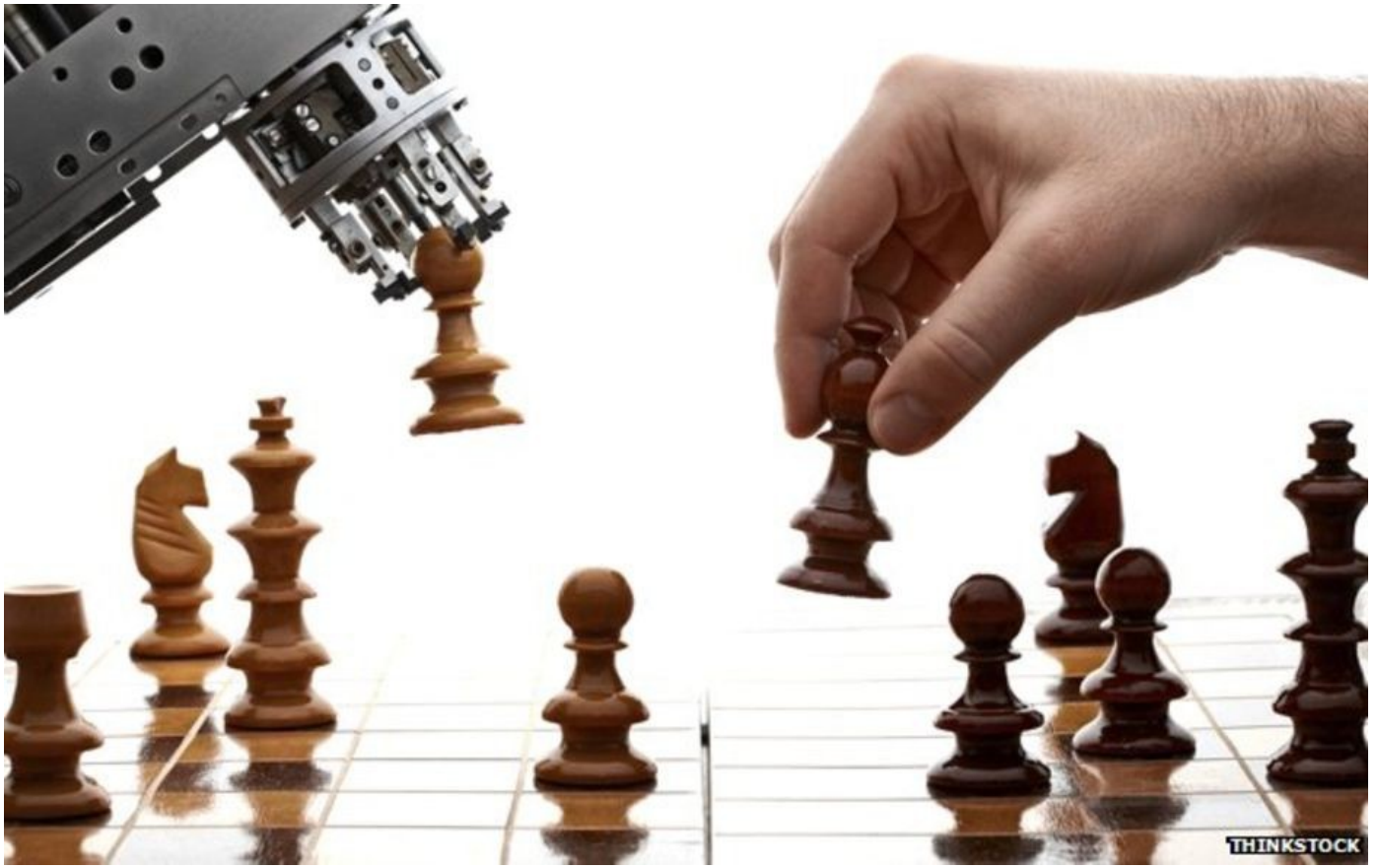


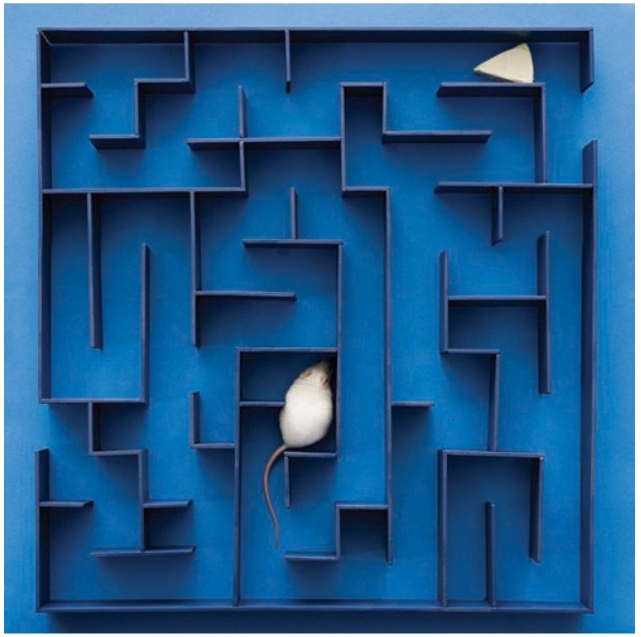
Introduction to Artificial Intelligence

COSC 4550 / COSC 5550

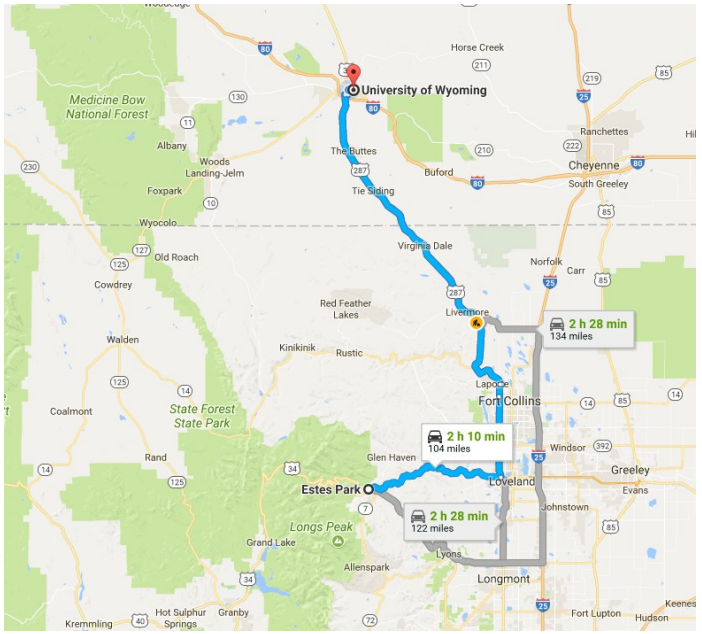
Professor Cheney

