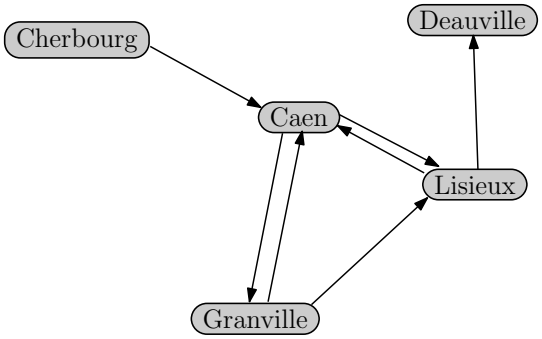
Structures des graphes

EXERCICE 1 – STRUCTURES DE DONNÉES À TRAVERS UN EXEMPLE

Soit G le graphe suivant :



Q1. Écrivez le tableau listant les nom des sommets triés selon l'ordre alphabétique. Puis représentez le graphe G sous forme de matrice d'adjacence. Pourquoi faut-il garder le tableau contenant les noms des sommets ? (Ne cherchez pas très loin pour la dernière question.)

Q2. En prenant comme fonction de hachage :

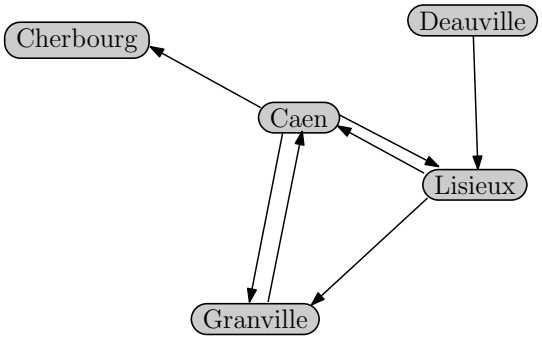
$$\text{chaine de caracteres} \mapsto \text{position de la première lettre dans l'alphabet modulo } 10$$

représentez le graphe comme un dictionnaire (table de hachage) où les clés sont le nom des villes et les valeurs les listes des successeurs des sommets sous forme de listes (simplement) chaînées.

EXERCICE 2 – RETOUR À L'ENVOYEUR

On veut écrire une fonction qui étant donné un graphe orienté G retourne le graphe miroir ; c'est-à-dire le graphe obtenu à partir de G en retournant le sens de chacune des arêtes.

Par exemple, pour le graphe de l'exercice précédent, on souhaite renvoyer



Q1. Écrire un algorithme qui prend un graphe M sous forme de matrice d'adjacence et qui renvoie le graphe miroir de M. Quelle est la complexité de cet algorithme ?

Q2. Maintenant on suppose qu'un graphe est un dictionnaire où les clefs sont les noms des sommets et les valeurs sont les listes des successeurs sous formes de listes.

Écrire un algorithme qui prend un tel graphe G et qui renvoie le graphe miroir de G. Quelle est la complexité de cet algorithme ?

Évidemment, dans le pseudo-code, on peut parcourir les clefs d'un dictionnaire, ainsi que les valeurs dans une liste, avec une boucle "Pour".

Complexité chez les graphes

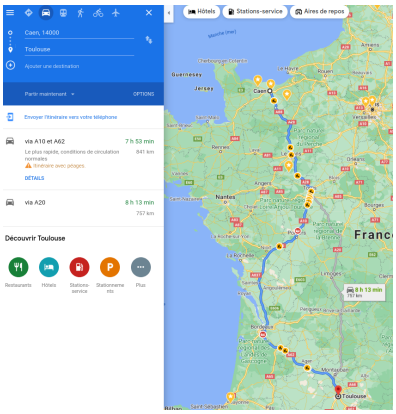
EXERCICE 3 – QUEL ALGORITHME DE PPC CHOISIR ?

