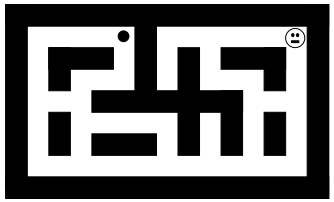
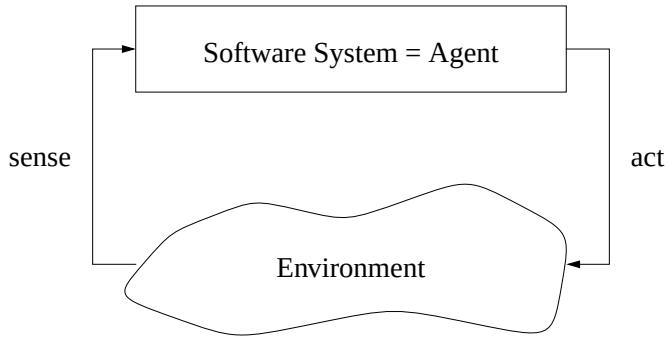
Artificial Intelligence
Uninformed Search

Nilsson - Chapters 7 and 8
Russell and Norvig - Chapter 3

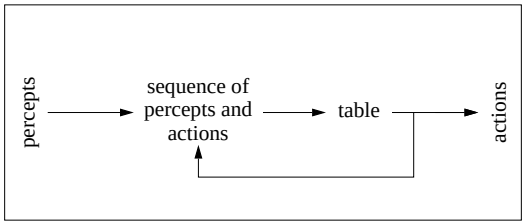


agents



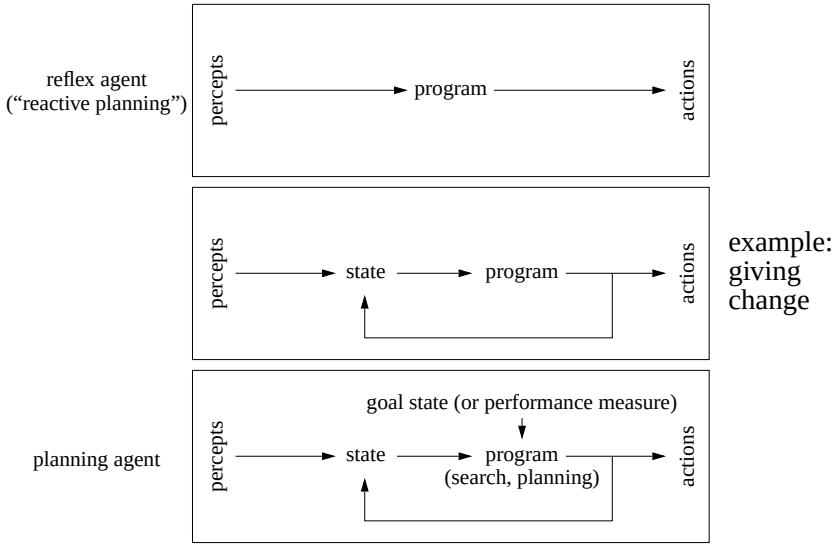
Performance Measure

agent architectures



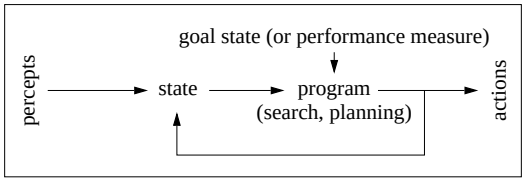
Gold Standard

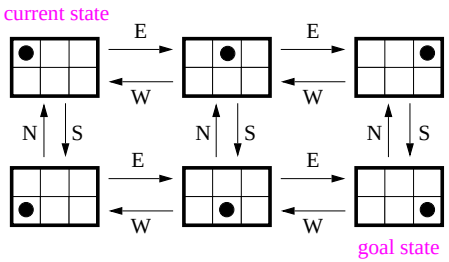
- huge table
- inflexible



possible agent architectures (agent = architecture + program)

state spaces





state space

examples of state spaces

eight puzzle

states?
actions?

examples of state spaces

missionaries and cannibals

Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find the quickest 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.

states?
actions?

examples of state spaces

travelling salesperson problem

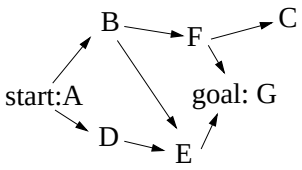
Given a map with n cities. Find a shortest tour that visits each city once and then returns to the start city.

states?
actions?