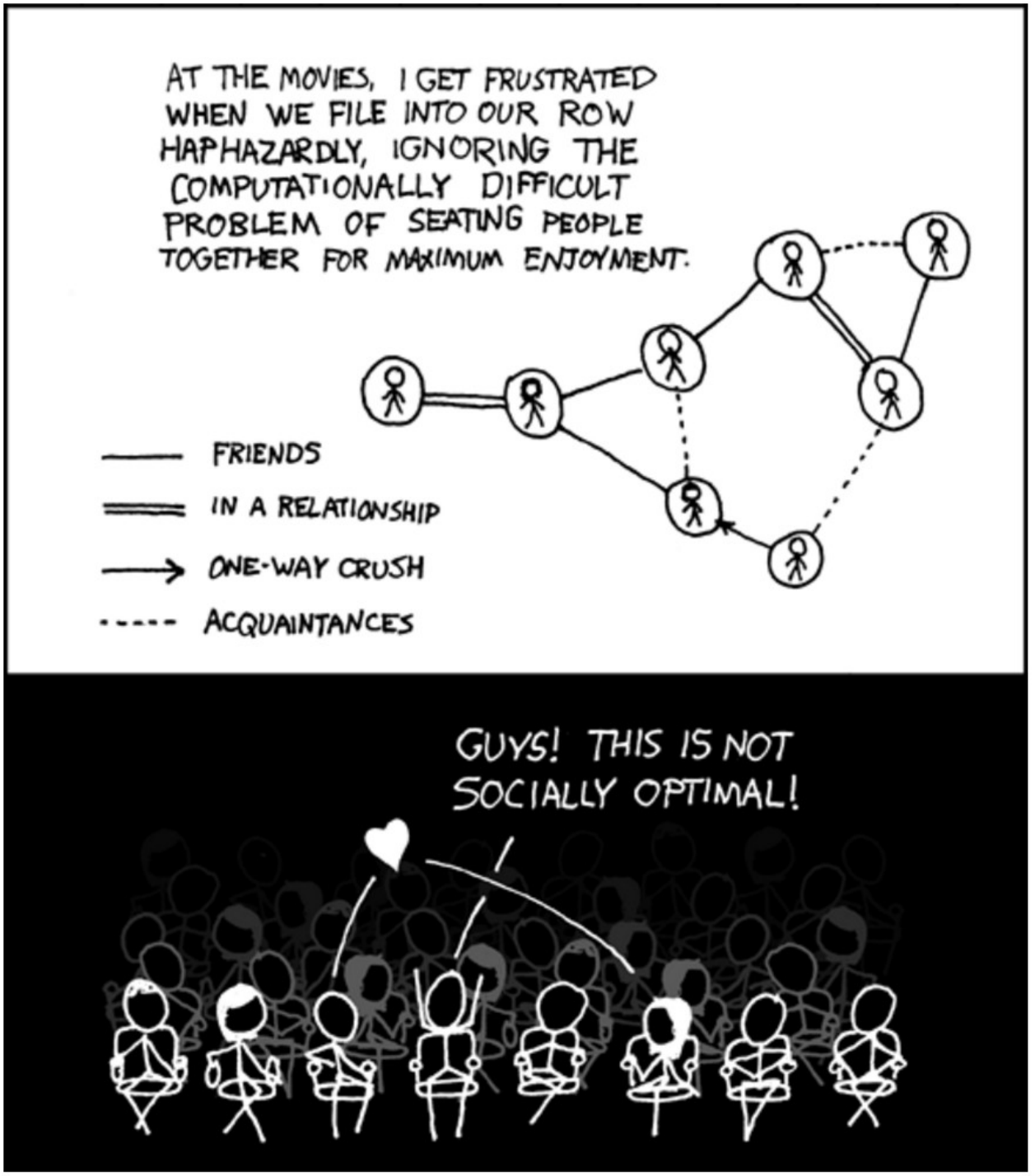
9/8/17

Search



Hugely important over many varied domains



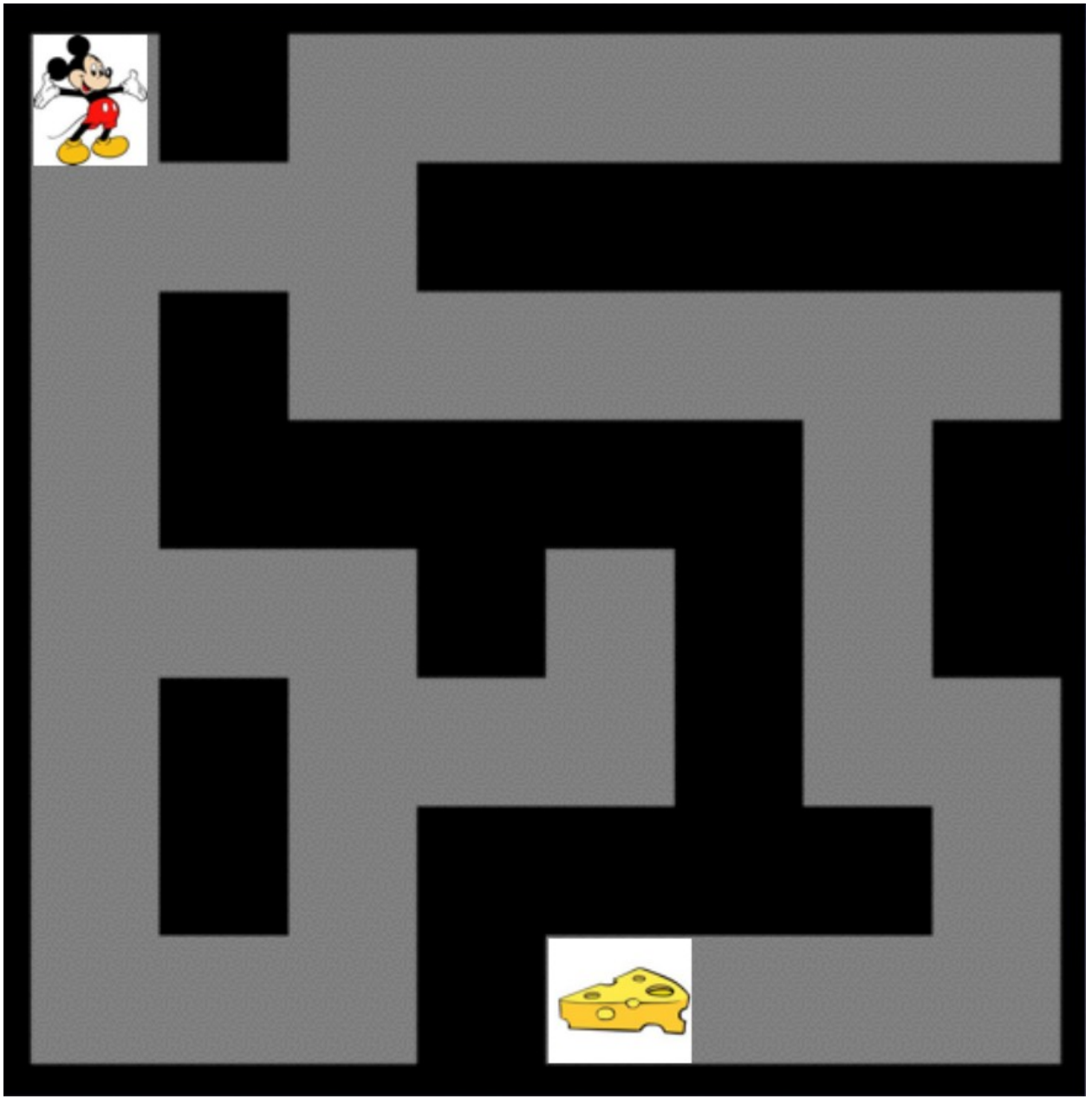

AT THE MOVIES, I GET FRUSTRATED WHEN WE FILE INTO OUR ROW HAPHAZARDLY, IGNORING THE COMPUTATIONALLY DIFFICULT PROBLEM OF SEATING PEOPLE TOGETHER FOR MAXIMUM ENJOYMENT.
FRIENDS
IN A RELATIONSHIP
ONE-WAY CRUSH
ACQUAINTANCES
GUYS! THIS IS NOT SOCIALLY OPTIMAL!

