Week 20: Search I

Martha Lewis
(based on slides from Raul Santos Rodriguez)

Have a look at ...

... Russell and Norvig (Ch. 3)

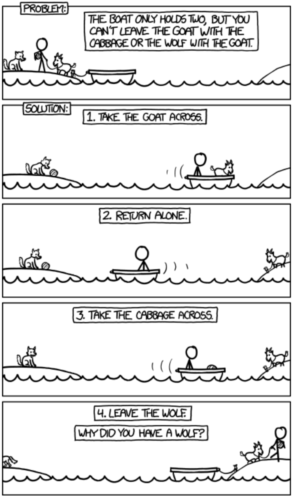
Question

A farmer wants to get his cabbage, goat, wolf across a river. He has a boat that only holds two. He cannot leave cabbage and goat alone or goat and wolf alone.

How many river crossings does he need?

- 4
- 5
- 6
- 7
- No solution

One solution...



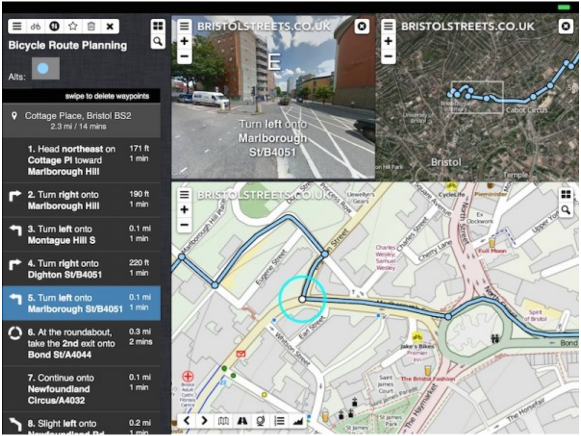
http://xkcd.com/1134/

Outline

Search or how to find a sequence of actions that achieves a goal when no single action will do. We will cover

- Backtracking search
- Depth-first search
- Breadth-first search
- DFS with iterative deepening

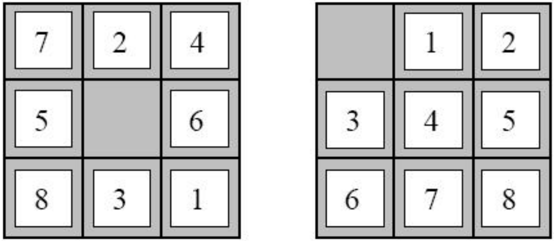
Application: route finding



Actions: go straight, turn left, turn right

Objective: shortest? fastest? most scenic?

Application: solving puzzles



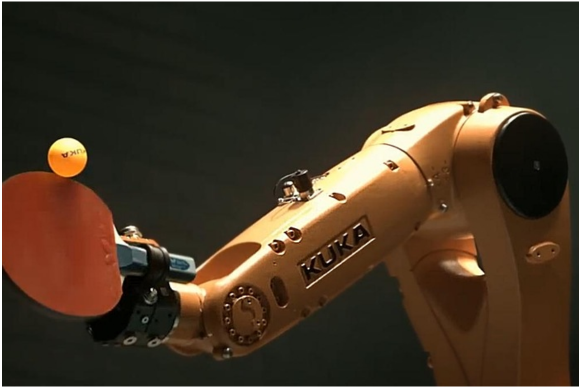
Start State   Goal State

8-puzzle

Actions: move pieces

Objective: reach a certain configuration

Application: robot planning



Actions: translate and rotate joints

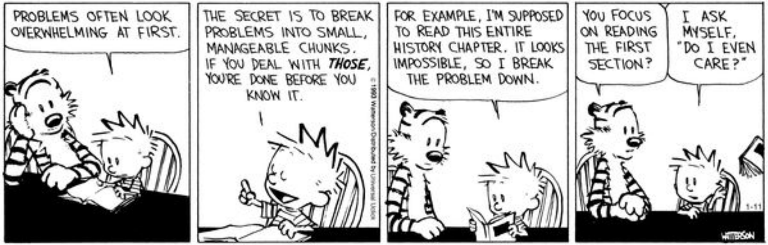
Objective: fastest? most energy efficient? safest?

