CpSc810 – Goddard – Notes Chapter 1

Searching

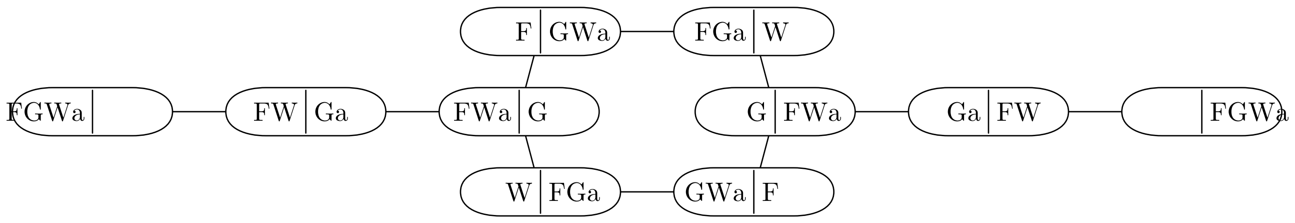
A common idea is searching for a solution. Exhaustive search is exhausting. So we must look for methods to reduce the work.

1.1 State-space Search

In ***state space search*** we explore the state space graph. In this graph, the nodes represent the distinct states in the problem solving process. The initial and goal states are specified. The arcs represent the possible transitions, actions or operators. We are looking for a goal, or maybe the best path to the goal.

> Fox-Goose-Wheat. *An agronomist wants to move herself, a silver fox, a fat goose, and some tasty wheat across a river. Unfortunately, her boat is so tiny she can take only one of her possessions across on any trip. Worse yet, an unattended fox will eat a goose, and an unattended goose will eat wheat, so the agronomist must not leave the fox alone with the goose nor the goose alone with the wheat. What is she to do?*

This is a representation in English. It takes a while to work out what is required, and then a while to solve. But is there a better description? Or a systematic way of finding such a solution? Try a diagram. Create a node for each safe position. Then draw the possible next moves by a link from the one to the other. The end result is a nice picture with 10 nodes shown here.

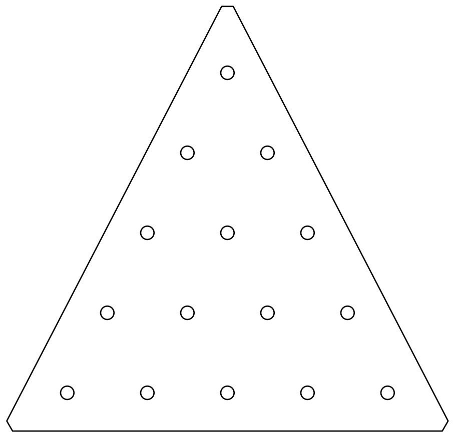


All one has to do is find a shortest path from the node with all the participants on one side to the node with all the participants on the other side. (If you can't do this by hand, then there's always Dijkstra's algorithm.)

The general approach then is to search the paths until either the goal is reached or all paths are abandoned. Note that for storage reasons we often only generate the graph on the fly. We need a control mechanism—a ***search algorithm***.

A general problem is that a state can be reached by several paths. Thus we would like to avoid loops and cycles. Directed graphs that are guaranteed not to have cycles are nice. For example, in playing a solitaire game whereby each turn a piece is added to the board, there is inexorable progress. When a tree results, so much the better.

> SOLITAIRE WITH 15 PEGS. *In a common puzzle, there is a wooden board with 15 holes arranged in a triangular grid. In 14 of these holes a peg is placed. The goal is to make moves to remove all but 1 peg. Each move consists of one peg jumping over another peg to the vacant hole directly on the other side of the peg, and removing the peg that was jumped (just like jumping in checkers).*



1.2 Depth-first and Breadth-first Search

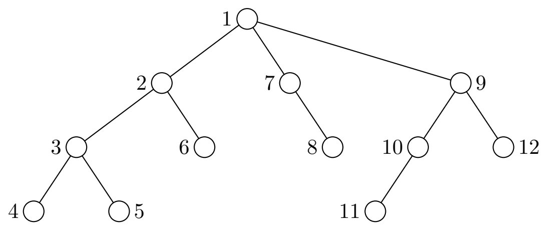
There are two blind procedures. Depth-first search dives into the search tree, while breadth-first search pushes uniformly into the search tree. While the underlying graph might be complex, the search procedures produce a tree.

In ***depth-first search***, at each node one immediately chooses a child to be explored. Other alternatives at the same level are, for the moment, ignored. However, when a dead end is reached, the process backtracks to the most recent ancestor node which still has an unexplored alternative.

DFS

```
• initialize Stack to have only root node
• while (Goal not reached) and (Stack not empty)
    • pop first node from Stack
    • push all successors onto Stack
• if Goal reached announce success
  else announce failure
```

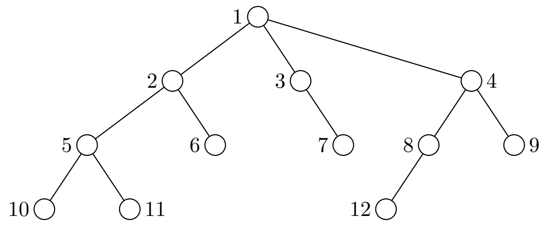
To retain the path, one remembers the parent of each node on the tree. Note that depth-first search is sometimes implemented using recursion rather than explicitly creating a stack. See the section on Constraint Satisfaction in a few pages. The following example shows the order in which nodes are explored.