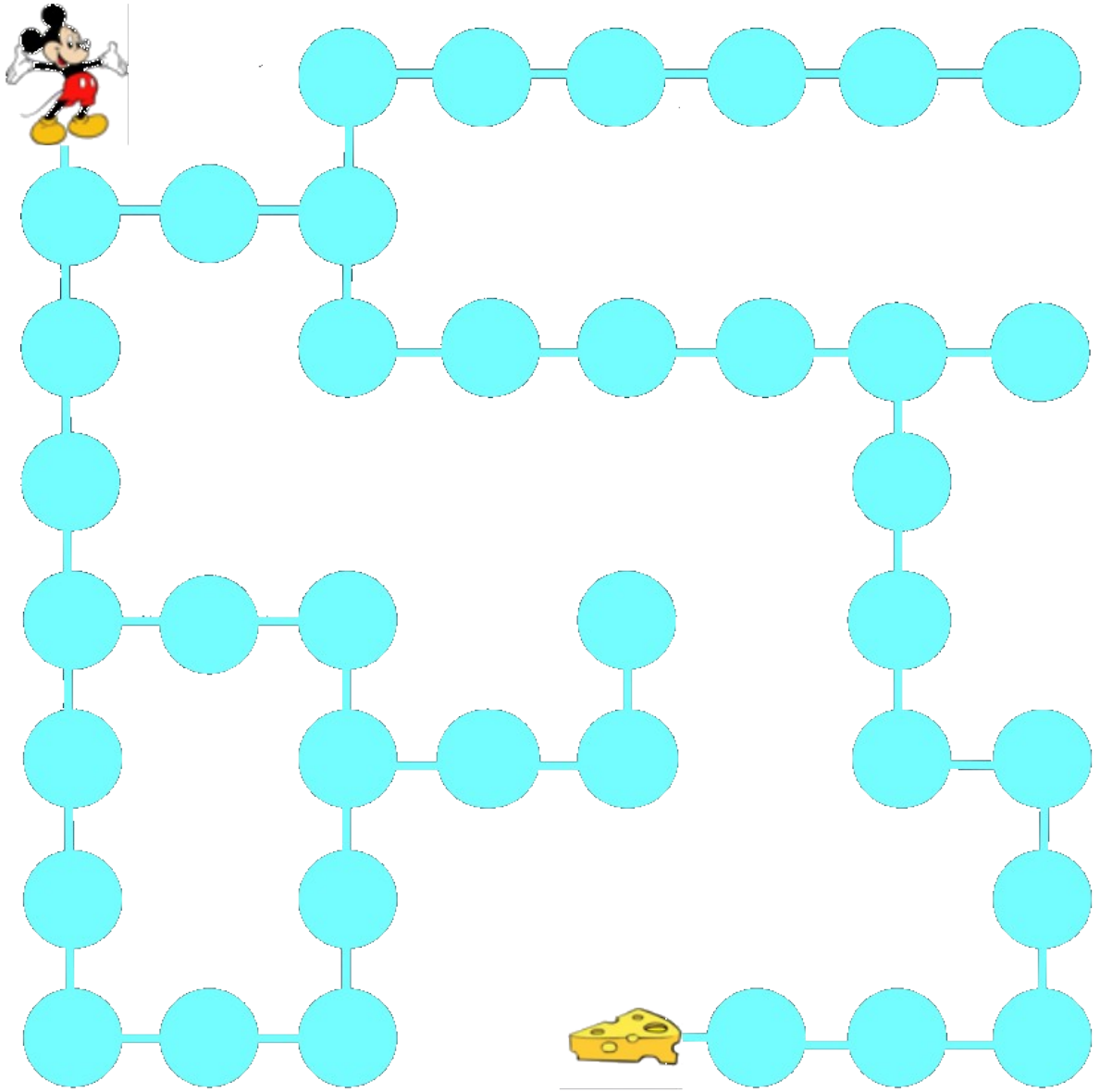
systematic way of looking for ~~things~~
optimal solutions

a series of decisions (actions)
to reach a desired goal (reward)

[based on available information (state)]

