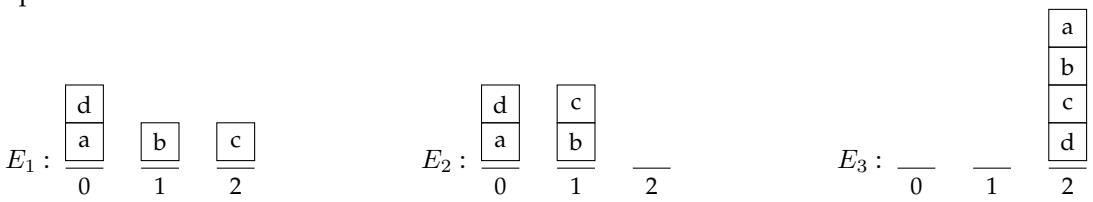
TP n°2 d'AAIA : A* et AWA* pour le monde des blocs

Attention : Vous devez répondre (avant le 30 avril) à un questionnaire Moodle.

Les algorithmes A* et AWA* sont utilisés pour résoudre des problèmes de planification modélisés sous la forme de recherches de plus courts chemins dans des graphes d'états. Dans ce TP, nous allons utiliser et comparer ces algorithmes pour résoudre le problème du monde des blocs.

1 Modélisation du problème du monde des blocs sous la forme d'un problème de planification

Nous supposerons qu'il y a n blocs et k piles (numérotées de 0 à $k-1$). Nous dirons d'une pile qu'elle est vide s'il n'y a aucun bloc posé dessus. Trois exemples d'états possibles avec $n = 4$ blocs (a, b, c, et d) et $k = 3$ piles sont représentés ci-dessous :



L'état E_2 a une pile vide (2), et l'état E_3 a deux piles vides (0 et 1).

Une action consiste à prendre un bloc au sommet d'une pile non vide p_1, et à le déposer au sommet d'une autre pile p_2. Une telle action est notée $p_1 \to p_2$. Nous noterons $actions(E)$ l'ensemble de toutes les actions possibles pour un état E donné. Par exemple, les actions possibles pour l'état E_2 sont :

$$actions(E_2) = \{0 \to 1, 0 \to 2, 1 \to 0, 1 \to 2\}$$

L'application d'une action $i \to j \in actions(E)$ sur un état E permet d'obtenir un nouvel état que nous noterons $t(E, i \to j)$. Par exemple, l'application de l'action $2 \to 1$ sur E_1 permet d'obtenir l'état $E_2 = t(E_1, 2 \to 1)$.

L'objectif du TP est d'écrire un programme pour trouver la plus petite séquence d'actions permettant de passer d'un état initial (où les blocs sont uniformément répartis sur les piles) à un état final (où tous les blocs sont empilés sur la dernière pile, du plus grand au plus petit). Par exemple, la plus petite séquence d'actions permettant de passer de l'état initial E_1 à l'état final E_3 est : $< 2 \to 1, 0 \to 2, 1 \to 2, 1 \to 2, 0 \to 2 >$.

Questions (réponses à donner sur Moodle) :

Q1 : Combien d'actions différentes sont-elles possibles pour l'état E_1 ?
Q2 : Etant donné un état E de n blocs sur k piles, quelle est la taille maximale de $actions(E)$?
Q3 : Etant donné un état E de n blocs sur k piles ayant v piles vides, quelle est la taille de $actions(E)$?
Q4 : Quel est l'ordre de grandeur du nombre total d'états différents possibles ?

2 Définition du graphe d'états

Etant donnés n et k, nous définissons le graphe d'états $G = (S, A)$ tel que

— S est l'ensemble des états possibles, chaque état étant une configuration différente des n blocs sur les k piles ;
— $A = \{(E, E') \mid \exists i \to j \in actions(E), E' = t(E, i \to j)\}$

Pour trouver la plus petite séquence d'actions permettant de passer d'un état initial E_0 à un état final E_f, nous allons rechercher le plus court chemin allant de E_0 à E_f dans G.

Questions (réponses à donner sur Moodle) :

Q5 : Le graphe G est-il orienté ?
Q6 : Quels sont les algorithmes qui peuvent être utilisés pour rechercher ce plus court chemin ? (plusieurs réponses possibles)
Q7 : Quel est l'algorithme le plus efficace pour rechercher ce plus court chemin ?
Q8 : Quelle est la complexité en temps de cet algorithme par rapport à $|S|$ et $|A|$?
Q9 : Quelle est la complexité en temps de cet algorithme par rapport au nombre de blocs (n) et de piles (k) ?