In ***breadth-first search***, downward motion proceeds level by level until the goal is reached. At each node all children are explored before grandchildren.

BFS

```
• initialize Queue to have only root node
• while (Goal not reached) and (Queue not empty)
    • dequeue first node from Queue
    • append all successors to back of Queue
• if Goal reached announce success
  else announce failure
```



These methods are called ***blind*** or uninformed because they make no use of the structure of the problem. They are, however, very general. The advantage of depth-first search is that the memory requirements tend to be a lot less. The disadvantage is that it can get stuck in futile alleys and may never find the goal in a potentially infinite search space.

One method that has been proposed as a better blind method is ***depth first search with iterative deepening***. In this, one performs DFS to level 1. Then to level 2 and so on. Each time starting afresh but going to a limited depth in the graph. This does not require much more work than DFS, since most of the work is done at the last level and this is only explored once.

Reductions

With searching it often helps to avoid re-searching duplicate positions. There are two ideas to try. One is that, at the start, certain moves are ***symmetric***: trying on the left might clearly be the same as trying on the right. For example, there are only three distinct opening moves in tictactoe. The other idea is to remember, in some dictionary or hash table data structure, all states that have been seen. Of course, this adds the expense of looking up each position in the dictionary.

1.3 Heuristically-Informed Search

There are times when heuristics are called for. These are informed guesses: might do well, might be fallible. They have been very useful in game playing and expert systems. We start with their use in searching. The two ideas are: operator-ordering and state evaluation.

Hill Climbing

If you add quality measurements to DFS, you get ***hill climbing***. Here the successors of each node are assessed and the best one is chosen for exploration. The reason for the name is that this is like trying to get to the top of the hill by always heading up in the steepest direction.

```
HillClimb
• initialize stack to have only root node
• while (Goal not reached) and (Stack not empty)
    • pop first node from Stack
    • sort successors by estimated distance to Goal
    • push onto stack in order
• if Goal reached announce success
  else announce failure
```

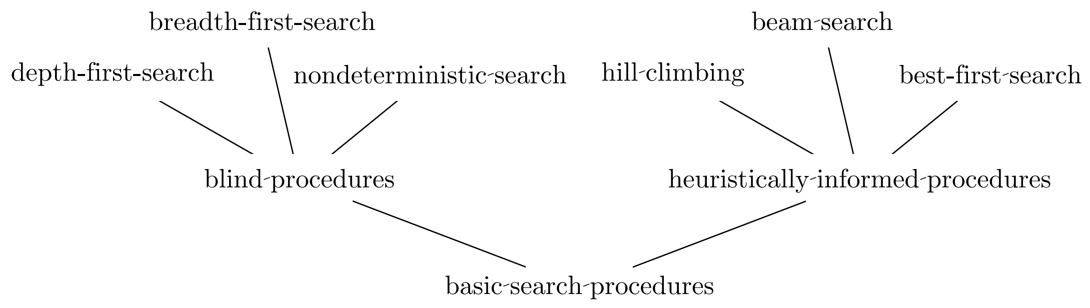
Of course one can run into problems. These are referred to as foothills (local maxima), plateaus (no direction is up) or narrow ridges.

Best-First Search

In ***best-first search***, quality measurements are taken more notice of. Instead of backtracking to the most recently created open node, one backtracks to the best open node so far, no matter where the node is in the developing tree.

You can add the pseudocode.

Winston summarizes the basic search procedures in the diagram below. Nondeterministic search chooses the next node to be expanded at random. Beam search looks only at a few successor nodes of each node. We do not explore these here.



Winston's mottoes:

"More knowledge generally leads to reduced search time"

"If you think you need a better search method, find another space to search instead."

Exercises

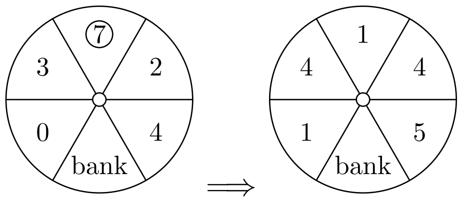
1.1. Felix has two weirdly shaped jugs: one holds exactly 3 liters and one exactly 4 liters, but they have no subdivision marking. A recipe calls for exactly 5 liters. He has a source of water but no other jug. Draw the state-space graph and determine the steps in getting exactly 5 liters that wastes the least amount of water.

1.2. A prince, a baron, a knight, a soldier and a serf arrive at a river bank to find a single boat that is capable of carrying at most three people. The problem is that if the serf is left alone with any single person he will escape, if the baron is left alone with the prince he will depose him, and if the soldier is left alone with the knight he will kill him. The problem is to get everyone across the river without any untoward things happening.

Use a graph and searching to solve this diplomatic problem.

1.3. **Project.** Mancala is played with 6 "holes" arranged in a circle. One hole is designated "bank". Each hole contains some number of "stones". In our simplified version, the player starts a move by picking up all the stones in one hole (but not the bank); she then proceeds clockwise (starting in the next hole) placing the stones one in each hole, until all the stones are placed. Note that the player gets to choose which of the 5 holes to start with; after that the process is mechanical.

Here is an example where the 7 stones are picked up from hole 3 and then redistributed.



The objective is to get all the stones into the bank. The game is played as a race between two players, and there are rules allowing interaction. But here we consider a single-player solitaire version.

Code up breadth-first-search to find the minimum sequence of moves to get all the stones into the bank. Use a hash table to avoid searching duplicate positions.