Pour chacun des graphes suivants, indiquez s'il vaut mieux choisir :

- l'algorithme de Dijkstra avec une complexité en $O(|S|^2)$ (implémentation tableau).
- l'algorithme de Dijkstra avec une complexité en $O(|A| \times \log(|S|))$ (implémentation tas binaire).
- ou l'algorithme de Bellman-Ford (complexité $O(|S| \times |A|)$).

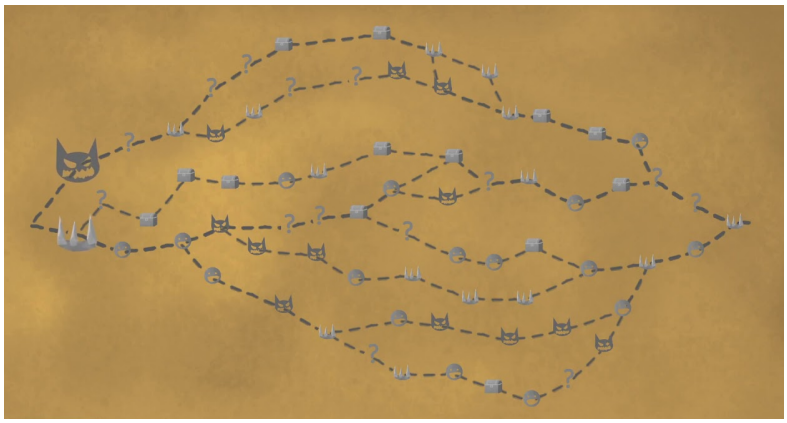
Bien sûr, la complexité est primordiale !

Q1. le graphe constitué des 10000 plus grandes communes françaises, où tout couple de sommets est reliée par une arête dont le poids est égal au temps de trajet en voiture entre les deux communes (avec comme référence Google Maps – par exemple)



Q2. le graphe de la ville de Caen, où chaque sommet est une intersection et où les arêtes représentent les rues reliant les intersections

Q3. un graphe provenant d'un donjon dans un jeu vidéo. Certaines salles nous font perdre des points de vie, d'autres nous en fait gagner. On veut accéder à la salle au trésor depuis l'entrée du donjon en minimisant la perte totale de points de vie.



Q4. un graphe non orienté avec des arêtes au poids positifs où le degré moyen serait $\sqrt{|S|}$ où $|S|$ est le nombre de sommets.

EXERCICE 4 – UNE FONCTION MYSTÈRE

Voici une fonction mystère sur un graphe non orienté G et un sommet s appartenant à G :

```
Fonction Mystère_mystérieux(G : graphe, s : sommet){
        Pour tout sommet u de G
        Faire Colorier u en blanc

        Colorier s en noir

        P = pile contenant uniquement s
        nb = 1

        Tant que P n'est pas vide
        Faire   t = sommet de P
                Dépiler P
                Pour tout voisin v de t
                Faire   Si v est colorié en blanc
                        Alors   Empiler v dans P
                                Colorier v en noir
                                nb = nb + 1

        Renvoyer nb
```

Q1. Que fait la fonction mystère ? (On ne demande pas de justifier.)

Q2. Quelle est la complexité de la fonction mysère ?

Q3. A-t-on procédé à un parcours en largeur ? en profondeur ?

Problèmes algorithmique sur les graphes

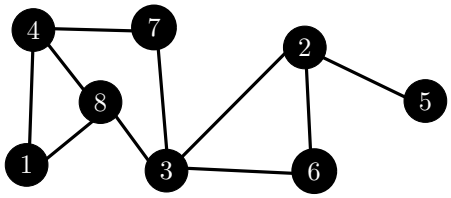
EXERCICE 5 – NOMBRE D'ARÊTES INTERNES

Soit G un graphe non orienté.

Écrire en pseudo-code une fonction qui étant donnés un graphe non orienté sans boucle G et une liste de sommets X donne le nombre d'arêtes qui ont leurs deux extrémités dans X.

