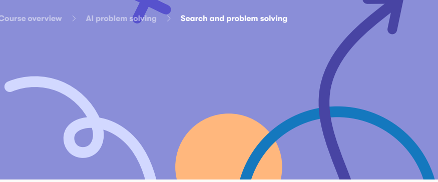
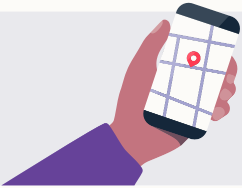


I. Search and problem solving

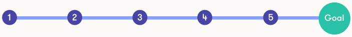
Many problems can be phrased as search problems. This requires that we start by formulating the alternative choices and their consequences.

Search in practice: getting from A to B

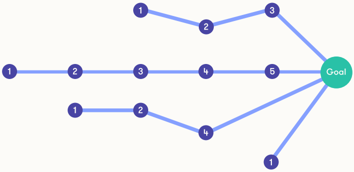
Imagine you're in a foreign city, at some address (say a hotel) and want to use public transport to get to another address (a nice restaurant, perhaps). What do you do? If you are like many people, you pull out your smartphone, type in the destination and start following the instructions.



This question belongs to the class of search and planning problems. Similar problems need to be solved by self-driving cars, and (perhaps less obviously) AI for playing games. In the game of chess, for example, the difficulty is not so much in getting a piece from A to B as keeping your pieces safe from the opponent.



Often there are many different ways to solve the problem, some of which may be more preferable in terms of time, effort, cost or other criteria. Different search techniques may lead to different solutions, and developing advanced search algorithms is an established research area.



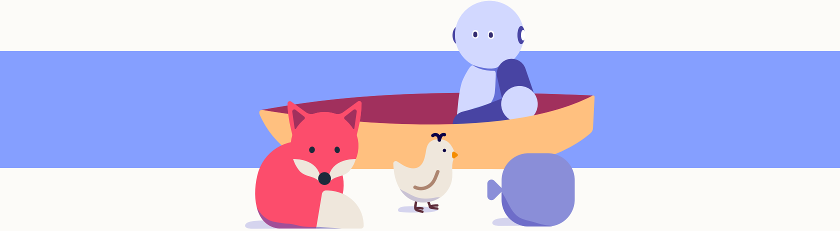
We will not focus on the actual search algorithms. Instead, we emphasize the first stage of the problem solving process: defining the choices and their consequences, which is often far from trivial and can require careful thinking. We also need to define what our goal is, or in other words, when we can consider the problem solved. After this has been done, we can look for a sequence of actions that leads from the initial state to the goal.

In this chapter, we will discuss two kinds of problems:

- Search and planning in static environments with only one "agent"
- Games with two-players ("agents") competing against each other

These categories don't cover all possible real-world scenarios, but they are generic enough to demonstrate the main concepts and techniques.

Before we address complex search tasks like navigation or playing chess, let us start from a much simplified model in order to build up our understanding of how we can solve problems by AI.



Toy problem: chicken crossing

We'll start from a simple puzzle to illustrate the ideas. A robot on a rowboat needs to move three pieces of cargo across a river: a fox, a chicken, and a sack of chicken-feed. The fox will eat the chicken if it has the chance, and the chicken will eat the chicken-feed if it has the chance, and neither is a desirable outcome. The robot is capable of keeping the animals from doing harm when it is near them, but only the robot can operate the rowboat and only two of the pieces of cargo can fit on the rowboat together with the robot. How can the robot move all of its cargo to the opposite bank of the river?

Note

The easy version of the rowboat puzzle

If you have heard this riddle before, you might know that it can be solved even with less space on the boat. That will be an exercise for you after we solve this easier version together.

We will model the puzzle by noting that five movable things have been identified: the robot, the rowboat, the fox, the chicken, and the chicken-feed. In principle, each of the five can be on either side of the river, but since only the robot can operate the rowboat, the two will always be on the same side. Thus there are four things with two possible positions for each, which makes for sixteen combinations, which we will call states:

