

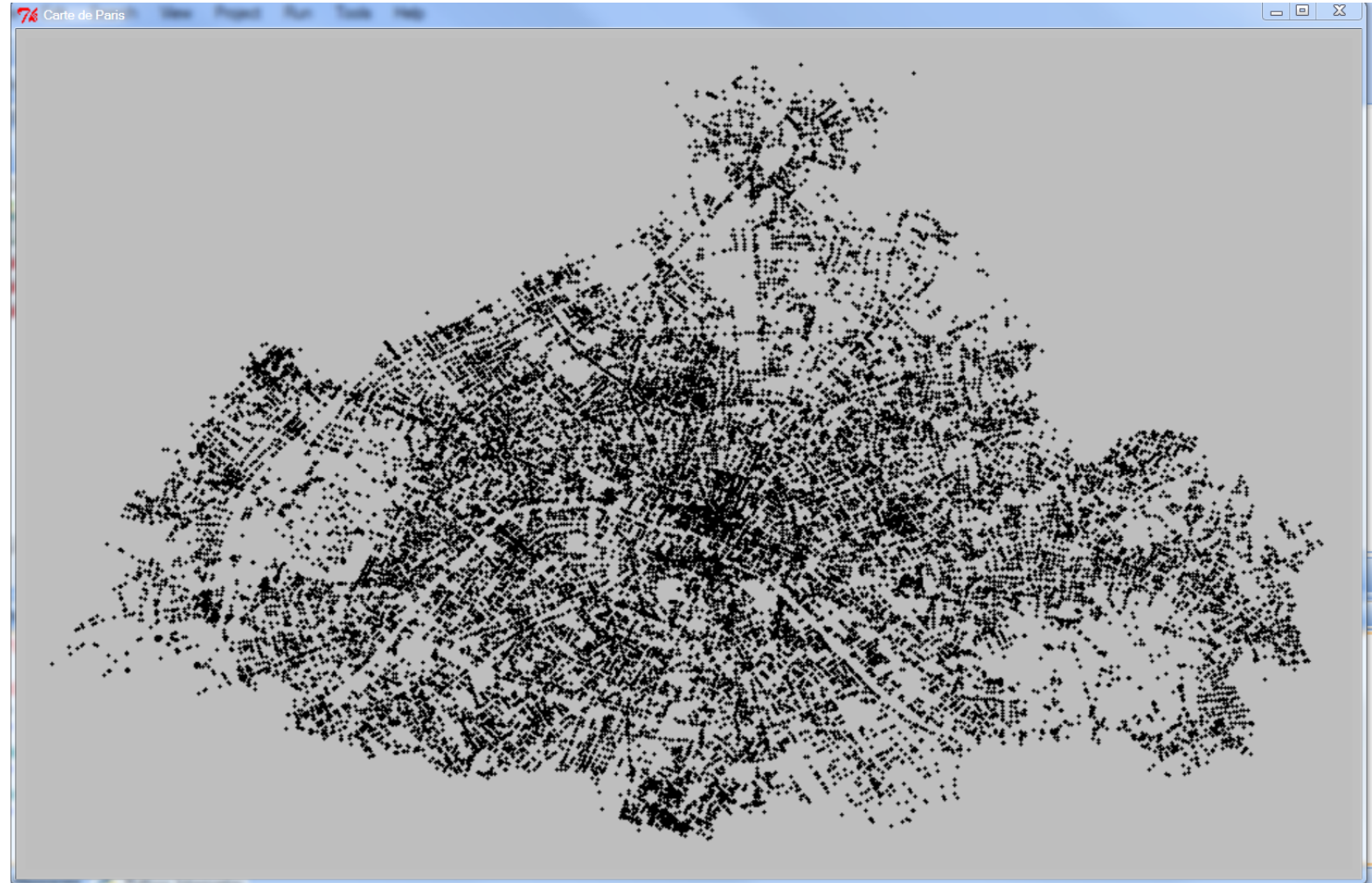
THÉORIE DES GRAPHS

TP DIJKSTRA

Polytech Tours
2019-2020

TP Dijkstra

- Ville de Paris
 - 29086 noeuds
 - 64538 arcs



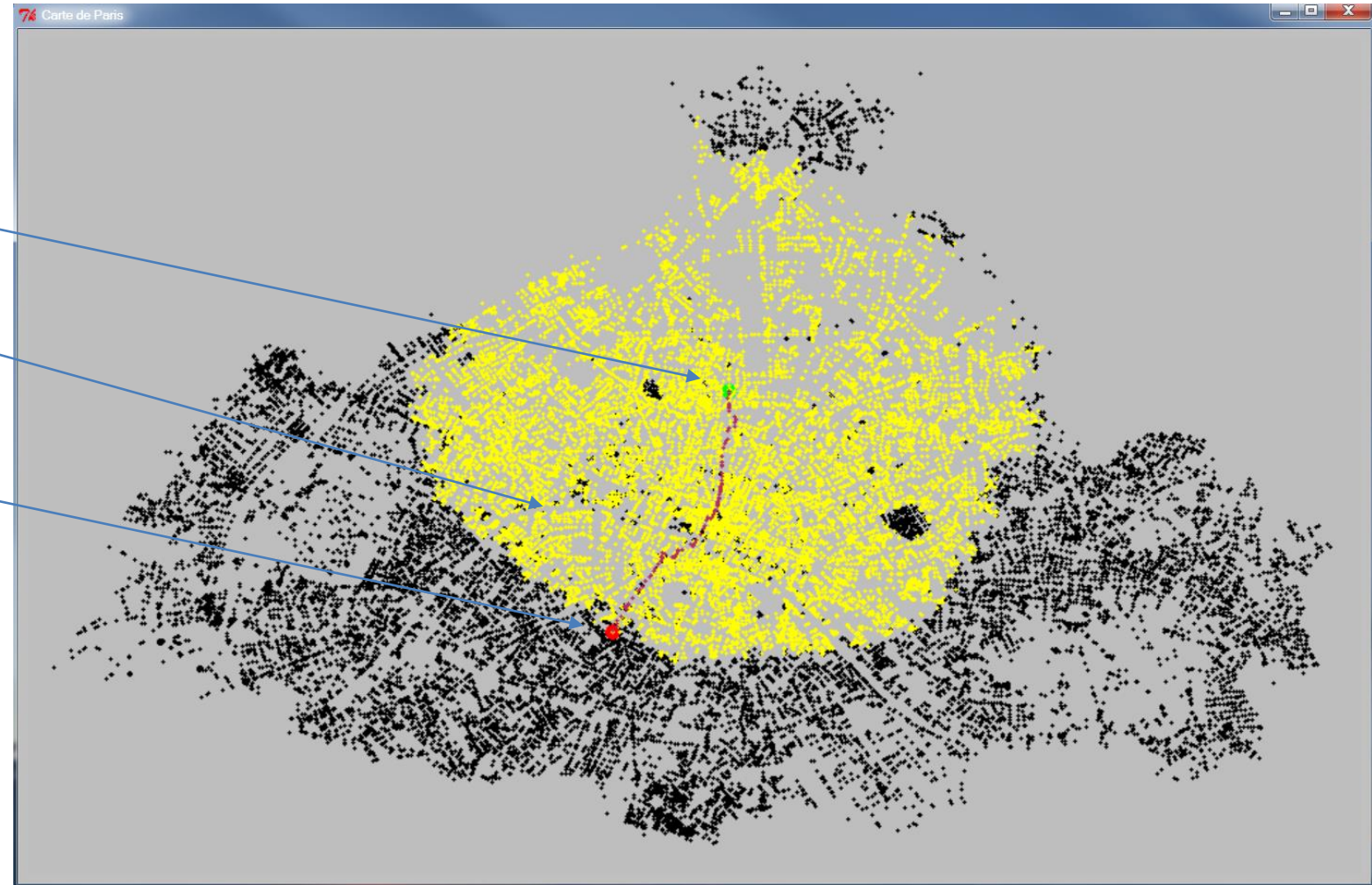
TP Dijkstra

- Ville de Paris
 - 29086 noeuds
 - 64538 arcs

Départ

Sommets parcourus

Arrivée



TP Dijkstra

Préalable

```
# #####  
# Importation de packages  
# #####  
  
• from tkinter import *  
• import os  
• import math  
• import time # pour avoir le temps de calcul cpu  
  
# #####  
# Lecture des fichiers  
# #####  
  
• fichier_noeuds = 'Noeuds.csv'  
• fichier_arcs = 'Arcs.csv'  
  
# PARAMETRES DEPENDANT DU FICHIER EN ENTREE  
  
• sommet_depart = 23160  
• sommet_destination = 27195  
  
• degre_vers_radian = math.pi/180.0
```


TP Dijkstra