Par exemple, pour le graphe ci-dessous, et pour `X = [1,2,3,6]`, la fonction doit renvoyer `T`.

Quelle est la complexité de votre algorithme ?

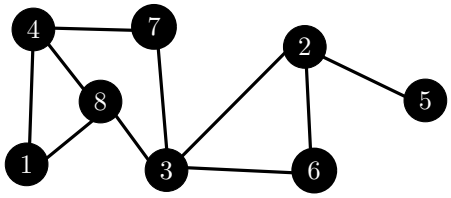


EXERCICE 6 – ENSEMBLE DOMINANT

Soit G un graphe non orienté.

Un ensemble D de sommets est dit dominant pour G si tout sommet du graphe G est soit dans D, soit voisin d'un sommet de D.

Q1. Existe-t-il un ensemble dominant à deux sommets pour le graphe ci-dessous?



Q2. Ecrire en pseudo-code une fonction qui prend un graphe G et une liste de sommets D, et qui renvoie vrai si D est dominant ; faux sinon.

Votre algorithme doit être efficace !

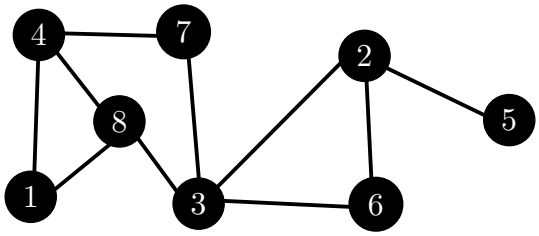
Q3. Montrez la complexité de votre algorithme. Il devrait être en $O(|S| + |A|)$.

EXERCICE 7 – MEILLEUR SCORE (CT 2021 DEUXIÈME SESSION)

Pour cet exercice, les graphes sont non orientés et chaque sommet porte une étiquette, qui est un nombre.

Le **score d'un sommet** est le produit de son étiquette avec le nombre de ses voisins qui portent une étiquette plus grande que lui.

Par exemple, pour le graphe :



le sommet `4` a pour score 4 (son étiquette) fois 2 (seuls 2 voisins de 4 ont une étiquette plus grande que 4 : il s'agit de `7` et `8`), c'est-à-dire 8. Dans le même ordre d'idée, le score de `1` est 2, le score de `2` est 6, le score de `3` est 9 et les scores de `5`, `6`, `7` et `8` valent 0.

Q1. Écrire une fonction `score` qui prend en paramètre un graphe étiqueté `G` et un sommet `s` et qui renvoie le score de `s`.

Q2. Écrire une fonction `sommet_score_maximum` qui prend en paramètre un graphe étiqueté `G` qui renvoie un des sommets de score maximal. Par exemple, pour le graphe ci-dessus, le score maximal est 9, donc il doit renvoyer `3`.

Q3. Quelle est la complexité de votre fonction `sommet_score_maximum` ? Justifiez.

Partie TP

EXERCICE 8

Le but de cette partie TP est de voir l'influence de la structure de données (matrice d'adjacence ou listes d'adjacence) sur l'efficacité des algorithmes de graphe. Un sujet `python` est disponible depuis `ecampus`.

Q1. Écrire une fonction `construire_graphe` où :

- **Entrée :** liste de triplets $(sommet1, sommet2, poids)$
- **Sortie :** un dictionnaire de dictionnaires G tel que si un triplet $(s1, s2, p)$ existe, alors $G[s1][s2]$ vaut le poids p.