Graph Representation

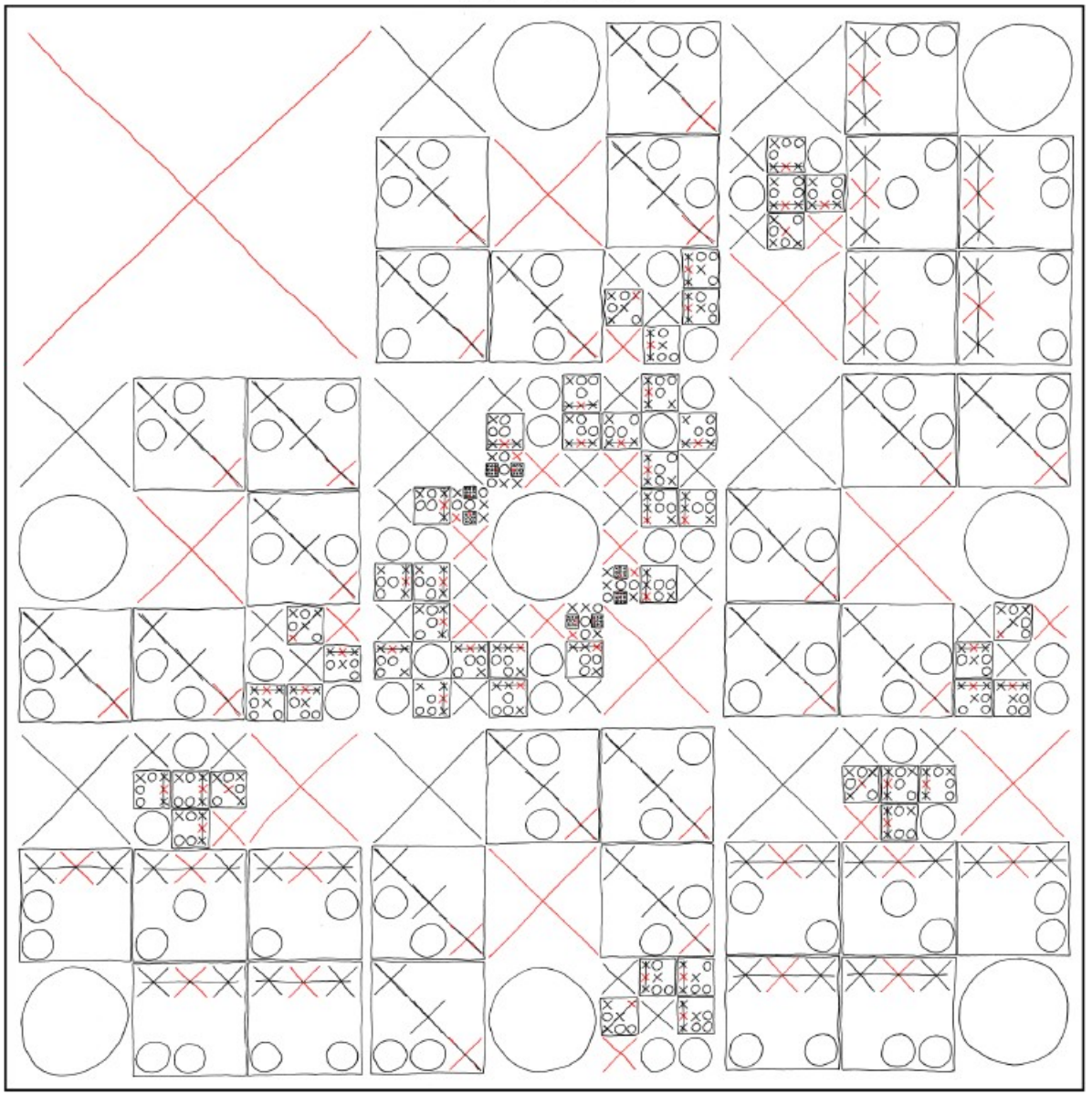




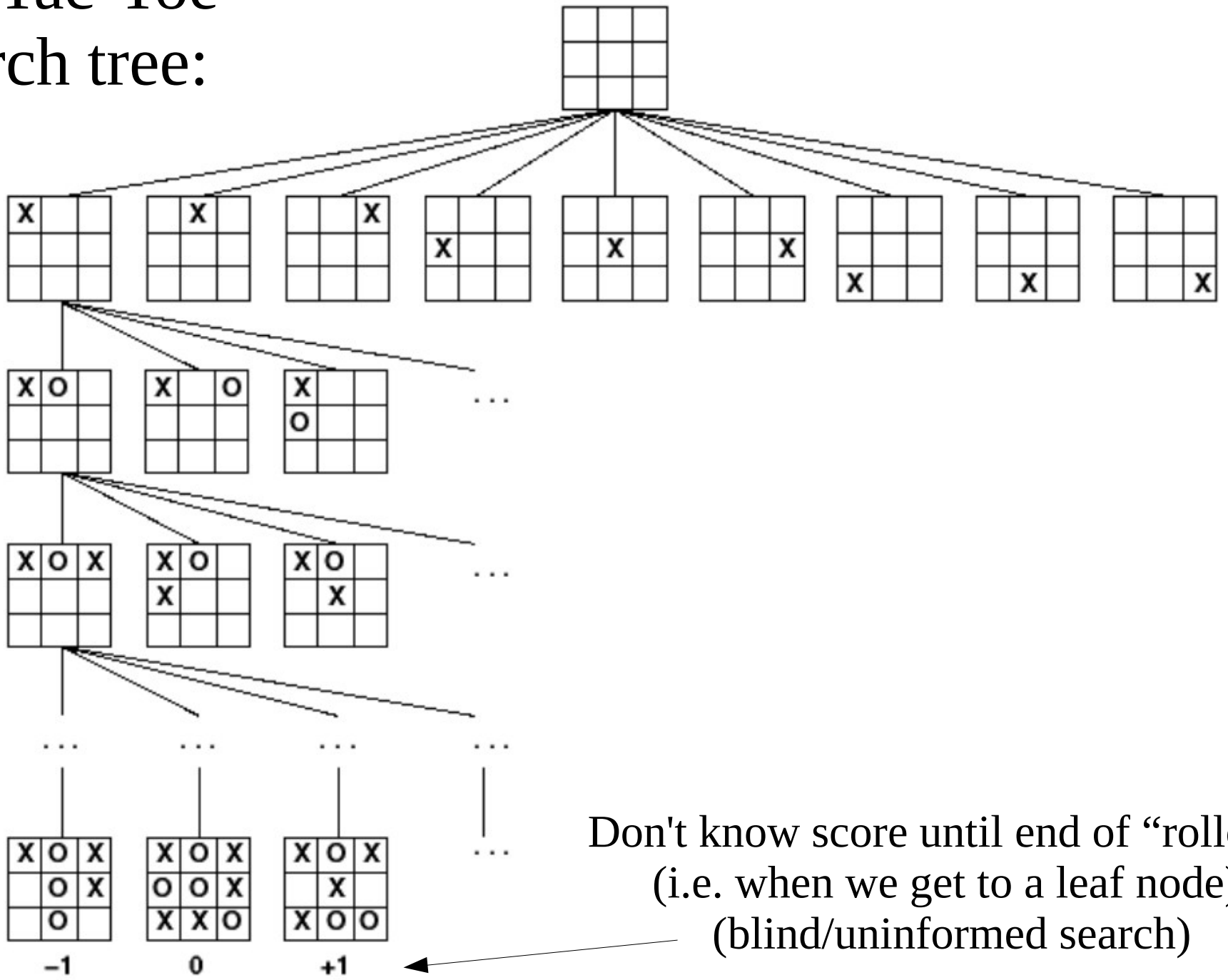
COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.

