DM577/DM879 Introduction to Artificial Intelligence

List of exercises – version of February 3, 2023

N.B.: *Many exercises are reproduced or adapted from the list of exercises at* `https://aimacode.github.io/aima-exercises/`*, included in the first three editions of the reference book. They are reproduced here for convenience.*

1 Intelligent agents

1. For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.
   (a) An agent that senses only partial information about the state cannot be perfectly rational.
   (b) There exist task environments in which no pure reflex agent can behave rationally.
   (c) There exists a task environment in which every agent is rational.
   (d) The input to an agent program is the same as the input to the agent function.
   (e) Every agent function is implementable by some program/machine combination.
   (f) Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.
   (g) It is possible for a given agent to be perfectly rational in two distinct task environments.
   (h) Every agent is rational in an unobservable environment.
   (i) A perfectly rational poker-playing agent never loses.
2. For each of the following activities, give a PEAS description of the task environment and characterize it.
   (a) Playing soccer.
   (b) Exploring the subsurface oceans of Titan.
   (c) Shopping for used AI books on the Internet.
   (d) Playing a tennis match.
   (e) Practicing tennis against a wall.
   (f) Performing a high jump.
   (g) Knitting a sweater.
   (h) Bidding on an item at an auction.
3. Let us examine the rationality of various vacuum-cleaner agent functions.
   (a) Show that the simple vacuum-cleaner agent function described in Figure 2.3 is indeed rational.
   (b) Describe a rational agent function for the case in which each movement costs one point. Does the corresponding agent program require internal state?
   (c) Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn? If not, why not?

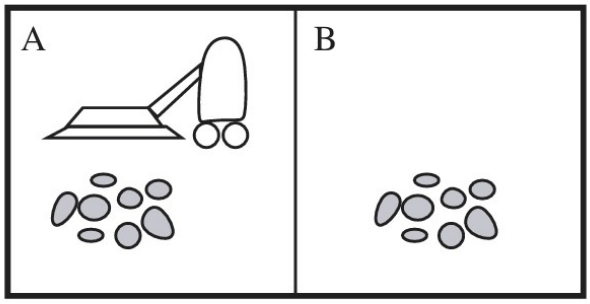

Figure 2.2 A vacuum-cleaner world with just two locations.

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

4. Define in your own words the following terms: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent.

5. Consider a modified version of the vacuum environment in which the agent is penalized one point for each movement.

   (a) Can a simple reflex agent be perfectly rational for this environment? Explain.

   (b) What about a reflex agent with state? Design such an agent.

   (c) How do your answers to a and b change if the agent's percepts give it the clean/dirty status of every square in the environment?

6. Consider a modified version of the vacuum environment in which the geography of the environment – its extent, boundaries, and obstacles – is unknown, as is the initial dirt configuration. In this variant, the agent can go Up and Down as well as Left and Right.

   (a) Can a simple reflex agent be perfectly rational for this environment? Explain.

   (b) Can a simple reflex agent with a randomized agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.

   (c) Can you design an environment in which your randomized agent will perform poorly? Show your results.

(d) Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

7. Repeat the previous exercise for the case in which the location sensor is replaced with a "bump" sensor that detects the agent's attempts to move into an obstacle or to cross the boundaries of the environment. Suppose the bump sensor stops working; how should the agent behave?

8. The vacuum environments in the preceding exercises have all been deterministic. Discuss possible agent programs for each of the following stochastic versions:

   (a) Murphy's law: twenty-five percent of the time, the Suck action fails to clean the floor if it is dirty and deposits dirt onto the floor if the floor is clean. How is your agent program affected if the dirt sensor gives the wrong answer 10% of the time?

   (b) Small children: At each time step, each clean square has a 10% chance of becoming dirty. Can you come up with a rational agent design for this case?

9. This exercise explores the differences between agent functions and agent programs.

   (a) Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.

   (b) Are there agent functions that cannot be implemented by any agent program?

   (c) Given a fixed machine architecture, does each agent program implement exactly one agent function?

   (d) Given an architecture with n bits of storage, how many different possible agent programs are there?

   (e) Suppose we keep the agent program fixed but speed up the machine by a factor of two. Does that change the agent function?