Search

In all of these examples → sequence of actions

Idea

Decompose our very complex problem into small problems



Tree search: example

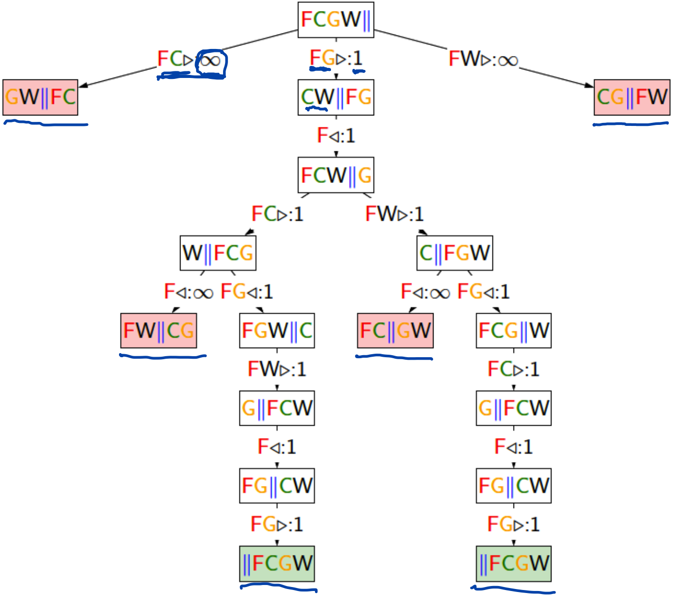
Farmer Cabbage Goat Wolf

Actions:

F▷	F◁
FC▷	FC◁
FG▷	FG◁
FW▷	FW◁

[Liang 2014]

Tree search: example



Search problem

Search algorithms require a structure to keep track of the search tree that is being constructed:

Search

- s_{start}: starting state
- Actions(s): possible actions
- Cost(s,a): action cost
- Succ(s,a): successors
- Goal(s): found solution?

Measuring performance

Completeness: Is the algorithm guaranteed to find a solution when there is one?

Optimality: Does the strategy find the optimal solution?

Time complexity: How long does it take to find a solution?

Space complexity: How much memory is needed to perform the search?

Complexity

Complexity is expressed in terms of three quantities:

- b, the branching factor or maximum number of successors of any node;
- d, the depth of the shallowest goal node (i.e., the number of steps along the path from the root);
- D, the maximum length of any path in the state space.

Backtracking search

BacktrackingSearch(s,path)

If Goal(s): update minimum cost path
Else for each action $a \in$ Actions(s):
  Extend path with Succ(s,a) and Cost(s,a)
  Call BacktrackingSearch(Succ(s,a), path)
Return minimum cost path

If b actions per state, maximum depth is D actions:

- Memory: $O(D)$ (small)
- Time: $O(b^D)$ (huge)

$1 + b + b^2 + \dots + b^D$ b^D

Depth-first search

Idea: Backtracking search + stop when find the first goal state

Assumption: Action costs Cost(s,a) = 0

If b actions per state, maximum depth is D actions:

- Memory: $O(D)$ (small)
- Time: $O(b^D)$ (huge) but that is just for the worst case (could be much better if solutions are easy to find)