MAP FOR X:

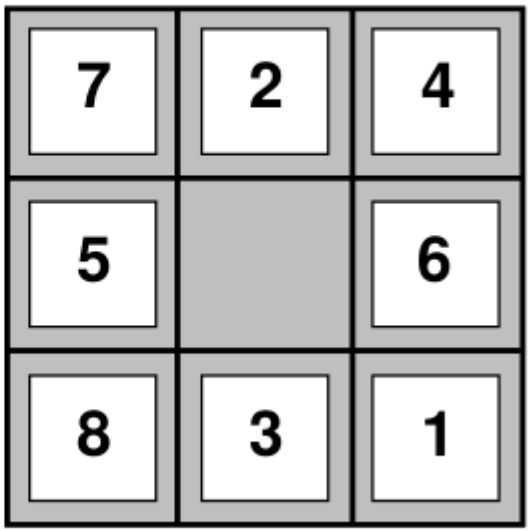


Tic-Tac-Toe search tree:

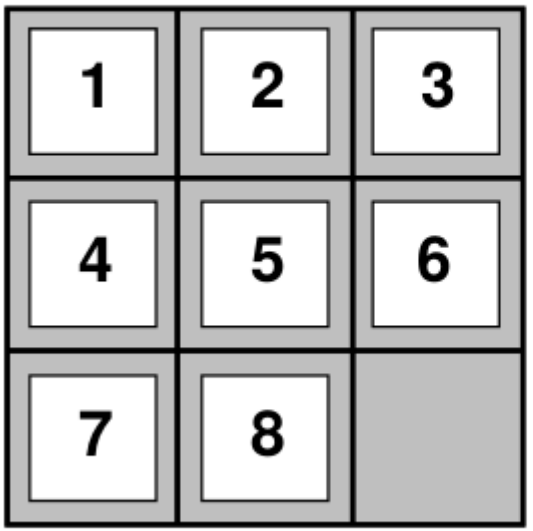


Don't know score until end of “rollout”
(i.e. when we get to a leaf node)
(blind/uninformed search)

