MPSI - ITC - 18 mai 2026 - TP 29

Graphes : l'algorithme A*

Fourni dans le squelette de code

Le squelette de code `mpsi_itc_tp_29_squelette.py` fourni :

- Une implémentation naïve de files de priorité avec des fonctions `file_prio`, `est_vide`, `enfiler_ou_maj` et `defiler`.
- Une implémentation de l'algorithme de Dijkstra
- Une variable globale `graphe` contenant un dictionnaire de listes d'adjacence du métro parisien. Les arêtes sont pondérées par la durée en secondes nécessaire l'emprunter.
- Une variable globale `noms` contenant un dictionnaire qui à chaque numéro d'arrêt associe le nom de station associé
- Une variable globale `coords` contenant un dictionnaire qui à chaque numéro d'arrêt associe ces coordonnées dans un repère cartésien.

Exercice 1 (À vol d'oiseau)

Écrire une fonction `vol_oiseau(depart, arrivee)` qui prend en entrée deux numéros d'arrêt, et qui renvoie le temps qu'il faudrait à un métro pour relier les deux arrêts en question, en supposant qu'il se déplace à une vitesse constante de 10 mètres par seconde. Dans le système de coordonnées utilisées, une unité correspond à 25,7 mètres.

Exercice 2 (Algorithme A*)

Codez une fonction `a_etoile(depart, arrivee)` qui explore le graphe du métro, en utilisant comme heuristique le temps à vol d'oiseau entre deux stations, pour trouver le plus court chemin (en terme de temps) entre `depart` et `arrivee`.

Exercice 3 (Comparaison Dijkstra - A* : Résultats)

Pour différentes paires d'arrêts de métro, comparez les résultats que vous obtenez avec A* et avec Dijkstra.
On admet pour l'instant que quand toutes les arêtes ont un poids positif, l'algorithme de Dijkstra est correct (il calcule bien, pour une source donnée, les plus courtes distances entre cette source et tous les autres sommets du graphe). Que pouvez-vous conjecturer concernant l'algorithme A* ?

Exercice 4 (Comparaison Dijkstra - A* : Nombre d'arrêts vus et visités)

1. Modifiez l'algorithme de Dijkstra fourni pour que :
   - Il prenne en deuxième argument un numéro d'arrêt `arrivee`
   - Il arrête d'explorer le graphe dès que `arrivee` est défilé de la file de priorité et sa distance ajoutée au dictionnaire des plus courtes distances
   - Il renvoie une paire (*nombre de sommets visités*, *nombre de sommets vus*) où
     - Le nombre de sommets visités est le nombre de sommets pour lesquels ont a déjà trouver leur plus petite distance depuis la source
     - Le nombre de sommets vus est égal au nombre de sommets visités plus le nombre de sommets encore présents dans la file de priorité
2. Modifiez l'algorithme A* pour qu'il renvoie une paire (*nombre de sommets visités*, *nombre de sommets vus*).
3. Pour différentes paires d'arrêts de métro, comparez les deux algorithmes en termes du nombre de sommets vus et visités. Commentez.

Exercice 5 (Comparaison Dijkstra - A* : Temps de calcul)

Pour différentes paires d'arrêts de métro, comparez le temps de calcul des deux algorithmes. Vous pouvez vous aider des éléments vus pendant le TP 12 du 1° décembre.