3 Heuristiques possibles pour le monde des blocs

Vous trouverez sur Moodle une implémentation de A* pour rechercher le plus court chemin allant de E_0 à E_f : le fichier `astar.cpp` contient une implémentation générique de l'algorithme A* utilisant les classes `State` et `StateGraph` pour définir le problème de planification, et les fichiers `stateGraph.cpp` et `state.cpp` contiennent une implémentation des classes `StateGraph` et `State` pour le problème du monde des blocs. La classe `StateGraph` contient une méthode `heuristic` prenant en paramètre une instance `s` de la classe `State` et retournant une borne inférieure du coût du plus court chemin allant de `s` à un état final dans le graphe d'états. Pour le moment, cette méthode retourne 0, ce qui fait que le programme n'est pas très efficace.

Compilez le programme à l'aide de la commande :

```
g++ -std=c++11 -O3 -Wall -Werror astar.cpp -o astar
```

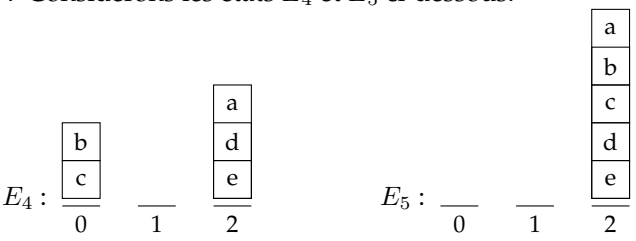
et exécutez `astar` en fixant le nombre de piles à 4, et en fixant le nombre de blocs à 6, puis 7, et enfin 8. Nous voyons qu'il y a peu de chances d'arriver à résoudre le problème en un temps raisonnable au delà de 10 blocs.

Pour améliorer les performances, nous allons commencer par introduire des heuristiques. Nous décrivons ci-dessous trois heuristiques possibles pour le monde des blocs (nous rappelons que l'état final est l'état où tous les blocs sont empilés sur la dernière pile, du plus grand au plus petit, comme illustré dans l'état E_5 ci-dessous).

— h_1 : Nombre de blocs ne se trouvant pas sur la dernière pile.
— h_2 : Nombre de blocs ne se trouvant pas sur la dernière pile, plus deux fois le nombre de blocs b tels que b se trouve sur la dernière pile mais il devra nécessairement être enlevé de cette pile pour ajouter et/ou supprimer d'autres blocs sous lui.
— h_3 : Nombre de blocs de E_i ne se trouvant pas sur la dernière pile, plus le nombre de blocs se trouvant au dessus de chaque bloc ne se trouvant pas sur la dernière pile.

Questions (réponses à donner sur Moodle) : Rappelons qu'une heuristique est admissible si elle retourne toujours une valeur inférieure ou égale à la longueur du plus court chemin jusqu'à un état final, et qu'une heuristique h est plus informée qu'une heuristique h' si pour tout état E, $h(E) \leq h'(E)$, tandis que les deux heuristiques sont incomparables s'il existe deux états E et E' tels que $h(E) < h'(E)$ et $h(E') > h'(E')$.

Q10 : Considérons les états E_4 et E_5 ci-dessous.



Quelles sont les valeurs de $h_1(E_4)$, $h_2(E_4)$ et $h_3(E_4)$ dans le cas où l'état final est E_5 ?

Q11 : L'heuristique h_1 est-elle admissible ?
Q12 : L'heuristique h_2 est-elle admissible ?
Q13 : L'heuristique h_2 est elle plus informée, moins informée, ou incomparable par rapport à l'heuristique h_1.
Q14 : L'heuristique h_3 est-elle admissible ?
Q15 : L'heuristique h_3 est elle plus informée, moins informée, ou incomparable par rapport à l'heuristique h_2.

Votre travail : Implémentez les heuristiques admissibles, parmi h_1, h_2 et h_3, et étudiez l'impact de ces heuristiques sur les temps de résolution. Concevez une nouvelle heuristique admissible encore mieux informée, et implémentez la. Cette nouvelle heuristique devrait vous permettre de répondre à la question Q17.

Questions (réponses à donner sur Moodle) :

Q16 : Quelle est la longueur (en nombre d'actions) de la solution optimale pour $n = 16$ blocs et $k = 4$ piles ?
Q17 : Quelle est la longueur (en nombre d'actions) de la solution optimale pour $n = 20$ blocs et $k = 4$ piles ?

4 Implémentation de AWA*

Implémentez l'algorithme AWA* vu en cours, en adaptant le code de A*. Observez l'influence du paramètre w sur la résolution.

Questions (réponses à donner sur Moodle) :

Q18 : Quelle est la longueur (en nombre d'actions) de la solution optimale pour $n = 24$ blocs et $k = 5$ piles ?