2 Basic search

1. Give a complete problem formulation for each of the following problems. Choose a formulation that is precise enough to be implemented.

   (a) There are six glass boxes in a row, each with a lock. Each of the first five boxes holds a key unlocking the next box in line; the last box holds a banana. You have the key to the first box, and you want the banana.

   (b) You start with the sequence ABABAECCEC, or in general any sequence made from A, B, C, and E. You can transform this sequence using the following equalities: AC = E, AB = BC, BB = E, and Ex = x for any x. For example, ABBC can be transformed into AEC, and then AC, and then E. Your goal is to produce the sequence E.

   (c) There is an $n \times n$ grid of squares, each square initially being either unpainted floor or a bottomless pit. You start standing on an unpainted floor square, and can either paint the square under you or move onto an adjacent unpainted floor square. You want the whole floor painted.

   (d) A container ship is in port, loaded high with containers. There 13 rows of containers, each 13 containers wide and 5 containers tall. You control a crane that can move to any location above the ship, pick up the container under it, and move it onto the dock. You want the ship unloaded.

   (e) Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.

   (f) A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high crates.

(g) You have a program that outputs the message "illegal input record" when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.

(h) You have three jugs, measuring 12 liters, 8 liters, and 3 liters, and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one liter.

2. A robot has the task of navigating out of a maze. The robot starts in the center of the maze facing north. It can turn to face north, east, south, or west. The robot can also move forward a certain distance, although it will stop before hitting a wall.

   (a) Formulate this problem. How large is the state space?

   (b) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

   (c) From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

   (d) In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

3. Consider a 9×9 grid of squares, each of which can be colored red or blue. The grid is initially colored all blue, but you can change the color of any square any number of times. Imagining the grid divided into nine 3×3 sub-squares, you want each sub-square to be all one color but neighboring sub-squares to be different colors.

   (a) Formulate this problem in the straightforward way. Compute the size of the state space.

   (b) You need color a square only once. Reformulate, and compute the size of the state space. Would breadth-first graph search perform faster on this problem than on the one in (a)? How about iterative deepening tree search?

   (c) Given the goal, we need consider only colorings where each sub-square is uniformly colored. Reformulate the problem and compute the size of the state space.

   (d) How many solutions does this problem have?

   (e) Parts (b) and (c) successively abstracted the original problem (a). Can you give a translation from solutions in problem (c) into solutions in problem (b), and from solutions in problem (b) into solutions for problem (a)?

4. Explain why problem formulation must follow goal formulation.

5. Consider the problem of finding the shortest path between two points on a plane that has convex polygonal obstacles as shown in Figure 3.31. This is an idealization of the problem that a robot has to solve to navigate in a crowded environment.

   (a) Suppose the state space consists of all positions (x, y) in the plane. How many states are there? How many paths are there to the goal?

   (b) Explain briefly why the shortest path from one polygon vertex to any other in the scene must consist of straight-line segments joining some of the vertices of the polygons. Define a good state space now. How large is this state space?

   (c) Define the necessary functions to implement the search problem, including an function that takes a vertex as input and returns a set of vectors, each of which maps the current vertex to one of the vertices that can be reached in a straight line. (Do not forget the neighbors on the same polygon.) Use the straight-line distance for the heuristic function.

   (d) Apply one or more of the algorithms in this chapter to solve a range of problems in the domain, and comment on their performance.

