search-exercises

March 26, 2019

1 3. Solving Problems By Searching

3.1 Explain why problem formulation must follow goal formulation.

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

1. 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.
2. 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.
3. 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.
4. 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.

3.3 Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

1. Formulate this problem. How large is the state space?
2. 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?
3. 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?

4. 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.

3.4 You have 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.

1. Formulate this problem in the straightforward way. Compute the size of the state space.
2. 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?
3. 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.
4. How many solutions does this problem have?
5. 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)?

3.5 [two-friends-exercise]Suppose two friends live in different cities on a map, such as the Romania map shown in . On every turn, we can simultaneously move each friend 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.

1. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)
2. 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) $D(i,j)/2$.
3. Are there completely connected maps for which no solution exists?
4. Are there maps in which all solutions require one friend to visit the same city twice?

3.6 [8puzzle-parity-exercise] Show that the 8-puzzle states are divided into two disjoint sets, such that any state is reachable from any other state in the same set, while no state is reachable from any state in the other set. (*Hint:* See @Berlekamp+al:1982.) Devise a procedure to decide which set a given state is in, and explain why this is useful for generating random states.

3.7 [nqueens-size-exercise] Consider the n-queens problem using the "efficient" incremental formulation given on page Section **??**. Explain why the state space has at least $\sqrt[3]{n!}$ states and estimate the largest n for which exhaustive exploration is feasible. (*Hint*: Derive a lower bound on the branching factor by considering the maximum number of squares that a queen can attack in any column.)

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

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

2. 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.

3. 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.

4. You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, 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 gallon.

3.9 [path-planning-exercise]Consider the problem of finding the shortest path between two points on a plane that has convex polygonal obstacles as shown in . This is an idealization of the problem that a robot has to solve to navigate in a crowded environment.

1. 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?

2. 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?

3. 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.

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

3.10 [negative-g-exercise]On page Section **??**, we said that we would not consider problems with negative path costs. In this exercise, we explore this decision in more depth.

1. Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.

2. Does it help if we insist that step costs must be greater than or equal to some negative constant c? Consider both trees and graphs.

3. Suppose that a set of actions forms a loop in the state space such that executing the set in some order results in no net change to the state. If all of these actions have negative cost, what does this imply about the optimal behavior for an agent in such an environment?

4. One can easily imagine actions with high negative cost, even in domains such as route finding. For example, some stretches of road might have such beautiful scenery as to far outweigh the normal costs in terms of time and fuel. Explain, in precise terms, within the context of state-space search, why humans do not drive around scenic loops indefinitely, and explain how to define the state space and actions for route finding so that artificial agents can also avoid looping.

5. Can you think of a real domain in which step costs are such as to cause looping?

3.11 [mc-problem] The problem is usually stated as follows. 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. 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.

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

2. Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?

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

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

3.13 What's the difference between a world state, a state description, and a search node? Why is this distinction useful?

3.14 An action such as 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 actions. Then every problem instance is solved by a single super-composite action, such as . Explain how search would work in this formulation. Is this a practical approach for speeding up problem solving?

3.15 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? (Adapted from , 1996.)

3.16 [graph-separation-property-exercise] Prove that satisfies the graph separation property illustrated in . (*Hint*: Begin by showing that the property holds at the start, then show that if it holds before an iteration of the algorithm, it holds afterwards.) Describe a search algorithm that violates the property.

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

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.

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

1. Draw the portion of the state space for states 1 to 15.
2. 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.
3. How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?
4. 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?
5. 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?

3.19 [brio-exercise]A basic wooden railway set contains the pieces shown in . 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.

1. Suppose that the pieces fit together *exactly* with no slack. Give a precise formulation of the task as a search problem.
2. Identify a suitable uninformed search algorithm for this task and explain your choice.
3. Explain why removing any one of the "fork" pieces makes the problem unsolvable.
4. 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.)

3.20 Implement two versions of the function for the 8-puzzle: one that copies and edits the data structure for the parent node s and one that modifies the parent state directly (undoing the modifications as needed). Write versions of iterative deepening depth-first search that use these functions and compare their performance.

3.21 [iterative-lengthening-exercise]On page Section **??**, we mentioned **iterative lengthening search**, an iterative analog of uniform cost search. The idea is to use increasing limits on path cost. If a node is generated whose path cost exceeds the current limit, it is immediately discarded. For each new iteration, the limit is set to the lowest path cost of any node discarded in the previous iteration.

1. Show that this algorithm is optimal for general path costs.
2. Consider a uniform tree with branching factor b, solution depth d, and unit step costs. How many iterations will iterative lengthening require?
3. Now consider step costs drawn from the continuous range $[\epsilon, 1]$, where $0 < \epsilon < 1$. How many iterations are required in the worst case?

4. Implement the algorithm and apply it to instances of the 8-puzzle and traveling salesperson problems. Compare the algorithm's performance to that of uniform-cost search, and comment on your results.

