Projet Playlist : acte 4 (algorithme de recommandation)

La station évalue le comportement d'écoute des utilisateurs de son service playlist et cherche à générer pour eux des suggestions d'autres chansons. À cette fin, les chansons sont stockées dans les nœuds d'un graphe orienté, dont les arcs représentent l'ordre dans lequel les chansons se sont succédé et avec quelle fréquence (représentée par un poids entier sur chaque arc).

L'implémentation se fera à l'aide :

- d'une classe Nœud qui a un attribut ***song*** de type chanson et un attribut booléen ***visité*** .
- d'une classe Graphe qui a un attribut **ListeSommets** qui est la liste de ses nœuds et un attribut ***mat*** de type matrice d'entiers qui est la matrice de transition du graphe en tenant compte du poids des arcs. On prévoira pour cette classe graphe une méthode ***get_index_node(song)*** qui prend en paramètre une chanson et renvoie son indice dans la liste de nœuds constituant le graphe et -1 si aucun nœud se référant à l'objet song n'est trouvé.

Ces nouvelles classes seront ajoutées dans le fichier *classes.py*

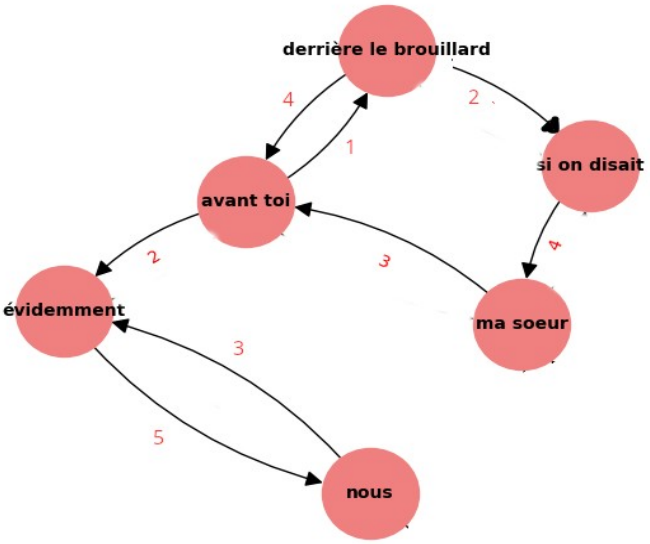
Les utilisateurs de la station devraient pouvoir afficher une playlist constituée de leurs chansons préférées. Cela implique qu'une chanson de démarrage soit choisie (qui apparaîtra en premier dans la playlist). Ensuite, parmi les chansons qui ne sont pas encore dans la playlist, celle qui a été jouée le plus fréquemment en succession directe avec la dernière chanson affichée, sera ajoutée à la playlist. Il s'agira donc d'un parcours en profondeur du graphe qu'on implémentera avec une pile (dont la classe est donnée dans le fichier *classes.py*). À cet effet, lorsqu'un nœud possède plusieurs voisins, on les empilera dans l'ordre croissant des poids. On pourra pour cela utiliser la fonction ci-dessous qui prend en paramètre une liste de couples de la forme (rang,poids) et qui renvoie la liste des rangs correspondants aux poids non nuls croissants dans une liste.

```
def trier_par_poids_non_nul(liste):
    #filtrer les éléments où le poids est non nul
    liste_filtree = [couple for couple in liste if couple[1]!=0]

    #trie la liste filtrée par le poids (index 1 du tuple)
    liste_triee = sorted(liste_filtree,key = lambda x: x[1])

    #récupère les rangs triés
    rangs_tries = [couple[0] for couple in liste_triee]

    return rangs_tries
```

Pour ce projet, nous considérerons que les chansons sont celles stockées dans la table *chansons* mise à disposition dans la base de données *chansons.db* et que le profil de l'utilisateur est détaillé dans un fichier. Pour ce travail, un fichier *leo.py* est donné qui contient la chanson de départ et le comportement de Léo, utilisateur de la playlist, dans une matrice de transition entre les chansons.

Projet Playlist : acte 4 (algorithme de recommandation)

On donne ci-dessous le graphe correspondant au comportement de Léo.



Remarque : par soucis de concision , seul le titre de la chanson apparaît dans chaque nœud.

Le travail consiste donc à afficher la playlist constituée des chansons préférées de Léo dans le bon ordre.

On rendra dans une archive zip:

- le fichier *classes.py* enrichi des classes Nœud et Graphe
- un fichier *recommandations.py* implémentant le contrôleur de l'architecture MVC (modèle,vue,contrôleur)
- le fichier *interface_playlist.py* modifié pour l'instanciation de la vue dans *recommandations.py*