GRAPHES ET RÉSEAUX **ISC - HEIG-VD**
T. Franchetti, J.-F. Hêche **Printemps 2025**

Travail pratique **Algorithme A***

Objectifs

L'objectif de ce travail pratique est d'observer le comportement de l'algorithme A* et d'évaluer et comparer les performances obtenues par rapport à l'algorithme de Dijkstra.

Algorithme A*

Applicable à la recherche d'un plus court chemin d'une source s à une destination t, l'algorithme A* est une version modifiée de celui de Dijkstra, utilisant une heuristique pour approcher la distance restant à parcourir pour atteindre la destination.

Dans l'algorithme de Dijkstra les sommets sont traités dans l'ordre croissant de la distance les séparant de la source s, jusqu'à atteindre la destination t. Dans la variante A*, l'ordre de traitement se base sur l'estimation de la longueur totale d'un chemin de s à t passant par un sommet donné i. Cette estimation est égale à la somme de la longueur du meilleur chemin connu de s à i et de l'estimation fournie par l'heuristique de la distance séparant i de t. En traitant les sommets dans l'ordre croissant de cette somme on espère diminuer le nombre d'itérations nécessaires avant d'atteindre la destination t. Le pseudocode de l'algorithme est disponible dans le support de cours.

Selon les propriétés et les caractéristiques de l'heuristique utilisée et des graphes auxquels il est appliqué, l'algorithme A* correspondra à une méthode exacte, fournissant toujours un itinéraire optimal, ou une méthode heuristique fournissant souvent une bonne solution mais sans garantie d'optimalité.

Création des graphes pondérés

Afin d'observer visuellement le comportement de l'algorithme, la première étape consistera à créer un labyrinthe, à partir d'une grille carrée de n rangées par n colonnes, qui sera ensuite transformé en un graphe pondéré non orienté (par suppression de murs/ajout d'arêtes et ajout de poids). Pour cela il vous suffira de rependre votre classe `DfsGenerator` du travail pratique précédent.

Mise en œuvre de l'algorithme

Vous devez programmer la méthode A* en complétant l'implémentation de la classe `AStar` fournie. Pour les heuristiques, cinq variantes seront mises en place. Dans les descriptions qui suivent le paramètre $c_{\min}$ représente le poids minimum des arêtes du réseau (accessible par la méthode `minWeight`.

- ▷ Heuristique H_0 (`DIJKSTRA`) : L'estimation de la distance restant à parcourir est toujours nulle ($h_i = 0$ quel que soit le sommet i). Cette variante servira de référence car la méthode correspond alors à l'algorithme de Dijkstra.

▷ Heuristique H_1 (`INFINITY_NORM`) : L'estimation de la distance restant à parcourir est égale au plus grand écart entre les coordonnées de i et t :

$$h_i = c_{\min} \cdot \max(\Delta_x, \Delta_y) = c_{\min} \cdot \max(|x_t - x_i|, |y_t - y_i|)$$

où (x_t, y_t) correspond aux coordonnées de la case (dans la grille $n \times n$) associée au sommet destination t et (x_i, y_i) correspond aux coordonnées de la case associée au sommet i. Il s'agit de la distance associée à la norme L_∞.

▷ Heuristique H_2 (`EUCLIDEAN_NORM`) : L'estimation de la distance restant à parcourir est égale à la distance euclidienne séparant i de t arrondie à l'entier inférieur :

$$h_i = c_{\min} \left\lfloor \sqrt{\Delta_x^2 + \Delta_y^2} \right\rfloor = c_{\min} \left\lfloor \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \right\rfloor .$$

▷ Heuristique H_3 (`MANHATTAN`) : L'estimation de la distance restant à parcourir est égale à la distance de Manhattan :

$$h_i = c_{\min} (\Delta_x + \Delta_y) = c_{\min} (|x_t - x_i| + |y_t - y_i|) .$$

▷ Heuristique H_4 (`K_MANHATTAN`) : L'estimation de la distance restant à parcourir est égale à K fois celle de Manhattan :

$$h_i = K \cdot c_{\min} (|x_t - x_i| + |y_t - y_i|) .$$

Travail d'analyse

Afin d'observer et de quantifier les gains apportés par l'utilisation d'une heuristique pour guider l'algorithme de Dijkstra, plusieurs jeux de paramètres sont à votre disposition dans la classe `Experiment` dont l'implémentation doit être complétée. Pour chaque jeux de paramètres fournis vous générerez successivement $N = 100$ réseaux.

▷ Pour chacun d'eux vous appliquerez les quatre premiers algorithmes (afin de résoudre le même problème d'itinéraire optimal, la source et la destination étant fournies dans la classe `Experiment`) et récolterez la longueur des solutions trouvées et le nombre de sommets traités par chaque méthode (un sommet traité est un sommet retiré de la queue de priorité et dont la liste des voisins est parcourue dans le but de chercher à améliorer les meilleures distances connues).

▷ Vous afficherez ensuite, pour chaque algorithme, la longueur moyenne obtenue sur les N graphes générés, le nombre moyen de sommets traités et la diminution (en %) de ce nombre par rapport à l'algorithme de Dijkstra.

▷ Pour l'heuristique H_4 (qui n'est pas optimiste et ne définit pas un algorithme exact), vous l'appliquerez à chacun des N réseaux en faisant varier K de 2 à 8. Dans chaque cas vous présenterez des statistiques plus détaillées que ci-dessus, comprenant au minimum le pourcentage de solutions optimales obtenues, les erreurs relatives minimale, moyenne et maximale obtenues et le gain moyen (en %) pour le nombre de sommets traités.

Modalités et délais

- ▷ Le travail de programmation est à effectuer par groupe de deux, en Java, version 21 ou 23.
- ▷ L'archive contenant les sources du projet est disponible sur le site Cyberlearn du cours.
- ▷ Vous devez rendre une archive (au format `zip`) contenant toutes les sources soigneusement documentées de votre projet complété ainsi qu'un fichier texte (ou markdown) contenant les statistiques sur les performances obtenues pour les jeux de paramètres fournis.
- ▷ Vous devez rendre votre travail sur Cyberlearn au plus tard le **dimanche 27 avril 2025** (avant minuit).