Fichier Noeuds.csv

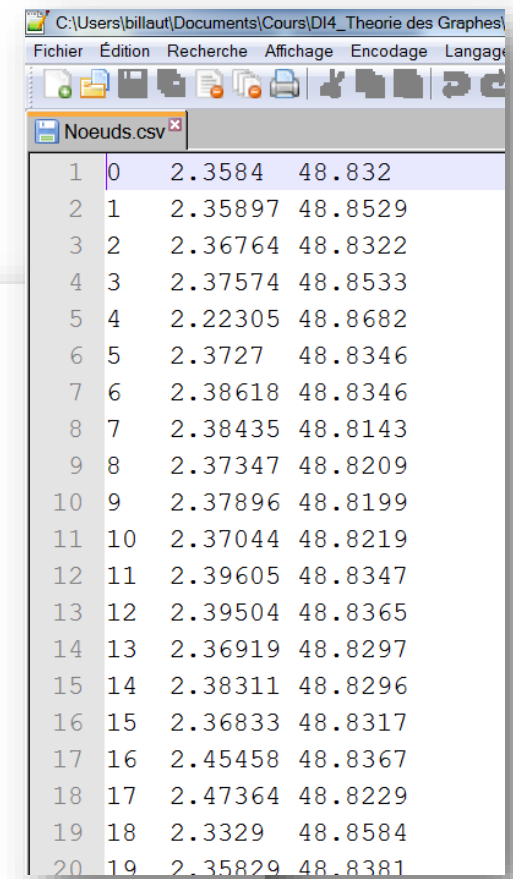
- Fichier Noeuds.csv
 - Numéro du nœud
 - Longitude
 - Latitude
- Code pour ouvrir le fichier et lire son contenu fourni.

```
• print('*** Lecture Noeuds ***')
• LesNoeuds = open(fichier_noeuds,"r")
  # format du fichier : indice \t Long  \t Lat \n
• touslesnoeuds = LesNoeuds.readlines()
• LesNoeuds.close()

• Longitude = []
• Latitude = []

• for un_noeud in touslesnoeuds:
  # Decoupage du contenu d'une ligne
  ce_noeud = un_noeud.split("\t")
  noeud = int(ce_noeud[0])
  Long = float(ce_noeud[1])
  Long = Long * degre_vers_radian      # conversion en radian
  Longitude.append(Long)
  Lat = float(ce_noeud[2].strip("\n"))
  Lat = Lat * degre_vers_radian        # conversion en radian
  Latitude.append(Lat)

• minLat = min(Latitude)
• maxLat = max(Latitude)
• minLong = min(Longitude)
• maxLong = max(Longitude)
• NbNoeuds = len(Longitude)
```



	indice	Long	Lat
1	0	2.3584	48.832
2	1	2.35897	48.8529
3	2	2.36764	48.8322
4	3	2.37574	48.8533
5	4	2.22305	48.8682
6	5	2.3727	48.8346
7	6	2.38618	48.8346
8	7	2.38435	48.8143
9	8	2.37347	48.8209
10	9	2.37896	48.8199
11	10	2.37044	48.8219
12	11	2.39605	48.8347
13	12	2.39504	48.8365
14	13	2.36919	48.8297
15	14	2.38311	48.8296
16	15	2.36833	48.8317
17	16	2.45458	48.8367
18	17	2.47364	48.8229
19	18	2.3329	48.8584
20	19	2.35829	48.8381

TP Dijkstra

Fichier Arcs.csv

- Fichier Arcs.csv
 - Origine de l'arc
 - Destination de l'arc
 - Distance (mètres)
 - Autre indicateur (non utilisé ici)
- Code pour ouvrir le fichier et lire son contenu fourni.