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

3.23 Write a program that will take as input two Web page URLs and find a path of links from one to the other. What is an appropriate search strategy? Is bidirectional search a good idea? Could a search engine be used to implement a predecessor function?

3.24 [vacuum-search-exercise]Consider the vacuum-world problem defined in .

1. Which of the algorithms defined in this chapter would be appropriate for this problem? Should the algorithm use tree search or graph search?

2. Apply your chosen algorithm to compute an optimal sequence of actions for a 3×3 world whose initial state has dirt in the three top squares and the agent in the center.

3. Construct a search agent for the vacuum world, and evaluate its performance in a set of 3×3 worlds with probability 0.2 of dirt in each square. Include the search cost as well as path cost in the performance measure, using a reasonable exchange rate.

4. Compare your best search agent with a simple randomized reflex agent that sucks if there is dirt and otherwise moves randomly.

5. Consider what would happen if the world were enlarged to $n \times n$. How does the performance of the search agent and of the reflex agent vary with n?

3.25 [search-special-case-exercise] Prove each of the following statements, or give a counterexample:

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

2. Depth-first search is a special case of best-first tree search.

3. Uniform-cost search is a special case of A search.

3.26 Compare the performance of A and RBFS on a set of randomly generated problems in the 8-puzzle (with Manhattan distance) and TSP (with MST—see) domains. Discuss your results. What happens to the performance of RBFS when a small random number is added to the heuristic values in the 8-puzzle domain?

3.27 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.

3.28 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?

3.29 [a*-failure-exercise]Devise a state space in which A using returns a suboptimal solution with an $h(n)$ function that is admissible but inconsistent.

3.30 Accurate heuristics don't necessarily reduce search time in the worst case. Given any depth d, define a search problem with a goal node at depth d, and write a heuristic function such that $|h(n) - h^*(n)| \leq O(\log h^*(n))$ but A* expands all nodes of depth less than d.

3.31 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 $w = 0$, $w = 1$, and $w = 2$?

3.32 Consider the unbounded version of the regular 2D grid shown in . The start state is at the origin, (0,0), and the goal state is at (x, y).

1. What is the branching factor b in this state space?
2. How many distinct states are there at depth k (for $k > 0$)?
3. What is the maximum number of nodes expanded by breadth-first tree search?
4. What is the maximum number of nodes expanded by breadth-first graph search?
5. Is $h = |u - x| + |v - y|$ an admissible heuristic for a state at (u, v)? Explain.
6. How many nodes are expanded by A graph search using h?
7. Does h remain admissible if some links are removed?
8. Does h remain admissible if some links are added between nonadjacent states?

3.33 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; 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_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.
4. Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations? Explain.
    1. $\sum_{i=1}^{n} h_i$.
    2. $\max\{h_1, \ldots, h_n\}$.
    3. $\min\{h_1, \ldots, h_n\}$.

3.34 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.

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

2. 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?

    1. $\min\{h_1, \ldots, h_k\}$.
    2. $\max\{h_1, \ldots, h_k\}$.
    3. $\sum_{i=1}^{k} h_i$.

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

3.35 We saw on page Section **??** 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?

3.36 Invent a heuristic function for the 8-puzzle that sometimes overestimates, and show how it can lead to a suboptimal solution on a particular problem. (You can use a computer to help if you want.) Prove that if h never overestimates by more than c, A using h returns a solution whose cost exceeds that of the optimal solution by no more than c.

3.37 [consistent-heuristic-exercise]Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

3.38 [tsp-mst-exercise]The traveling salesperson problem (TSP) can be solved with the minimum-spanning-tree (MST) heuristic, which estimates the cost of completing a tour, given that a partial tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

1. Show how this heuristic can be derived from a relaxed version of the TSP.

2. Show that the MST heuristic dominates straight-line distance.

3. Write a problem generator for instances of the TSP where cities are represented by random points in the unit square.

4. Find an efficient algorithm in the literature for constructing the MST, and use it with A graph search to solve instances of the TSP.

3.39 [Gaschnig-h-exercise]On page Section **??** , we defined the relaxation of the 8-puzzle in which a tile can move from square A to square B if B is blank. The exact solution of this problem defines **Gaschnig's heuristic** @Gaschnig:1979. Explain why Gaschnig's heuristic is at least as accurate as h_1 (misplaced tiles), and show cases where it is more accurate than both h_1 and h_2 (Manhattan distance). Explain how to calculate Gaschnig's heuristic efficiently.

3.40 We gave two simple heuristics for the 8-puzzle: Manhattan distance and misplaced tiles. Several heuristics in the literature purport to improve on this—see, for example, @Nilsson:1971, @Mostow+Prieditis:1989, and @Hansson+al:1992. Test these claims by implementing the heuristics and comparing the performance of the resulting algorithms.