TP 5 Algorithmique et structures des données

Algorithme de Floyd-Warshall

Nous souhaitons implémenter l'algorithme de Floyd-Warshall . L'algorithme calcule les chemins les plus courts entre toute paire de sommets d'un graphe. Si le graphe est orienté, il peut contenir des poids négatifs, mais pas de circuits négatifs (une condition qui *ne sera pas testée* par l'algorithme).

L'algorithme Floyd-Warshall prend en entrée un graphe G représenté par une matrice d'adjacence W, définie comme il suit : soit G = (S, A, w), avec $i,j \in S$

$W[i,j] = 0$ si $i = j$

$W[i,j] = INF$ si $i \neq j$ et $(i,j) \notin A$

$W[i,j] = w(i,j)$ si $(i,j) \in A$

Les sommets sont numérotés de 0 à |S| -1.

Vous utilisez la classe GraphMat définie dans le fichier **graphMATRICE**

Un graphe a une taille size, un type (ORIENTED, NON_ORIENTED) et une matrice.

A) Définir une fonction :

??? `floyd_warshall(graphe,....?)`

qui calcule le plus court chemin entre toute paire de sommets d'un graphe représenté par une matrice d'adjacence.
Pour chaque paire de sommets i et j, l'algorithme calcule deux informations :

- le coût **D[i,j]** d'un plus court chemin de **i** à **j**,
- le prédécesseur **P[i,j]** du sommet **j** sur un plus court chemin de **i** à **j** (ou la valeur NIL qui indique que le sommet **j** n' a pas de prédécesseur).

Les valeurs calculées D et P seront enregistrés dans une (ou deux) structure de données accessible après l'exécution de l'algorithme :à votre choix la structure sera passée en paramètre ou retournée par la fonction.

B) Définir une fonction :

`affiche_chemin( ?...int i, int j)`

qui, pour tout sommet **i** et **j**, affiche ***tous les sommets dans l'ordre*** sur un plus court chemin de **i** à **j** ou une valeur qui indique qu'il n'y a pas de chemin entre i et j. La fonction utilise l'information calculée P[i,j]. (Suggestion: définir simplement la fonction d'une façon récursive similaire à `print_chemin` dans les support du cours).