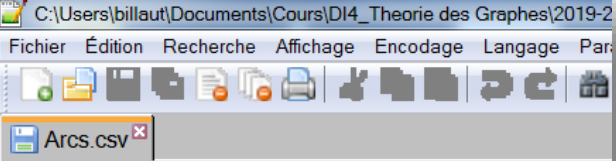
```
• print('*** Lecture Arcs ***')

• LesArcs = open(fichier_arcs,"r")
  # format du fichier : origine \t destination \t longueur \t dangerosite \n
• touslesarcs = LesArcs.readlines()
• LesArcs.close()

• Origine = []
• Destination = []
• Longueur = []
• Dangerosite = []
• suiv=[[] for j in range(NbNoeuds)]

• for un_arc in touslesarcs:
•     cet_arc = un_arc.split("\t")
•     Orig = int(cet_arc[0])
•     Origine.append(Orig)
•     Dest = int(cet_arc[1])
•     Destination.append(Dest)
•     Long = int(cet_arc[2])
•     Longueur.append(Long)
•     Dang = int(cet_arc[3].strip("\n"))
•     Dangerosite.append(Dang)
•     suiv[Orig].append(Dest)

• Nbarcs = len(Origine)
• print('NbArcs=',Nbarcs)
```



	1	2	3	4
1	1703	1704	143	286
2	1703	2087	110	440
3	1703	10748	105	420
4	1704	1703	143	286
5	1704	2702	147	588
6	1704	10747	102	408
7	1704	14814	5	10
8	1817	6208	126	252
9	1817	6241	106	212
10	1817	28462	61	244
11	1817	28459	200	800
12	1804	10906	42	168
13	1812	1804	66	264
14	28432	1812	34	136
15	1813	6355	46	184
16	1839	2503	59	236
17	1839	16902	29	116
18	1839	18631	26	104
19	28140	28184	115	115

TP Dijkstra

Figure

- TraceCercle(j, couleur, rayon) est une fonction qui trace un cercle pour la ville j, de couleur et de rayon donnés.
- Les paramètres servent à un bon affichage.

```
def TraceCercle(j, couleur, rayon):
    x=(Longitude[j]-minLong)*ratioWidth + border
    y=(Latitude[j]-minLat)*ratioHeight+ border
    y=winHeight-y
    can.create_oval(x-rayon, y-rayon, x+rayon, y+rayon, outline = couleur, fill = couleur)

fen = Tk()
fen.title('Carte de Paris')
coul_fond = "grey" #['purple','cyan','maroon','green','red','blue','orange','yellow']
coul_noeud = "black"

Delta_Long = maxLong-minLong
Delta_Lat = maxLat-minLat

border = 20 # taille en px des bords
winWidth_int = 900
winWidth = winWidth_int+2*border # largeur de la fenetre
winHeight_int = Delta_Lat*(winWidth_int/0.8)/Delta_Long
winHeight = winHeight_int+2*border # hauteur de la fenetre : recalculée en fonction de la taille

can = Canvas(fen, width = winWidth, height = winHeight, bg =coul_fond)
can.pack(padx=5,pady=5)

# cercles
rayon_noeud = 1 # rayon pour dessin des points
rayon_od = 5 # rayon pour origine et destination
for i in range(0,NbNoeuds):
    TraceCercle(i,coul_noeud,rayon_noeud)
```

TP Dijkstra

Algorithme de Dijkstra

- Ecrire une fonction $\text{Arc}(i,j)$ qui renvoie le numéro de l'arc (i,j) .
- $\text{time_start} = \text{time.clock}()$
- Les listes :
 - P_i contiendra les potentiels des sommets
 - Marque contiendra l'information que le sommet a déjà son potentiel définitif
 - LePrec gardera une trace du sommet précédent
- Initialiser ces listes
- $P_i[\text{sommet_depart}] = 0$
- $\text{Marque}[\text{sommet_depart}] = \text{True}$
- Initialiser les potentiels des successeurs de sommet_depart à la longueur de l'arc

