = [36.8558694,10.3151931] , popup = "Roman Amphitheatre" , tooltip  
folium.Marker( location = [36.8654349,10.3297122] , popup = "North Africa American Cemetery  
folium.Marker( location = [36.1338883,10.3742362] , popup = "ENFIDAVILLE WAR CEMETERY" , to  
folium.Marker( location = [36.42329105,9.21870005] , popup = "Dougga" , tooltip = "click he  
folium.Marker( location = [36.9604541,11.0799265] , popup = "Kerkouane" , tooltip = "click  
folium.Marker( location = [36.5596925,8.7541243] , popup = "Bulla Regia - Site Archéologiqu  
folium.Marker( location = [36.489542,8.5769602] , popup = "Chemtou" , tooltip = "click here  
folium.Marker( location = [35.29519525,10.7063484] , popup = "El Jem" , tooltip = "click he  
folium.Marker( location = [35.17026285,8.82476795] , popup = "Kasserine ruin" , tooltip = "  
folium.Marker( location = [35.56696765,8.4438944] , popup = "Haydrah" , tooltip = "click he  
folium.Marker( location = [35.2315127,9.12433965] , popup = "Subaytilah" , tooltip = "click  
folium.Marker( location = [33.5398386,10.67358255] , popup = "gigthis" , tooltip = "click h  
folium.Marker( location = [36.859336,10.3187431] , popup = "Cisterns of La Malga" , tooltip  
folium.Marker( location = [36.8524951,10.3227768] , popup = "Byrsa Hill" , tooltip = "click  
folium.Marker( location = [36.8532268,10.3243991] , popup = "Carthage Museum" , tooltip = "  
folium.Marker( location = [36.8512445,10.3313709] , popup = "Quartier Magon" , tooltip = "c  
folium.Marker( location = [36.8589587,10.3306693] , popup = "Odeon Theater" , tooltip = "cl  
folium.Marker( location = [36.8542306,10.3349601] , popup = "Antonine Baths" , tooltip = "c  
folium.Marker( location = [36.8578685,10.3317216] , popup = "Kobba Bent el Rey" , tooltip =  
folium.Marker( location = [336.845036,10.3254533] , popup = "Les Ports Puniques de Carthage  
folium.Marker( location = [36.8412563,10.3228645] , popup = "Salamambo Tophet" , tooltip = "  
folium.Marker( location = [36.7985292,10.16922265] , popup = "Medina of Tunis" , tooltip =  
folium.Marker( location = [36.8094589,10.1340362] , popup = "Bardo National Museum" , toolt  
folium.Marker( location = [36.8614744,10.3309952] , popup = "Basilica Fausti" , tooltip = "  
folium.Marker( location = [36.6382335,10.1295877] , popup = "Zaghuan Aqueduct" , tooltip =  
folium.Marker( location = [37.01646735,9.94475065] , popup = "Utica" , tooltip = "click her  
folium.Marker( location = [36.532864,10.4567689] , popup = "Tubernuc" , tooltip = "click he  
folium.Marker( location = [36.845036,10.3254533] , popup = "Les Ports Puniques de Carthage"
```

# Application de l'algorithme: HAMILTONIEN

En mathématiques, dans le cadre de **la théorie des graphes**, un chemin **hamiltonien** d'un graphe orienté ou non orienté est un chemin qui passe par tous les sommets une fois et une seule.

Un **cycle hamiltonien** est un chemin hamiltonien qui est un cycle.

Un **graphe hamiltonien** est un graphe qui possède un cycle hamiltonien.

**Théorème** — Un graphe simple à  $n$  sommets ( $n \geq 3$ ) dont chaque sommet est au moins de degré  $\frac{n}{2}$  est hamiltonien.

=> Chaque sommet a 26 degré : **VERIFIÉ**

# Application de l'algorithme: HAMILTONIEN