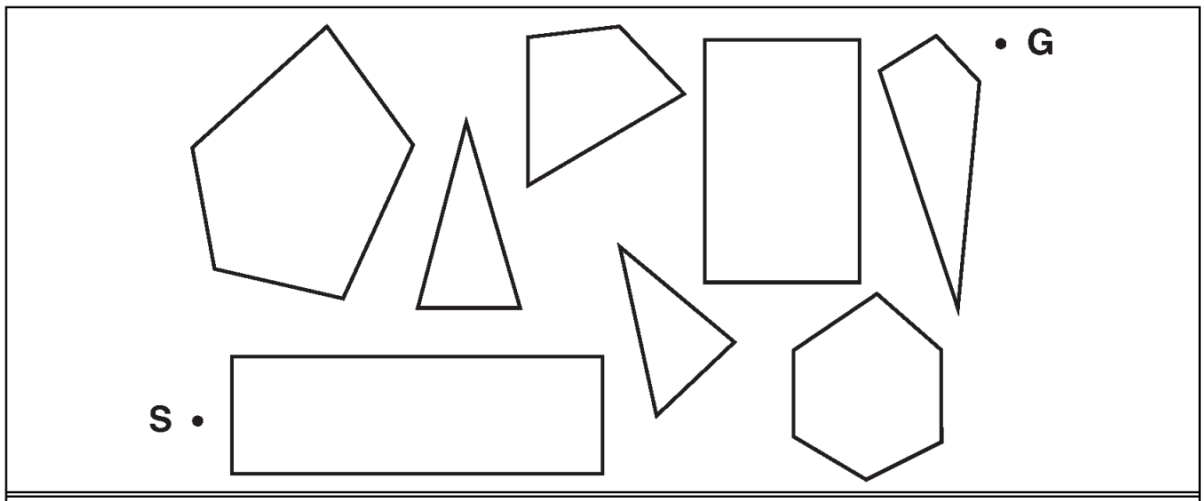

Figure 3.31 A scene with polygonal obstacles. S and G are the start and goal states.

6. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. [1]

   (a) Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

   (b) Solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?

   (c) Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

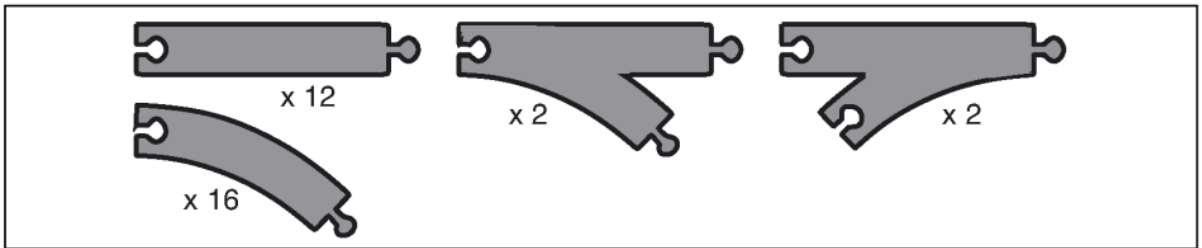

Figure 3.32 The track pieces in a wooden railway set; each is labeled with the number of copies in the set. Note that curved pieces and "fork" pieces ("switches" or "points") can be flipped over so they can curve in either direction. Each curve subtends 45 degrees.

7. A basic wooden railway set contains the pieces shown in Figure 3.32. The task is to connect these pieces into a railway that has no overlapping tracks and no loose ends where a train could run off onto the floor.

   (a) Suppose that the pieces fit together exactly with no slack. Give a precise formulation of the task as a search problem.

   (b) Identify a suitable uninformed search algorithm for this task and explain your choice.

   (c) Explain why removing any one of the "fork" pieces makes the problem unsolvable.

[1] This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

(d) Give an upper bound on the total size of the state space defined by your formulation. (Hint: think about the maximum branching factor for the construction process and the maximum depth, ignoring the problem of overlapping pieces and loose ends. Begin by pretending that every piece is unique.)

8. Consider a state space where the start state is number 1 and each state k has two successors: numbers $2k$ and $2k + 1$.

   (a) Draw the portion of the state space for states 1 to 15.

   (b) Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.

   (c) How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

   (d) Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

   (e) Call the action going from k to $2k$ Left, and the action going to $2k + 1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

9. An action such as Go(Sibiu) really consists of a long sequence of finer-grained actions: turn on the car, release the brake, accelerate forward, etc. Having composite actions of this kind reduces the number of steps in a solution sequence, thereby reducing the search time. Suppose we take this to the logical extreme, by making super-composite actions out of every possible sequence of Go actions. Then every problem instance is solved by a single super-composite action, such as Go(Sibiu)–Go(Rimnicu Vilcea)–Go(Pitesti)–Go(Bucharest). Explain how search would work in this formulation. Is this a practical approach for speeding up problem solving?

10. Define in your own words the following terms: state, state space, search tree, search node, goal, action, transition model, and branching factor.

11. What is the difference between a world state, a state description, and a search node? Why is this distinction useful?

12. Does a finite state space always lead to a finite search tree? How about a finite state space that is a tree? Can you be more precise about what types of state spaces always lead to finite search trees?