TP Dijkstra

Algorithme de Dijkstra

- $\text{Fini} \leftarrow \text{Faux}$
- **Tant que** $\text{Fini} = \text{Faux}$ **Faire**
 - Rechercher le sommet avec le plus petit potentiel
 - Marquer ce sommet, le tracer en jaune
 - **Si** ce sommet est le sommet_destination **Alors** $\text{Fini} \leftarrow \text{Vrai}$
 - **Pour** chaque successeur k de ce sommet **Faire**
 - **Si** k n'est pas marqué **Alors**
 - Calculer le nouveau potentiel de k
 - **Si** le nouveau potentiel est inférieur à l'actuel **Alors**
 - Mettre à jour le potentiel de k
 - **Finsi**
 - **Finsi**
 - **FinPour**
- **FinTQ**

NbSommetsMarques = 1

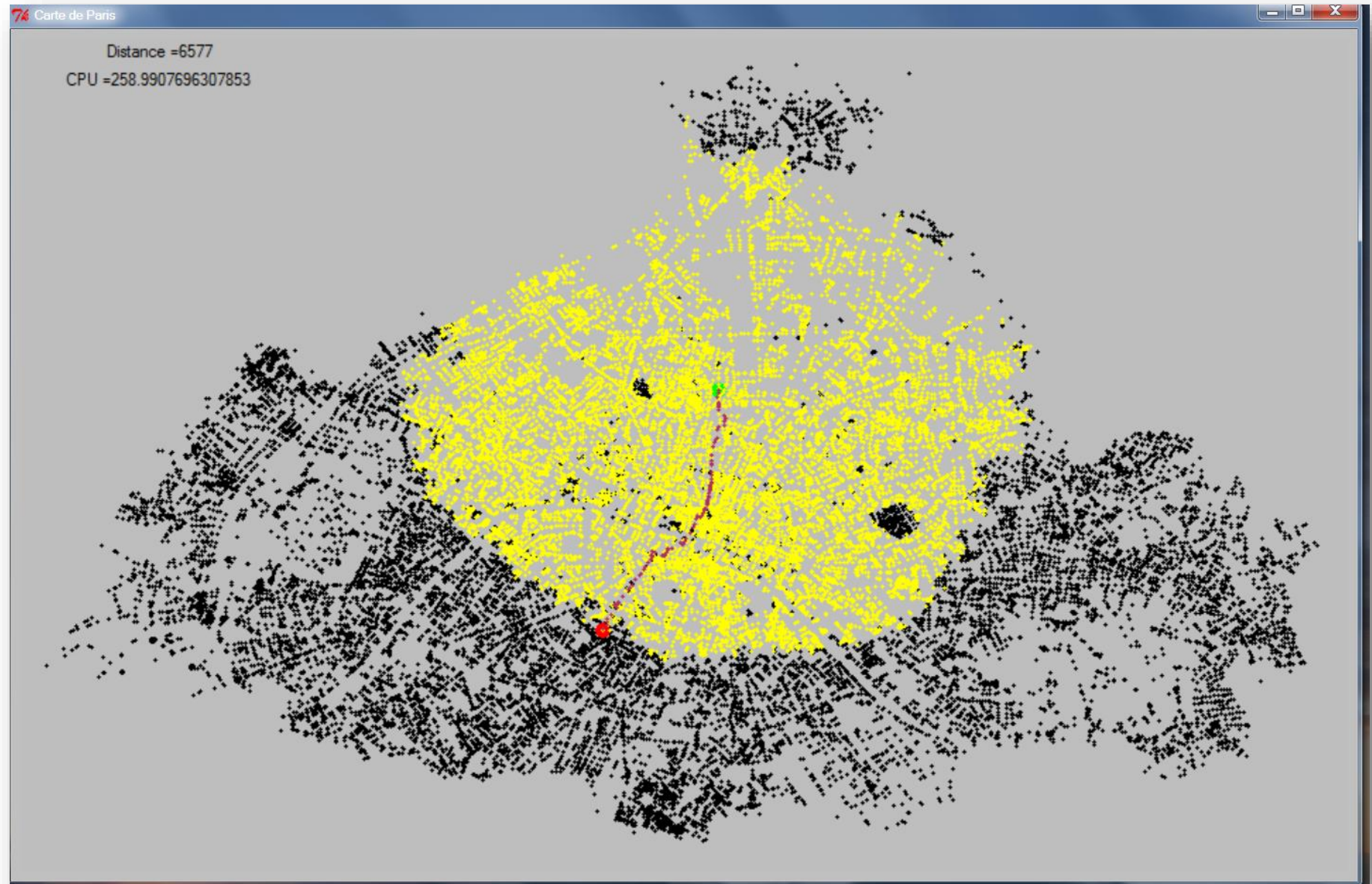
NbSommetsMarques += 1
tous les 100 sommets
print(NbSommetsMarques)