```
dictionnaire = {}  
for line in nx.generate_adjlist(G):  
    print(line)  
    line = line.split()  
    dictionnaire[line[0]] = line[1:]  
print(dictionnaire)
```

Basilica\_Fausti Bulla\_Regia\_-\_Site\_Archéologique Kerkouane El\_Jem Subaytilah Dougga Utica gigth  
is Tubernuc Chemtou Cisterns\_of\_La\_Malga Quartier\_Magon Salammbo\_Tophet Byrsa\_Hill Zaghouan\_Aqu  
educt Carthage\_Museum Les\_Ports\_Puniques\_de\_Carthage Roman\_Amphitheatre Odeon\_Theater Antonine\_  
Baths Kobba\_Bent\_el\_Rey Medina\_of\_Tunis Kasserine\_ruin ENFIDAVILLE\_WAR\_CEMETERY North\_Africa\_Am  
erican\_Cemetery\_and\_Memorial Haydrah Bardo\_National\_Museum

Bulla\_Regia\_-\_Site\_Archéologique Kerkouane El\_Jem Subaytilah Dougga Utica gigthis Tubernuc Chem  
tou Cisterns\_of\_La\_Malga Quartier\_Magon Salammbo\_Tophet Byrsa\_Hill Zaghouan\_Aqueduct Carthage\_M  
useum Les\_Ports\_Puniques\_de\_Carthage Roman\_Amphitheatre Odeon\_Theater Antonine\_Baths Kobba\_Bent  
\_el\_Rey Medina\_of\_Tunis Kasserine\_ruin ENFIDAVILLE\_WAR\_CEMETERY North\_Africa\_American\_Cemetery\_  
and\_Memorial Haydrah Bardo\_National\_Museum

Kerkouane El\_Jem Subaytilah Dougga Utica gigthis Tubernuc Chemtou Cisterns\_of\_La\_Malga Quartier  
\_Magon Salammbo\_Tophet Byrsa\_Hill Zaghouan\_Aqueduct Carthage\_Museum Les\_Ports\_Puniques\_de\_Carth  
age Roman\_Amphitheatre Odeon\_Theater Antonine\_Baths Kobba\_Bent\_el\_Rey Medina\_of\_Tunis Kasserine  
\_ruin ENFIDAVILLE\_WAR\_CEMETERY North\_Africa\_American\_Cemetery\_and\_Memorial Haydrah Bardo\_Nation  
al\_Museum

El\_Jem Subaytilah Dougga Utica gigthis Tubernuc Chemtou Cisterns\_of\_La\_Malga Quartier\_Magon Sal  
ammbo\_Tophet Byrsa\_Hill Zaghouan\_Aqueduct Carthage\_Museum Les\_Ports\_Puniques\_de\_Carthage Roman\_  
Amphitheatre Odeon\_Theater Antonine\_Baths Kobba\_Bent\_el\_Rey Medina\_of\_Tunis Kasserine\_ruin ENFI  
DAVILLE\_WAR\_CEMETERY North\_Africa\_American\_Cemetery\_and\_Memorial Haydrah Bardo\_National\_Museum

```
def hamilton(G, size, pt, path=[]):  
    print('hamilton called with pt={}, path={}'.format(pt, path))  
    if pt not in set(path):  
        path.append(pt)  
        if len(path)==size:  
            return path  
        for pt_next in G.get(pt, []):  
            res_path = [i for i in path]  
            candidate = hamilton(G, size, pt_next, res_path)  
            if candidate is not None: # skip Loop or dead end  
                return candidate  
        print('path {} is a dead end'.format(path))  
    else:  
        print('pt {} already in path {}'.format(pt, path))  
    # Loop or dead end, None is implicitly returned
```