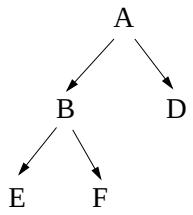
search problems

graph search	search problem
graph	state space
vertices	states
edges	actions = operators = successor function
edge costs	action costs
start vertex	start state
goal vertices	goal states or goal test
solution is a (minimum cost) path from the start vertex to the goal vertex	solution is a (minimum cost) action sequence from the start state to the goal state (or a state that satisfies the goal test)
	abstraction

state space



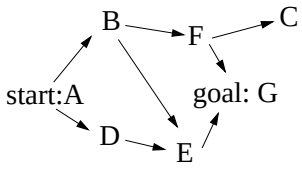
search tree



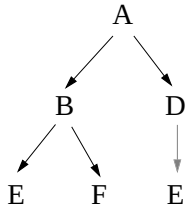
terminology:
- successor states

terminology:
- node (parent node, child node)
- leaf
- fringe = frontier
- depth
- node expansion

state space



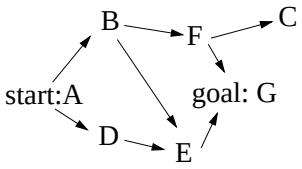
search tree



start with a tree that contains only the start state
pick a fringe node n
if fringe node n represents a goal state: stop
expand fringe node n
go to

uninformed (blind) search vs. informed (heuristic) search
today next time

breadth-first search



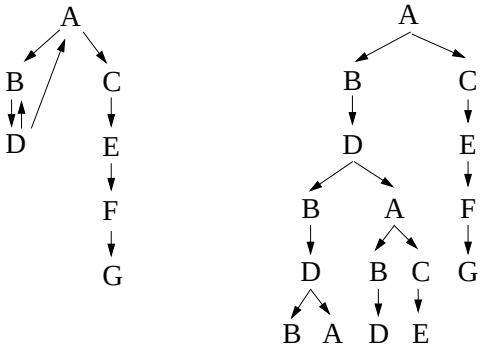
search tree?

start with a tree that contains only the start state
pick a fringe node n **with the smallest depth**
if fringe node n represents a goal state: stop
expand fringe node n
go to

non-uniform costs: uniform-cost search

breadth-first search: avoiding repeated states

not necessary, but can increase efficiency

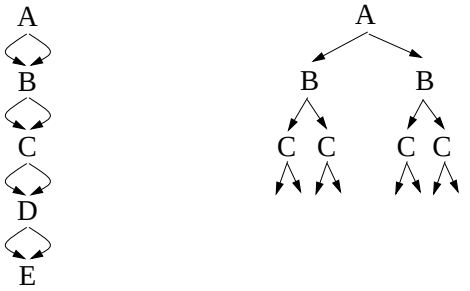


etc.

do not return to the state you just came from
do not create paths with cycles in them

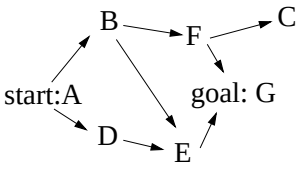
breadth-first search: avoiding repeated states

not necessary, but can increase efficiency



do not generate any state that was ever generated before

depth-first search

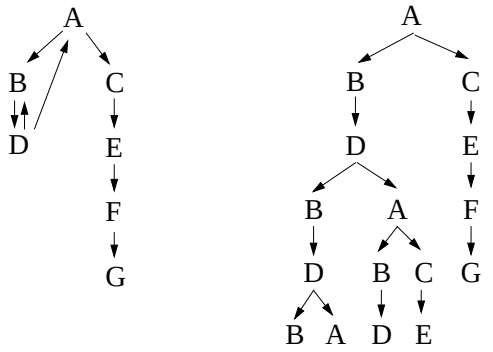


search tree?

start with a tree that contains only the start state
pick a fringe node n **with the largest depth**
if fringe node n represents a goal state: stop
expand fringe node n
go to

depth-first search: avoiding repeated states

guarantees termination in finite state spaces?