Example: The 8-puzzle



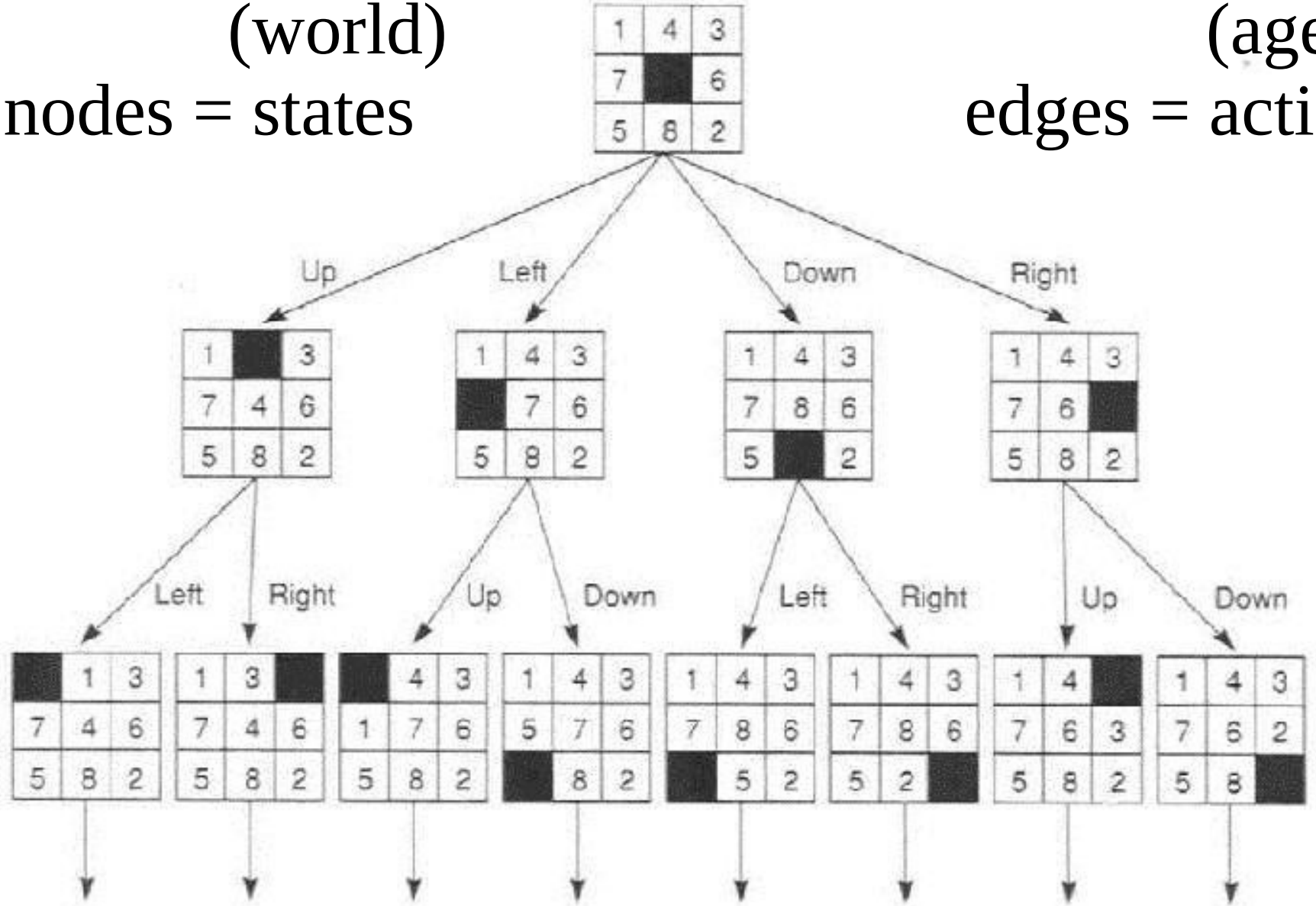
Start State



Goal State

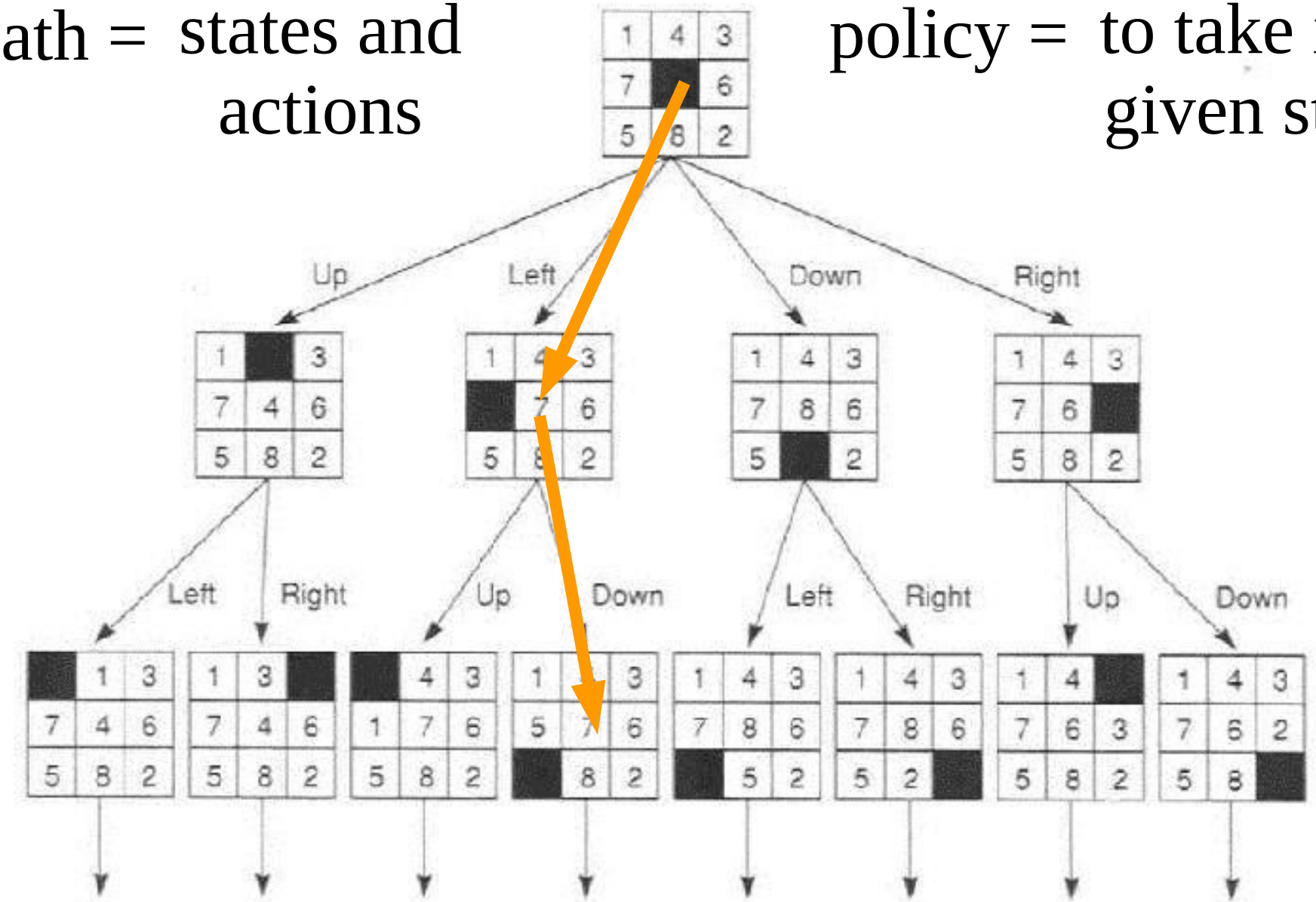
