EI339 Artificial Intelligence Homework 1

Zhou Litao 518030910407 F1803016

September 27, 2020, Fall Semester

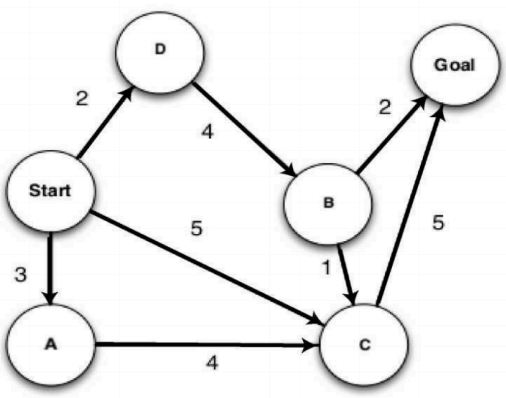
Exercise 1 (Search) For each of the following graph search strategies, work out the order in which states are expanded, as well as the path returned by graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. Remember that in graph search, a state is expanded only once.

1. Depth-first search.
2. Breadth-first search.
3. Uniform cost search.

Solution.

1. Depth-First Search:
   Expand Order: Start → A → C → Goal
   Returned Path: Start → A → C → Goal
2. Breadth-First Search:
   Expand Order: Start → A → C → D → Goal
   Returned Path: Start → C → Goal
3. Uniform cost Search:
   Expand Order: Start → A → C → D → B → Goal
   Returned Path: Start → C → Goal

□



Exercise 2 (Search) 2 SEARCH Which of the following are true and which are false?

1. Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic.
2. h(n) = 0 is an admissible heuristic for the 8 -puzzle.
3. A* is of no use in robotics because percepts, states, and actions are continuous.
4. Breadth-first search is complete even if zero step costs are allowed.
5. Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.

Solution.

1. No, DFS may find the goal on initial exploration.
2. Yes, since $0 \leq h^*(n)$ anyway.
3. No, delicate heuristic can be designed.
4. Yes, BFS doesn't count costs.
5. No, since $h^*(n) = 2$ if there exists no obstacles, but the heuristic can be greater than 2 if the goal is far enough.

□

Exercise 3 (Search) n vehicles occupy squares (1,1) through (n, 1) (i.e., the bottom row) of an n ×n grid. The vehicles must be moved to the top row but in reverse order; so the vehicle i that starts in (i, 1) must end up in (n − i + 1, n). On each time step, every one of the n vehicles can move one square up, down,left,or right,or stay put; but if a vehicle stays put, one other adjacent vehicle(but not more than one)can hop over it. Two vehicles cannot occupy the same square.

1. Calculate the size of the state space as a function of n.
2. Calculate the branching factor as a function of n.
3. Suppose that vehicle i is at (x_i, y) ; write a nontrivial admissible heuristic ha for the number of moves it will equire to get to its goal location $(n - i + 1, n)$, assuming no other vehicles are on the grid.
4. Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations?Explain.
   - $\sum_{i=1}^{n} h_i$
   - $\max\{h_1, \ldots, h_n\}$
   - $\min\{h_1, \ldots, h_n\}$

Solution.

1. n items can be distributed on $n \times n$ blanks. Hence there are $P_{n^2}^n$ states.
2. For every vehicle, there are five choices to make, so for every state, there will be 5^n possible next-state. The branching factor is 5^n
3. A heuristic for single vehicle can be the Manhattan distance

$$h_i = |n - i + 1 - x_i| + |n - y_i| \tag{1}$$

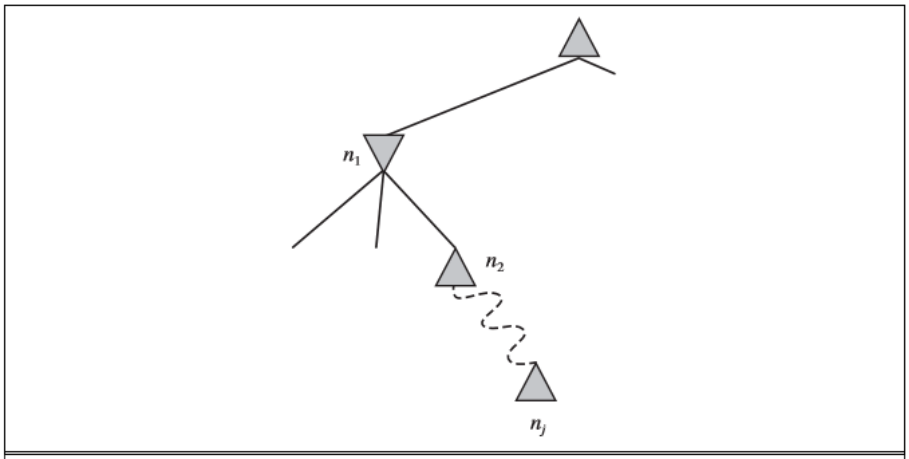


Figure 5.18 Situation when considering whether to prune node n_j.