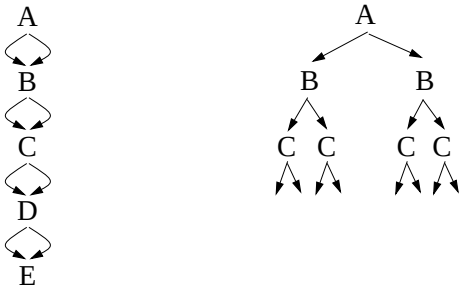


etc.

do not return to the state you just came from
do not create paths with cycles in them

depth-first search: avoiding repeated states

cannot be done?



do not generate any state that was ever generated before

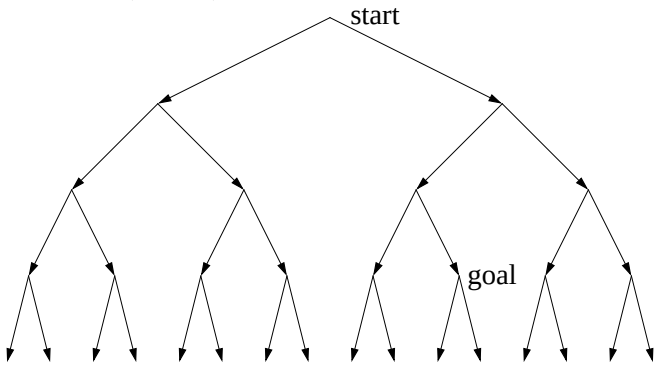
	breadth-first	depth-first	iterative-deepening
completeness			
optimality			
space complexity (= max nodes in memory)			
time complexity (= max nodes expanded)			

all numbers are “tight on the order of ...”

branching factor b
depth of goal d
depth of tree m

space and time complexity

state space = uniform tree
branching factor b (here: 2)
depth of goal d (here: 3)
depth of tree m (here: 4)

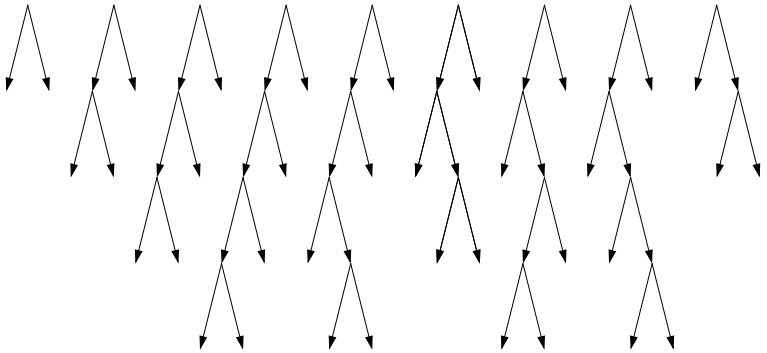


breadth-first search

depth	nodes	time	memory
0	1	1 millisecond	100 bytes
2	111	0.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabytes
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

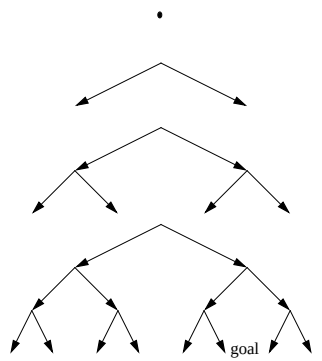
b = 10; 1000 nodes per second; 100 bytes per node

depth-first search



the best of both worlds: iterative-deepening depth-first search

limit = 0
perform depth-first search with depth limit *limit*
if goal found: stop
limit := *limit* + 1
go to



	breadth-first	depth-first	iterative-deepening
completeness	yes	no	yes
optimality	yes	no	yes
space complexity	b^d	bm	bd
time complexity	b^d	b^m	b^d

all numbers are “tight on the order of ...”

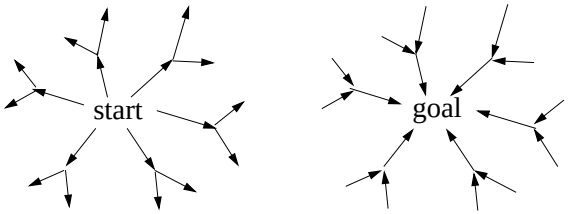
branching factor b
depth of goal d
depth of tree m

forward search (= search from start to goal)

backward search (= search from goal to start)

forward search is better than backward search if
-
-
-
-
-

combining forward and backward search: bi-directional search



need to be able to test for intersection of the search trees

	breadth-first	bi-directional
completeness	yes	yes
optimality	yes	yes
space complexity	b^d	$b^{d/2}$
time complexity	b^d	$b^{d/2}$

all numbers are “tight on the order of ...”

branching factor b
depth of goal d
depth of tree m