Algorithmes pour la Résolution de Problèmes L3

TP 3 *Recherche non-informé*

Exercice 0. Finir le TP précédent si ce n'est pas déjà fait !

Exercice 1. Un peu de théorie

Question 1. Quelle est la différence entre un graphe d'états et l'arbre de recherche ?

Question 2. Un graphe d'états fini mène-t-il toujours à un arbre de recherche fini ? Définissez précisement les types de graphe d'états qui impliquent toujours à des arbres de recherche finis.

Question 3. Quelle est la différence entre un état et un noeud de l'arbre de recherche ?

Exercice 2. Le retour du cavalier

Question 1. Reprenons le code du problème du cavalier étudié lors des derniers TPs. La méthode `search` proposée est une implémentation d'un algorithme de recherche non-informée. Lisez attentivement cette implémentation. Quel est l'algorithme de recherche implémenté ? Justifiez votre réponse.

Question 2. L'implémentation proposé est capable de fournir une solution pour une grille de taille 5×5, mais échoue lorsque la taille passe à 6×6. Indiquez la compléxité spatiale de cet algorithme pour les deux tailles de grilles. Justifiez votre réponse.

Question 3. Nous allons maintenant proposer d'autres implémentations pour le problème de manière à pouvoir résoudre des grilles de plus grande taille. En se basant sur la méthode `search`, implémentez une méthode `DFS` implémentant une version de l'algorithme de recherche en profondeur (DFS). *Rappel : Les algorithmes de recherche se distinguent par la façon dont ils gèrent la frontière.*

Question 4. Implémentez une méthode `IteratedDFS` implémentant une version de l'algorithme de recherche en profondeur itéré. Utilisez $\ell = 1$ comme profondeur maximale initiale, et incrémentez la profondeur maximale de 1 à chaque itération. Cet algorithme est pertinent pour ce problème ? Justifiez votre réponse.

Question 5. Pouvez-vous résoudre le problème du tour du cavalier ***fermé*** pour une grille de taille 6×6 avec vos deux nouvelles implémentations ? Et pour des tailles plus grandes ?

Exercice 3. Retour aux cannibales

Reprenons le code du problème des cannibales étudié lors du dernier TP.

Question 1. Utilisez les 3 algorithmes de recherche non-informée implémentés précedemment pour résoudre le problème des cannibales. Combien de noeuds chaque algorithme génère-t-il avant de trouver une solution ? Quel algorithme est le plus efficace pour ce problème ?

Question 2. Modifiez vos algorithmes pour qu'ils évitent de générer des états redondants. Combien de noeuds chaque algorithme génère-t-il maintenant avant de trouver une solution ? Avec ces modifications, quel algorithme est le plus efficace pour ce problème ? Justifiez vos réponses.