13. Consider the problem of find a path of links from one web URL to another. What is an appropriate search strategy? Is bidirectional search a good idea? Could a search engine be used to implement a predecessor function?

14. Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs $O(n)$).

3 Informed search

1. Suppose two friends live in different cities. On every turn, the two friends can simultaneously move to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

   (a) Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

   (b) Let $D(i, j)$ be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible?

      i. $D(i, j)$

ii. $2 \cdot D(i, j)$
iii. $D(i, j)/2$

(c) Are there completely connected maps for which no solution exists?

(d) Are there maps in which all solutions require one friend to visit the same city twice?

2. Consider a regular infinite 2D grid. The start state is at the origin, (0,0), and the goal state is at (m, n).

(a) What is the branching factor b in this state space?

(b) How many distinct states are there at depth k (for $k > 0$)?

(c) What is the maximum number of nodes expanded by breadth-first tree search?

(d) What is the maximum number of nodes expanded by breadth-first graph search?

(e) Is $h = |u - m| + |v - n|$ an admissible heuristic for a state at (u, v)? Explain.

(f) How many nodes are expanded by A^* graph search using h?

(g) Does h remain admissible if some links are removed?

(h) Does h remain admissible if some links are added between nonadjacent states?

3. Consider the problem of moving k knights from k starting squares $s_1, \ldots, s_k$ to k goal squares $g_1, \ldots, g_k$, on an unbounded chessboard, subject to the rule that no two knights can land on the same square at the same time. Each action consists of moving up to k knights simultaneously. We would like to complete the maneuver in the smallest number of actions.

(a) What is the maximum branching factor in this state space, expressed as a function of k?

(b) Suppose h_i is an admissible heuristic for the problem of moving knight i to goal g_i by itself. Which of the following heuristics are admissible for the k-knight problem? Of those, which is the best?

i. $\sum_{i=1}^{n} h_i$
ii. $\max h_1, ..., h_n$.
iii. $\min h_1, ..., h_n$.

(c) Repeat (b) for the case where you are allowed to move only one knight at a time.

4. A set of n vehicles occupy squares $(1, 1)$ through $(n, 1)$ (i.e., the bottom row) of an $n \times n$ 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.

(a) Calculate the size of the state space as a function of n.

(b) Calculate the branching factor as a function of n.

(c) Suppose that vehicle i is at (x_i, y_i); write a nontrivial admissible heuristic h_i for the number of moves it will require to get to its goal location $(n - i + 1, n)$, assuming no other vehicles are on the grid.

(d) Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations? Explain.

i. $\sum_{i=1}^{n} h_i$
ii. $\max h_1, ..., h_n$.
iii. $\min h_1, ..., h_n$.

5. Can you draw some conclusions comparing the answers to the previous two exercises?

6. Trace the operation of A^* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f, g and h score for each node.

7. We saw that the straight-line distance heuristic leads greedy best-first search astray on the problem of going from Iasi to Fagaras. However, the heuristic is perfect on the opposite problem: going from Fagaras to Iasi. Are there problems for which the heuristic is misleading in both directions?

8. Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

9. The heuristic path algorithm (Pohl, 1977) is a best-first search in which the evaluation function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this complete? For what values is it optimal, assuming that h is admissible? What kind of search does this perform for $w = 0, w = 1, andw = 2$?

10. Devise a state space in which A^* using returns a suboptimal solution with an h function that is admissible but inconsistent.

11. Sometimes there is no good evaluation function for a problem but there is a good comparison method: a way to tell whether one node is better than another without assigning numerical values to either. Show that this is enough to do a best-first search. Is there an analog of A^* for this setting?

12. Which of the following are true and which are false? Explain your answers.

    (a) Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic.

    (b) $h(n) = 0$ is an admissible heuristic for the 8-puzzle.

    (c) A^* is of no use in robotics because percepts, states, and actions are continuous.

    (d) Breadth-first search is complete even if zero step costs are allowed.

    (e) 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

13. Prove each of the following statements, or give a counterexample:

    (a) Breadth-first search is a special case of uniform-cost search.

    (b) Depth-first search is a special case of best-first tree search.

    (c) Uniform-cost search is a special case of A^* search.