DFS is great when there are an abundance of solutions

Depth-first search

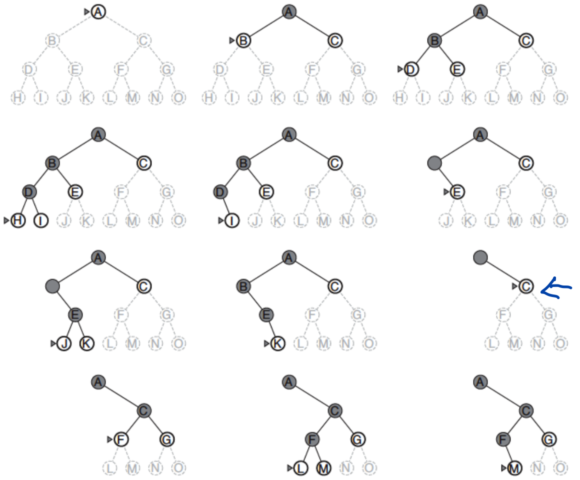


Figure 3.16 Depth-first search on a binary tree. The unexplored region is shown in light gray. Explored nodes with no descendants in the frontier are removed from memory. Nodes at depth 3 have no successors and M is the only goal node.

Breadth-first search

Idea: explore all nodes in order of increasing depth

Assumption: Action costs Cost(s,a)=c for some $c \geq 0$

If b actions per state, maximum depth is d actions:

- Memory: $O(b^d)$ (worse)
- Time: $O(b^d)$ (depends on d instead of D)

Breadth-first search

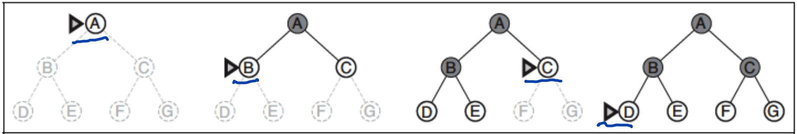


Figure 3.12 Breadth-first search on a simple binary tree. At each stage, the node to be expanded next is indicated by a marker.

[Russell and Norvig]

DFS with iterative deepening

Idea: Modify DFS to stop at a maximum depth

Idea: Call DFS for maximum depths $1, 2, \ldots$

Assumption: Action costs $\mathrm{Cost}(s, a) = c$ for some $c \geq 0$

If b actions per state, solution size d:

- Memory: $O(d)$ (better!)
- Time: $O(b^d)$ (same as BFS)

DFS with iterative deepening

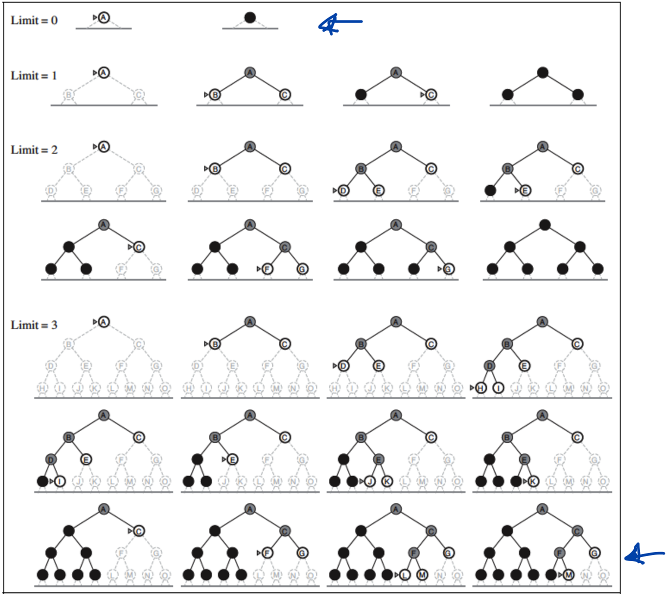


Figure 3.19 Four iterations of iterative deepening search on a binary tree.

Comparison

Algorithm	**Action costs**	**Space**	**Time**
DFS	zero	$O(D)$	$O(b^D)$
BFS	constant	$O(b^d)$	$O(b^d)$
DFS-ID	constant	$O(d)$	$O(b^d)$
Backtracking	any	$O(D)$	$O(b^D)$

b actions per state, solution depth d, maximum depth D

Summary

- Backtracking search - traverse all nodes of the graph to find the optimal route to the goal
- Depth first search - traverse a search tree by going deeper first
- Breadth-first search - traverse a tree by going broader first
- Depth first search with iterative deepening - increase the depth of the tree step by step
- Think about whether the algorithms are complete, optimal, and their complexity in time and space