```
hamilton(dictionnaire,"Roman_Amphitheatre","Kerkouane")
```

hamilton called with pt=Kerkouane, path=[]  
hamilton called with pt=El\_Jem, path=['Kerkouane']  
hamilton called with pt=Subaytilah, path=['Kerkouane', 'El\_Jem']  
hamilton called with pt=Dougga, path=['Kerkouane', 'El\_Jem', 'Subaytilah']  
hamilton called with pt=Utica, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga']  
hamilton called with pt=gigthis, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga', 'Utica']  
hamilton called with pt=Tubernuc, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga', 'Utica',  
'gigthis']  
hamilton called with pt=Chemtou, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga', 'Utica',  
'gigthis', 'Tubernuc']  
hamilton called with pt=Cisterns\_of\_La\_Malga, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga',  
'Utica', 'gigthis', 'Tubernuc', 'Chemtou']  
hamilton called with pt=Quartier\_Magon, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga', 'U  
tica', 'gigthis', 'Tubernuc', 'Chemtou', 'Cisterns\_of\_La\_Malga']  
hamilton called with pt=Salammbo\_Tophet, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga',  
'Utica', 'gigthis', 'Tubernuc', 'Chemtou', 'Cisterns\_of\_La\_Malga', 'Quartier\_Magon']  
hamilton called with pt=Byrsa\_Hill, path=['Kerkouane', 'El\_Jem', 'Subaytilah', 'Dougga', 'Utic  
a', 'gigthis', 'Tubernuc', 'Chemtou', 'Cisterns\_of\_La\_Malga', 'Quartier\_Magon', 'Salammbo\_Tophet']

# Application de l'algorithme: HAMILTONIEN

```
class Graph():
    def __init__(self, vertices):
        self.graph = [[0 for column in range(vertices)]
                       for row in range(vertices)]
        self.V = vertices

    ''' Check if this vertex is an adjacent vertex
        of the previously added vertex and is not
        included in the path earlier '''
    def isSafe(self, v, pos, path):
        # Check if current vertex and last vertex
        # in path are adjacent
        if self.graph[ path[pos-1] ][v] == 0:
            return False

        # Check if current vertex not already in path
        for vertex in path:
            if vertex == v:
                return False

        return True

    # A recursive utility function to solve
    # hamiltonian cycle problem
    def hamCycleUtil(self, path, pos):

        # base case: if all vertices are
        # included in the path
        if pos == self.V:
            # Last vertex must be adjacent to the
            # first vertex in path to make a cycle
            if self.graph[ path[pos-1] ][ path[0] ] == 1:
                return True
            else:
                return False
```

```
        # Try different vertices as a next candidate
        # in Hamiltonian Cycle. We don't try for 0 as
        # we included 0 as starting point in hamCycle()
        for v in range(1, self.V):

            if self.isSafe(v, pos, path) == True:

                path[pos] = v

                if self.hamCycleUtil(path, pos+1) == True:
                    return True

                # Remove current vertex if it doesn't
                # lead to a solution
                path[pos] = -1

        return False
    def hamCycle(self):
        path = [-1] * self.V

        ''' Let us put vertex 0 as the first vertex
            in the path. If there is a Hamiltonian Cycle,
            then the path can be started from any point
            of the cycle as the graph is undirected '''
        path[0] = 0

        if self.hamCycleUtil(path, 1) == False:
            print ("Solution does not exist\n")
            return False

        self.printSolution(path)
        return True
    def printSolution(self, path):
        print ("Solution Exists: Following is one Hamiltonian Cycle")
        for vertex in path:
            print (vertex)
```

# Résultat de l'algorithme:

## HAMILTONIEN

```
nx.algorithms.tournament.hamiltonian_path(GG)
```

Fair le tour : **validé**

Passer par le monuments  
une et une seul fois : **validé**

Prend le plus court chemin :  
**non validé**