States of the chicken crossing puzzle

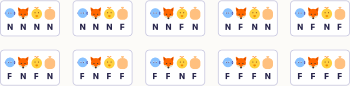
State	Robot	Fox	Chicken	Chicken-feed
NNNN	Near side	Near side	Near side	Near side
NNNF	Near side	Near side	Near side	Far side
NNFN	Near side	Near side	Far side	Near side
NNFF	Near side	Near side	Far side	Far side
NFNN	Near side	Far side	Near side	Near side
NFNF	Near side	Far side	Near side	Far side
NFFN	Near side	Far side	Far side	Near side
NFFF	Near side	Far side	Far side	Far side
FNNN	Far side	Near side	Near side	Near side
FNNF	Far side	Near side	Near side	Far side
FNFN	Far side	Near side	Far side	Near side
FNFF	Far side	Near side	Far side	Far side
FFNN	Far side	Far side	Near side	Near side
FFNF	Far side	Far side	Near side	Far side
FFFN	Far side	Far side	Far side	Near side
FFFF	Far side	Far side	Far side	Far side

We have given short names to the states, because otherwise it would be cumbersome to talk about them. Now we can say that the starting state is NNNN and the goal state is FFFF, instead of something like “in the starting state, the robot is on the near side, the fox is on the near side, the chicken is on the near side, and also the chicken-feed is on the near side, and in the goal state the robot is on the far side”, and so on.

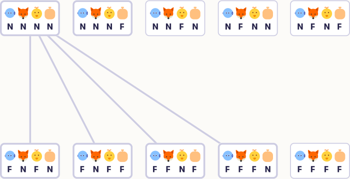
Some of these states are forbidden by the puzzle conditions. For example, in state NFFN (meaning that the robot is on the near side with the chicken-feed but the fox and the chicken are on the far side), the fox will eat the chicken, which we cannot have. Thus we can rule out states NFFN, NFFF, FNNF, FNNN, NNFF, and FFNN (you can check each one if you doubt our reasoning). We are left with the following ten states:

State	Robot	Fox	Chicken	Chicken-feed
NNNN	Near side	Near side	Near side	Near side
NNNF	Near side	Near side	Near side	Far side
NNFN	Near side	Near side	Far side	Near side
NFNN	Near side	Far side	Near side	Near side
NFNF	Near side	Far side	Near side	Far side
FNFN	Far side	Near side	Far side	Near side
FNFF	Far side	Near side	Far side	Far side
FFNF	Far side	Far side	Near side	Far side
FFFN	Far side	Far side	Far side	Near side
FFFF	Far side	Far side	Far side	Far side

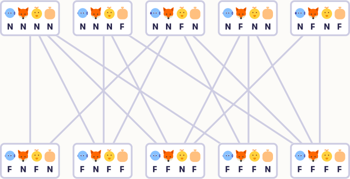
Next we will figure out which state transitions are possible, meaning simply that as the robot rows the boat with some of the items as cargo, what the resulting state is in each case. It’s best to draw a diagram of the transitions, and since in any transition the first letter alternates between N and F, it is convenient to draw the states starting with N (so the robot is on the near side) in one row and the states starting with F in another row:



Now let's draw the transitions. We could draw arrows that have a direction so that they point from one node to another, but in this puzzle the transitions are symmetric: if the robot can row from state NNNN to state FNFF, it can equally well row the other way from FNFF to NNNN. Thus it is simpler to draw the transitions simply with lines that don't have a direction. Starting from NNNN, we can go to FNFN, FNFF, FFNF, and FFFN:

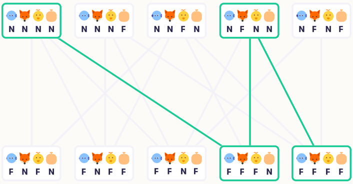


Then we fill in the rest:



We have now done quite a bit of work on the puzzle without seeming any closer to the solution, and there is little doubt that you could have solved the whole puzzle already by using your “natural intelligence”. But for more complex problems, where the number of possible solutions grows in the thousands and in the millions, our systematic or mechanical approach will shine since the hard part will be suitable for a simple computer to do. Now that we have formulated the alternative states and transitions between them, the rest becomes a mechanical task: find a path from the initial state NNNN to the final state FFFF.

One such path is colored in the following picture. The path proceeds from NNNN to FFFN (the robot takes the fox and the chicken to the other side), thence to NFNN (the robot takes the chicken back on the starting side) and finally to FFFF (the robot can now move the chicken and the chicken-feed to the other side).



State space, transitions, and costs

To formalize a planning problem, we use concepts such as the state space, transitions, and costs.