TP Dijkstra

Finalisation

- `time_end = time.clock()`
- Afficher le chemin trouvé en rouge
- Afficher la distance totale et le temps de calcul

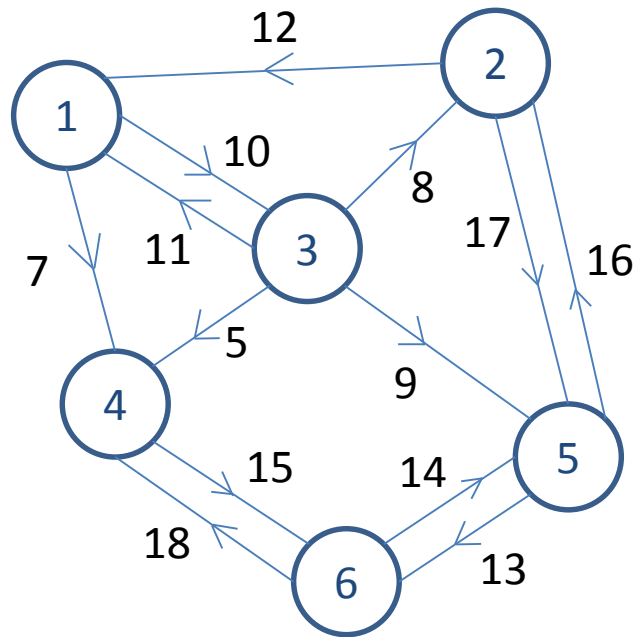
C'est trop long !!!



TP Dijkstra

Structure des données

- On crée une nouvelle liste :
long_suiv identique à suiv, mais qui
contient la longueur de l'arc plutôt
que le sommet successeur.



Origine

1	1	2	2	3	3	3	3	4	5	5	6	6
---	---	---	---	---	---	---	---	---	---	---	---	---

Destination

3	4	1	5	1	2	4	5	6	2	6	4	5
---	---	---	---	---	---	---	---	---	---	---	---	---

Longueur

10	7	12	17	11	8	5	9	15	16	13	18	14
----	---	----	----	----	---	---	---	----	----	----	----	----

suiv

1	2	3	4	5	6
3 4	1 5	1 2 4 5	6	2 6	4 5

long_suiv

1	2	3	4	5	6
10 7	12 17	11 8 5 9	15	16 13	18 14

TP Dijkstra

Modifications

- On n'utilise plus la fonction $\text{Arc}(i,j)$
- Créer une liste Candidats qui ne contient que les sommets qui ont un potentiel non infini. Initialiser cette liste avec les successeurs de sommet_depart.
- Restreindre la recherche du sommet ayant le plus petit potentiel aux sommets de cette liste (le retirer de la liste une fois choisi)

将对最低潜在峰顶的搜索限制在此列表的顶部（选择后将其从列表中删除）

TP Dijkstra

Finalisation

- Normalement, c'est mieux...