```
['Kobba_Bent_el_Rey',  
 'Zaghouan_Aqueduct',  
 'Carthage_Museum',  
 'Medina_of_Tunis',  
 'Antonine_Baths',  
 'Bardo_National_Museum',  
 'Odeon_Theater',  
 'Haydrah',  
 'ENFIDAVILLE_WAR_CEMETERY',  
 'North_Africa_American_Cemetery_and_Memorial',  
 'Les_Ports_Puniques_de_Carthage',  
 'Roman_Amphitheatre',  
 'Kasserine_ruin',  
 'Byrsa_Hill',  
 'Salambo_Tophet',  
 'Quartier_Magon',  
 'Cisterns_of_La_Malga',  
 'Chemtou',  
 'Tubernuc',  
 'gigthis',  
 'Utica',  
 'Dougga',  
 'Subaytilah',  
 'El_Jem',  
 'Kerkouane',  
 'Bulla_Regia_-_Site_Archéologique',  
 'Basilica_Fausti']
```

# Application des algorithmes:

## A\* STAR

- Principe**: éviter de parcourir des chemins estimés trop long par rapport à ceux connus.
- f(n)**: une fonction d'évaluation de son coût total pour un sommet n.
- h(n)**: une heuristique de coût estimé (minimum) pour aller d'un sommet n vers un autre n+1.
- g(n)**: le coût réel pour atteindre le sommet n.
- La fonction f(n) est le coût total estimé :  $f(n) = h(n) + g(n)$

```
nx.astar_path(G, source='Utica',target='gigithis',heuristic=None)
```



# Résultat de l'algorithme:

## A\* STAR

Fair le tour : **validé**

Passer par le monuments  
une et une seul fois : **validé**

Prend le plus court chemin :  
**validé**

k

```
['Utica',  
 'Bulla Regia - Site Archéologique',  
 'Chemtou',  
 'Dougga',  
 'Haydrah',  
 'Kasserine ruin',  
 'Subaytilah',  
 'gigthis',  
 'El Jem',  
 'ENFIDAVILLE WAR CEMETERY',  
 'Kerkouane',  
 'Tubernuc',  
 'Zaghuan Aqueduct',  
 'Bardo National Museum',  
 'Medina of Tunis',  
 'Salambo Tophet',  
 'Les Ports Punique de Carthage',  
 'Quartier Magon',  
 'Antonine Baths',  
 'Kobba Bent el Rey',  
 'Odeon Theater',  
 'Basilica Fausti',  
 'North Africa American Cemetery and Memorial',  
 'Cisterns of La Malga',  
 'Roman Amphitheatre',  
 'Byrsa Hill',  
 'Carthage Museum']
```



# LA STRATEGIE OPTIMALE

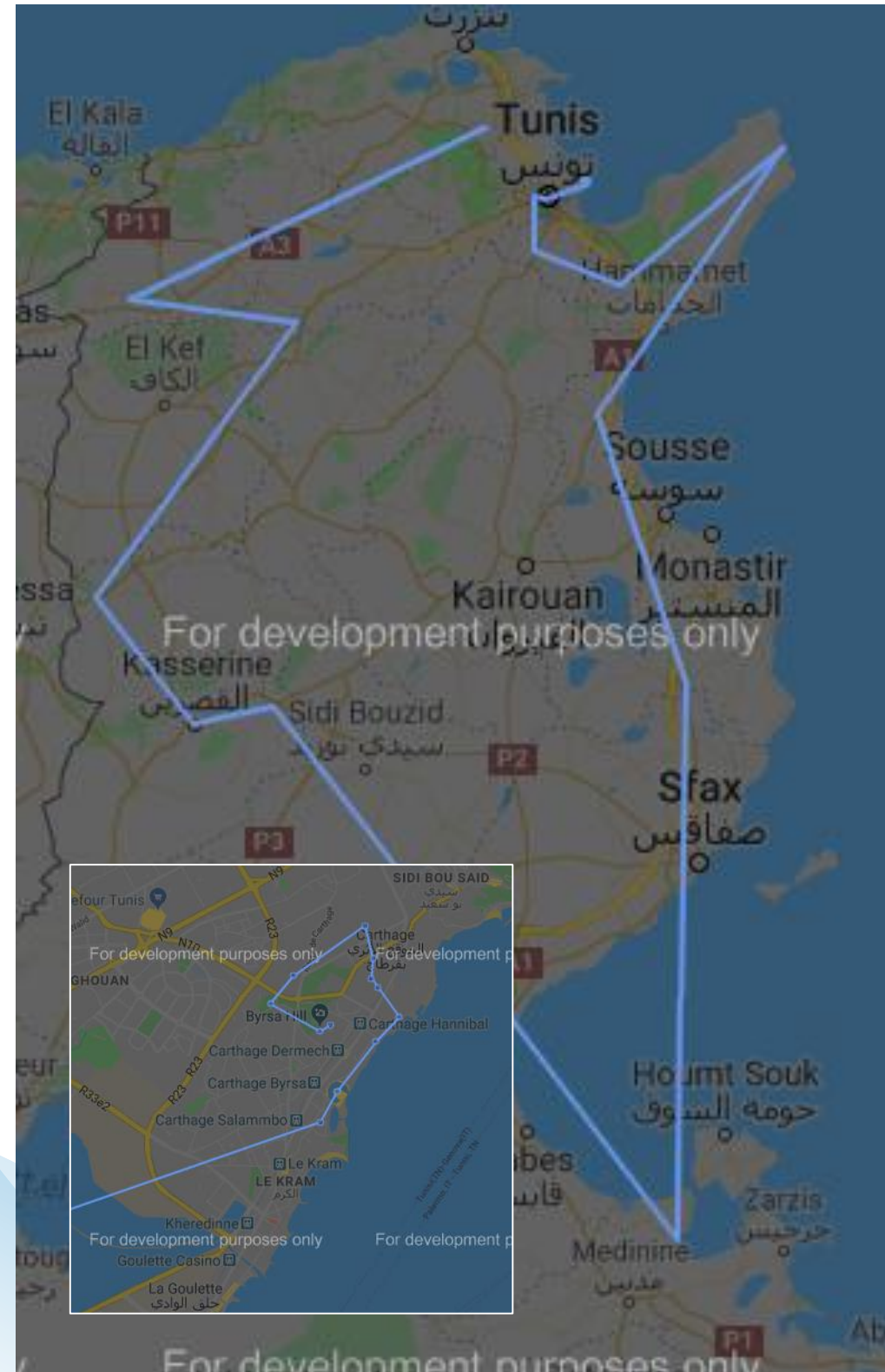


Calcule de la distance et la durée totale en utilisant Gmaps :

- ❑ Distance en vole de oiseau 1235km
- ❑ Distance routier 1636 km
- ❑ Durée de la tour 6jours

# LA STRATEGIE OPTIMALE

## MODELISATION DU CHEMIN