4. $\sum_{i=1}^{n} h_i$ cannot be an admissible heuristic since the vehicles can move simultaneously. As a result, it is likely that $h(n) > h^*(n)$

   $\max\{h_1, \ldots, h_n\}$ is an admissible heuristic, since if we want to make all the vehicles to their destinations, it must take at least the time for moving one particular vehicle to its destination. i.e. $h_i(n) \leq h^*_{all}(n)$ for any i. Therefore, $\max\{h_1, \ldots, h_n\} \leq h^*$.

   $\min\{h_1, \ldots, h_n\}$ is also admissible because $\min\{h_1, \ldots, h_n\} \leq \max\{h_1, \ldots, h_n\} \leq h^*$

□

Exercise 4 (Search) Develop a formal proof of correctness for alpha-beta pruning. To do this, consider the situation shown in Figure 5.18. The question is whether to prune node n_j which is a max-node and a descendant of node n_1 The basic idea is to prune it if and only if the minimax value of n_1 can be shown to be independent of the value of nj.(35)

1. Mode n_1 takes on the minimum value among its children: $n_1 = \min(n_2, n_{21}, \ldots, n_{2b_2})$ Find a similar expression for n_2 and hence an expression for n_1 in terms of n_j.

2. Let l_i be the minimum (or maximum) value of the nodes to the $left$ of node n_i at depth i whose minimax value is already known. Similarly, let r_i be the minimum (or maximum) value of the unexplored nodes to the right of n_i at depth i. Rewrite your expression for n_1 in terms of the l_i and r_i values.

3. Now reformulate the expression to show that in order to affect n_1, n_j must not exceed a certain bound derived from the l_i values.

4. Repeat the process for the case where n_j is a min-node.

Proof. According to the algorithm, we will prune n_j if

Let n_i denote the node on n_j's path to the root at the depth of i, and $n_{i1}, \ldots, n_{ib_i}$ denote n_i's siblings. Suppose n_j is a max-node, i.e. j is an even number. We can derive the utility value of n_1 as follows

$$\begin{aligned} n_1 &= \min(n_2, n_{21}, \ldots, n_{2b_2}) \\ &= \min(\max(n_3, n_{31}, \ldots, n_{3b_3}), n_{21}, \ldots, n_{2b_2}) \\ &= \min(\max(\ldots \min(n_j, n_{j1}, \ldots, n_{jb_j}) \ldots, n_{31}, \ldots, n_{3b_3}), n_{21}, \ldots, n_{2b_2}) \end{aligned} \tag{2}$$

Let l_i be the minimum (or maximum) value of the nodes to the left of node n_i at depth i and r_i be the minimum (or maximum) value of the unexplored nodes to the right of n_i at depth i.

$$\begin{aligned} n_1 &= \min(l_2, n_2, r_2) \\ &= \min(l_2, \max(l_3, n_3, r_3), r_2) \\ &= \min(l_2, \max(l_3, \min(\max(\ldots, \min(l_j, n_j, r_j), \ldots)), r_3), r_2) \end{aligned} \tag{3}$$

In order to let n_j make a difference on n_1, it has to be hold that

$$\max(l_{j-1}, l_{j-3}, \ldots, l_3) < n_j < \min(l_j, l_{j-2}, \ldots, l_2) \tag{4}$$

Similarly, if n_i is an odd number,

$$\max(l_j, l_{j-2}, \ldots, l_3) < n_j < \min(l_{j-1}, l_{j-3}, \ldots, l_2) \tag{5}$$

Let $\alpha = \max(l_3, l_5, \ldots)$, denoting the best option of maximum value on n_i's path to root, and let $\beta = \min(l_2, l_4, \ldots)$, denoting the best option of minimum value on n_i's path to root.

It can be shown that during exploration, if $n_i \leq \alpha$ or $n_i \geq \beta$, we can prune n_i.

□