Key terminology

The state space

means the set of possible situations. In the chicken-crossing puzzle, the state space consisted of ten allowed states NNNN through to FFFF (but not for example NFFF, which the puzzle rules don´t allow). If the task is to navigate from place A to place B, the state space could be the set of locations defined by their (x,y) coordinates that can be reached from the starting point A. Or we could use a constrained set of locations, for example, different street addresses so that the number of possible states is limited.

Transitions

are possible moves between one state and another, such as NNNN to FNFN. It is important to note that we only count direct transitions that can be accomplished with a single action as transitions. A sequence of multiple transitions, for example, from A to C, from C to D, and from D to B (the goal), is a **path** rather than a transition.

Costs

refer to the fact that, oftentimes the different transitions aren´t all alike. They can differ in ways that make some transitions more preferable or cheaper (in a not necessarily monetary sense of the word) and others more costly. We can express this by associating with each transition a certain cost. If the goal is to minimize the total distance traveled, then a natural cost is the geographical distance between states. On the other hand, the goal could actually be to minimize the time instead of the distance, in which case the natural cost would obviously be the time. If all the transitions are equal, then we can ignore the costs.

Unanswered

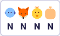
Exercise 5: A smaller rowboat

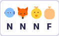
In the traditional version of this puzzle the robot can only fit one thing on the boat with it. The state space is still the same, but fewer transitions are possible.

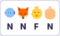
Using the diagram with the possible states below as a starting point, draw the possible transitions in it (it is MUCH easier to do this with a pencil and paper than without).

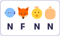
Having drawn the state transition diagram, **find the shortest path from NNNN to FFFF, and calculate the number of transitions on it.**

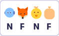
Please type your answer as the **number of transitions in the shortest path** (just a single number like "12"). Hint: Do *not* count the number of states, but the number of transitions. For example, the number of transitions in the path NNNN→FFNF→NFNF→FFFF is *3* instead of 4.

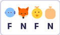


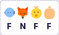


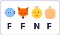


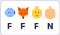


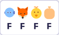












Answer

Submit

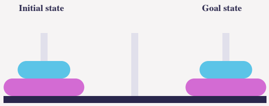
Unanswered

Exercise 6: The Towers of Hanoi

Let's do another puzzle: the well-known Towers of Hanoi. In our version, the puzzle involves three pegs, and two discs: one large, and one small (actually, there can be any number of discs but for the exercise, two is enough to demonstrate the principle).

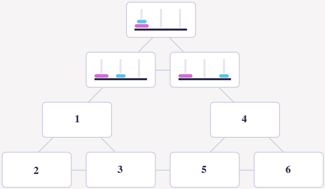
In the initial state, both discs are stacked in the first (leftmost) peg. The goal is to move the discs to the third peg. You can move one disc at a time, from any peg to another, as long as there is no other disc on top of it. It is not allowed to put a larger disc on top of a smaller disc.

This picture shows the initial state and the goal state. There are also seven other states so that the total number of possible states is nine: three ways to place the large disc and for each of them, three ways to place the small disc.

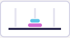


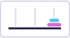
Your task: Draw the state diagram. The diagram should include all the nine possible states in the game, connected by lines that show the possible transitions. The picture below shows the overall structure of the state diagram and the positions of the first three states. It shows that from the starting state (at the top corner), you can move to two other states by moving the small disc. Complete the state diagram by placing the remaining states in the correct places. Note that the transitions are again symmetric and you can also move sideways (left or right) or up in the diagram.

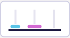
After solving the task using pen and paper, enter your solution by choosing which state belongs to which node in the diagram. (Hint: Each state belongs to exactly one node).

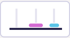


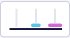
Choose for each node (1–6) in the above diagram the correct state A–F from below.

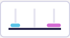

A


B


C


D


E


F

What state should be in box 1?	State A	State B	State C	State D	State E	State F
What state should be in box 2?	State A	State B	State C	State D	State E	State F
What state should be in box 3?	State A	State B	State C	State D	State E	State F
What state should be in box 4?	State A	State B	State C	State D	State E	State F
What state should be in box 5?	State A	State B	State C	State D	State E	State F
What state should be in box 6?	State A	State B	State C	State D	State E	State F

Submit

Next section

II. Solving problems with AI →

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Reaktor