```
import gmplot
```

```
gmap1 = gmplot.GoogleMapPlotter(33.8439408, 9.400138, 7 )
```

```
latitude_list = [37.01646735,36.5596925,36.489542,36.42329105,35.56696765,35.17026285,35.23  
36.845036,36.8512445,36.8542306,36.8578685,36.8589587,36.8614744,36.865434  
longitude_list = [9.94475065,8.7541243,8.5769602,9.21870005,8.4438944,8.82476795,9.12433965  
,10.3228645,10.3254533,10.3313709,10.3349601,10.3317216,10.3306693,10.330
```

```
gmap1.scatter( latitude_list, longitude_list, '# FF0000',  
size = 40, marker = False )
```

```
gmap1.plot(latitude_list, longitude_list,  
'cornflowerblue', edge_width = 2.5)
```

```
gmap1.draw(r"C:\Users\asus\Desktop\map111.html" )
```

# Etude de cout

---

Distance : 1636km

Consummation Moyenne du carburant :  
33L/100km

Coût du carburant : 1115DT

Coût d'hébergement : 250DT/PERS

Coût total du voyage : 350Dt

GAIN : 4040DT



# Les contraintes

---

On eu des difficultés pour afficher l'algorithme hamiltonien vu que notre graph est complexe et il contient 27 sommets et 729 arrêtes

On a voulu utilisés  $A^*$  et l'algorithme hamiltonien afin qu'on puisse visiter tout les monuments en passant par le plus court chemin

Algo hamiltonien : tous les algo connu pour résoudre des problèmes NP-complets ont un temps d'exécution exceptionnel

Merci pour votre  
attention !