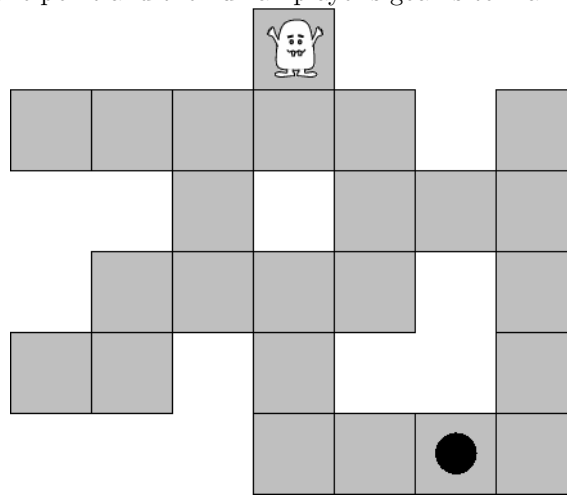


AI-EX-I Exercises

1. Pathfinding

Consider a simple computer game played in a maze like the one shown in the figure below. The maze is represented by the grey squares which are connected horizontally or vertically to other grey squares on a 2D grid. A human player places a disc at a fixed square inside the maze. The computer player controls a monster which always starts at the same square. The monster can move horizontally or vertically to an adjacent square. It can see the whole maze, including the position of the disc. The goal of the computer player is to move the monster to the position of the disc with a minimum number of moves. For each move the monster makes the human player gets one point and the human player's goal is to maximise the total score.



- Describe the A* algorithm to find a path from the monster's start position to the known position of the disc in the maze. Also specify an admissible, non-trivial heuristic for the A* algorithm and show that your heuristic is admissible.
- Does the greedy best-first search algorithm with an admissible heuristic in general find a better path for the monster than the A* algorithm? Is A* appropriate for the computer player to find an optimal path?
- Choose a standard graph search strategy to find an optimal position for the disc. Briefly justify your choice.

2. The 8-Puzzle

5	4	
6	1	8
7	3	2

Start State

1	2	3
8		4
7	6	5

Goal State

Consider the 8-puzzle as shown above. You have a 3×3 grid and 8 tiles numbered from 1 to 8. You can move tiles horizontally or vertically to the empty position. The goal is to reach the goal state shown in the right figure above.

- Formulate the problem as a path search problem and draw a small part of the search space (say, 7 states).

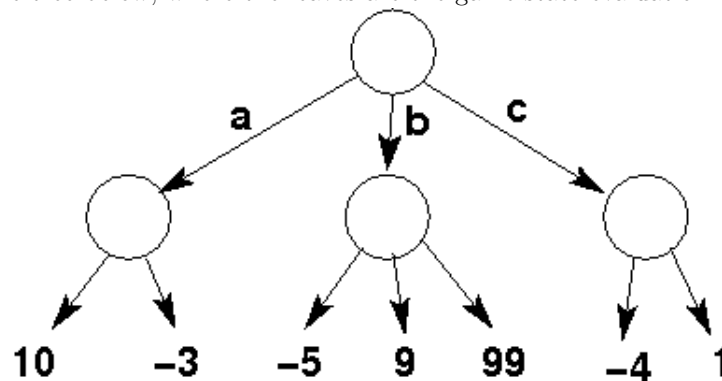
- (b) Suggest a suitable admissible heuristic function for this search problem. Why is your heuristic admissible?
- (c) Explain how A^* and greedy best-first search would traverse the state space. What is the difference between A^* and greedy best-first search?

3. A^* Algorithm

How should the cost function c and the heuristic h be computed to make A^* behave like (i) breadth first, (ii) uniform cost, and (iii) depth first search? Justify your answers.

4. Games

- (a) Consider the game tree below, where the leaves are the game state evaluation results.



Assuming the nodes are processed from left to right, which game states (identified by their values) do not have to be evaluated by minimax without changing the overall results? Explain why.

- (b) Now assume the opponent is a random player selecting one of the possible actions in the game tree, and the leaf nodes indicate the amount of money the top node player wins or loses. Which action is optimal in this situation? How does minimax have to be modified when playing against such an opponent? For such random opponent games in general, is it still possible to avoid evaluating all nodes?

5. Zero-Sum and Non-Zero-Sum Games

Consider a zero-sum two-player game with perfect information. Here, as is often the case in games, zero-sum means that one player's gain or loss is exactly balanced by the loss or gain of the other player. An example of such a game is illustrated by the game tree below. Player 1 has a choice between two moves. Then player 2 has a choice between three moves and finally player 1 can choose between another three moves to reach a terminal game state. The evaluation scores for the terminal states from the first player's view are indicated at the leaves of the tree: a positive number indicates the amount of money the first player wins from the second player, while a negative number indicates the amount of money the second player wins from the first player.