Exemple. Si L vaut :

```
[("CHU","Campus 2",12),("Café des images","CHU",8),
("Campus 1","Saint-Pierre",5), ("CHU","Saint-Pierre",24),
("CHU","Campus 1",14),("Saint-Pierre","Campus 1",6),
("Campus 1","CHU",14),("Saint-Pierre","Gare",11)]
```

alors la sortie de la fonction est le dictionnaire :

```
{'CHU': {'Campus 2': 12, 'Saint-Pierre': 24, 'Campus 1': 14}, 'Campus 2': {},
'Café des images': {'CHU': 8}, 'Campus 1': {'Saint-Pierre': 5, 'CHU': 14},
'Saint-Pierre': {'Campus 1': 6, 'Gare': 11}, 'Gare': {}}
```

Q2. Écrire une fonction `construire_matrice_adjacence` où :

- **Entrée :** liste de triplets $(sommet1, sommet2, poids)$
- **Sortie :** un couple (S, M) où S est la liste des sommets et M est une matrice de sorte que :
  - $M[i][j] = p$ s'il existe un triplet $(S[i], S[j], p)$ dans `L` ;
  - $M[i][j] = +\infty$ sinon.

Exemple. Si L vaut :

```
[("CHU","Campus 2",12),("Café des images","CHU",8),
("Campus 1","Saint-Pierre",5), ("CHU","Saint-Pierre",24),
("CHU","Campus 1",14),("Saint-Pierre","Campus 1",6),
("Campus 1","CHU",14),("Saint-Pierre","Gare",11)]
```

alors la sortie de la fonction peut être le couple (S, M) où S vaut :

```
['CHU', 'Campus 2', 'Café des images', 'Campus 1', 'Saint-Pierre', 'Gare']
```

et

$$M = \begin{pmatrix} +\infty & 12 & +\infty & 14 & 24 & +\infty \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty \\ 8 & +\infty & +\infty & +\infty & +\infty & +\infty \\ 14 & +\infty & +\infty & +\infty & 5 & +\infty \\ +\infty & +\infty & +\infty & 6 & +\infty & 11 \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty \end{pmatrix}$$

Q3. Écrire une fonction `sommet_minimal_matrice_adjacence` où :

- **Entrée :** deux paramètres S, M comme à la question 2
- **Sortie :** le sommet tel que la somme des poids des arêtes entrantes est minimale.

Exemple. Si S vaut :

```
['CHU', 'Campus 2', 'Café des images', 'Campus 1', 'Saint-Pierre', 'Gare']
```

et

$$M = \begin{pmatrix} +\infty & 12 & +\infty & 14 & 4 & +\infty \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty \\ 8 & +\infty & +\infty & +\infty & +\infty & 1 \\ 14 & +\infty & +\infty & +\infty & 3 & +\infty \\ +\infty & +\infty & 20 & +\infty & +\infty & 11 \\ +\infty & +\infty & +\infty & +\infty & 2 & +\infty \end{pmatrix}$$

alors `sommet_minimal_matrice_adjacence (S,M)` doit renvoyer `'Saint-Pierre'` car trois arêtes arrivent à `'Saint-Pierre'` et sont de poids respectifs 4, 3, 2. La somme vaut 9 et est minimale. En effet, la somme des poids des arêtes entrantes des autres sommets sont 22 (`'CHU'`), 12 (`'Campus 2'`), 20 (`'Café des images'`), 14 (`'Campus 1'`) et 12 (`'Gare'`).

Q4. Écrire une fonction `sommet_minimal` où :

- **Entrée :** un graphe G comme la sortie de la question 1
- **Sortie :** le sommet tel que la somme des poids des arêtes entrantes est minimale. (pareil qu'avant)

Q5. Écrire une fonction `bellman_ford_matrice_adjacence` où :

- **Entrée :** trois paramètres S, M, s_0 où (S, M) désigne un graphe comme à la question 2 et $s_0 \in S$.
- **Sortie :** un dictionnaire où les clefs sont les sommets et les valeurs sont les distances à s_0. Comme le nom l'indique, on utilisera Bellman-Ford !

Q6. Écrire une fonction `bellman_ford` où :

- **Entrée :** deux paramètres G, s_0 où G désigne un graphe comme à la question 1 et $s_0 \in S$.
- **Sortie :** pareil que précédemment !