Création d'un graphe et outils de conversion

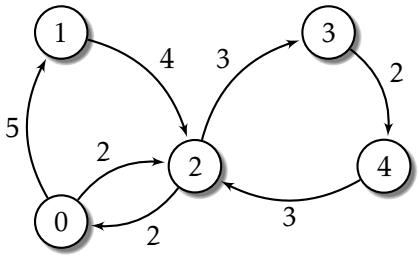
Objectif : L'objectif de ce tp est de rendre l'élève capable :

- de définir un graphe sous forme de listes de listes (listes d'adjacence ou matrices d'adjacence) ou sous forme de dictionnaire de dictionnaires
- de créer les outils de conversions entre les différents modes de représentation des graphes avec **Python**
- d'établir un graphe sous forme de dictionnaire de dictionnaires à partir de données stockées dans un fichier de type `.csv`

1 Graphes avec Python

1.1 Définition des graphes avec Python

Dans cette partie, on cherche à représenter, sous **Python** , le graphe ci-contre avec différentes structures de données.



↗	0	1	2	3	4
0	0	5	2	0	0
1	0	0	4	0	0
2	2	0	0	3	0
3	0	0	0	0	2
4	0	0	3	0	0

Q - 1 : *Traduire le graphe précédent sous forme de listes d'adjacence*

Q - 2 : *Traduire le graphe précédent sous forme d'une matrice d'adjacence*

Q - 3 : *Traduire le graphe précédent sous forme de dictionnaire de dictionnaires*

1.2 Outils de conversion des graphes

Q - 4 : *Écrire une fonction* `list2mat(graf)` *qui prend en argument une liste de listes* `graf` *représentant un graphe sous forme de liste d'adjacence et qui renvoie la matrice d'adjacence associé au graphe soit sous forme de liste de listes soit sous forme de tableau* ***numpy***.

Q - 5 : *Écrire une fonction* `mat2list(graf)` *fonction réciproque de la fonction* `list2mat(graf)`.

Q - 6 : *Écrire une fonction* `list2dic(graf)` *qui prend en argument une liste de listes* `graf` *représentant un graphe sous forme de liste d'adjacence et qui renvoie la représentation du graphe sous la forme d'un dictionnaire de dictionnaires.*

Q - 7 : *Écrire une fonction* `dic2list(graf)` *fonction réciproque de la fonction* `list2dic(graf)`.

2 Construire un graphe à partir d'une image

2.1 Données du problème

On dispose d'une image représentant le réseau autoroutier français.

Le fichier `Villes.csv` (tableur avec séparation des colonnes par `;`) contient des informations relative à l'image.

La première colonne donne l'identité d'un sommet sous la forme NX où X est le Xième sommet en balayant l'image du haut vers le bas et de gauche à droite.

Attention !
X est d'indice X-1 en langage **Python** .

La deuxième colonne donne le nom d'un sommet si celui-ci est associé à une ville. Dans le cas contraire, la case est vide.

Les colonnes 3 et 4 donne les coordonnées des sommets si on considère que l'origine est au centre de l'image, de largeur 10 et isométrique.

Les colonnes suivantes permettent de lister les voisins du sommet NX.

Q - 8 : *Afficher l'image* `autoroutes.jpg` *dans une figure* ***Python*** .

2.2 Construction du dictionnaire

Dans un premier temps, il convient de charger les données.

Q - 9 : *Lire le fichier* `Villes.csv` *en plaçant son contenu dans une liste de chaines de caractères* `data`.

On crée 3 listes vides `noms`, `coords` et `voisins` telles que au final :

- `noms` contient les valeurs de la deuxième colonne. Lorsque sur une ligne la valeur de la deuxième colonne est une chaine de caractère vide, on ajoute à `noms`, le nom absolu du sommet donné dans la première colonne.
- `coords` contient pour chaque ligne, un tuple avec les coordonnées du sommet
- `voisins` est une liste des listes des voisins. La ième valeur de la liste `voisins` et la liste des voisins de `i`.
  Attention ! comme les noms commencent à 1 et que **Python** commence à 0, il faudra retrancher 1 à la valeur des voisins.

Q - 10 : *Créer et remplir les listes* `noms`, `coords` *et* `voisins`.

Les coordonnées ne sont pas très intéressantes pour une superposition directe avec l'image du réseau routier. On propose donc de convertir les coordonnées.

Q - 11 : *Écrire une fonction* `conv(pt)` *qui prend en argument un tuple* `pt`, *coordonnées cartésienne initiale d'un sommet et qui renvoie les coordonnées en indice de pixel du sommet sur la carte.*

Remarque : l'image a une largeur de 583 pixels et une hauteur de 600 pixels.

Q - 12 : *Créer une liste* `ncoords` *dont les coordonnées sont les coordonnées en pixel des sommets de l'image du réseau routier.*

Q - 13 : *Afficher des points bleus sur chacun des sommets.*

Q - 14 : *A partir la liste* `voisins` *et la liste* `ncoords`, *tracer le réseau.*

Q - 15 : *Vérifier que les chemins sont bi-directionnels. Un sommet i doit être voisin d'un de ses voisins j. Corriger le fichier* `Villes.csv` *si besoin.*

Q - 16 : *Écrire une fonction* `distance(pt1, pt2)` *qui prend en argument deux tuples de coordonnées et qui renvoie la longueur (en pixels) entre ces deux points.*

Q - 17 : *A partir d'une distance connue entre deux sommets, modifier la fonction* `distance` *pour que cette dernière renvoie la valeur en km.*

Q - 18 : *Construire un dictionnaire* `graf` *où les clés i sont les numéros des sommets i et les valeurs des dictionnaires dont les clés j sont les numéros des sommets voisins j de la clé j. Les valeurs sont alors les distances entre le sommet i et le sommet j.*