(world)
nodes = states

(agent)
edges = actions

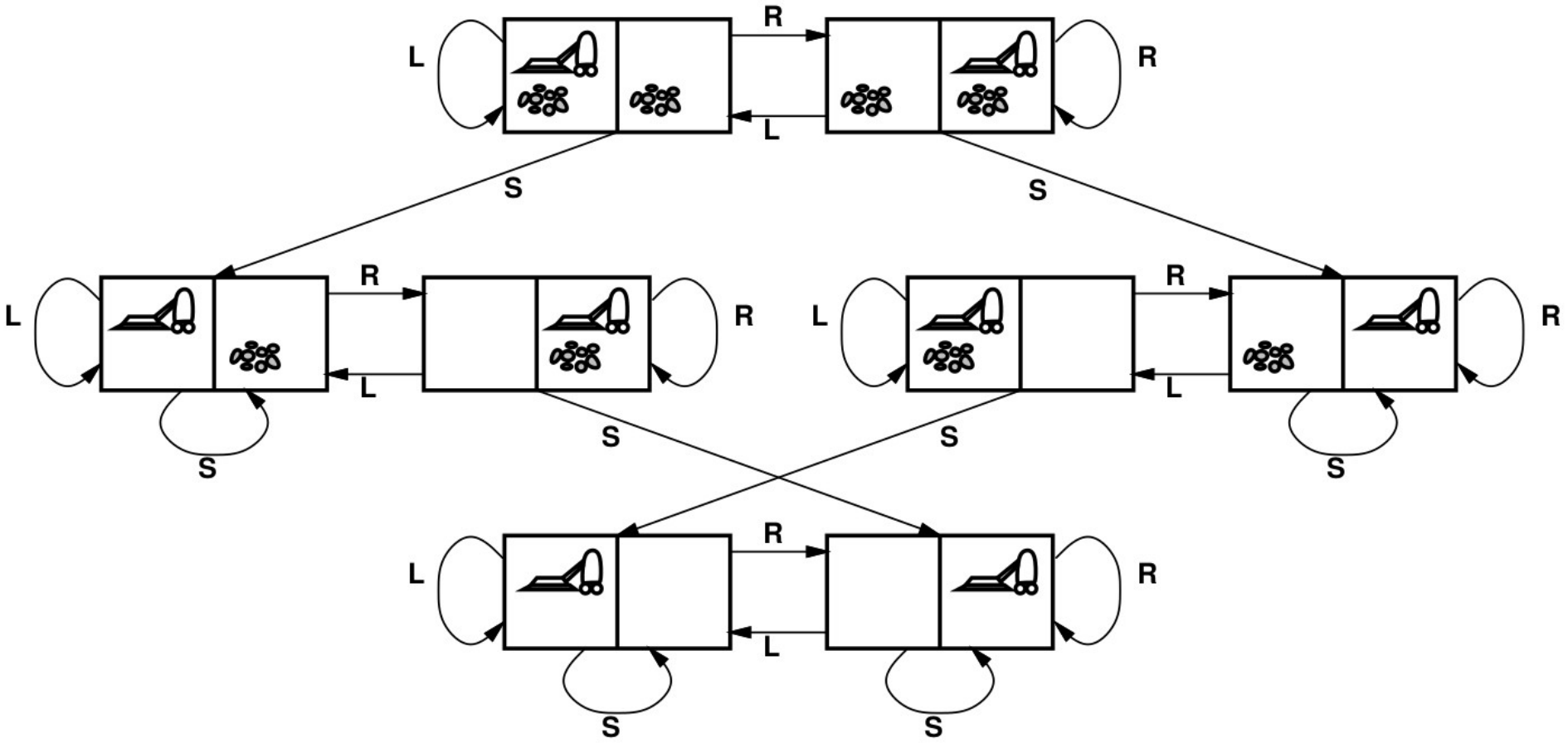


path = sequence of states and actions

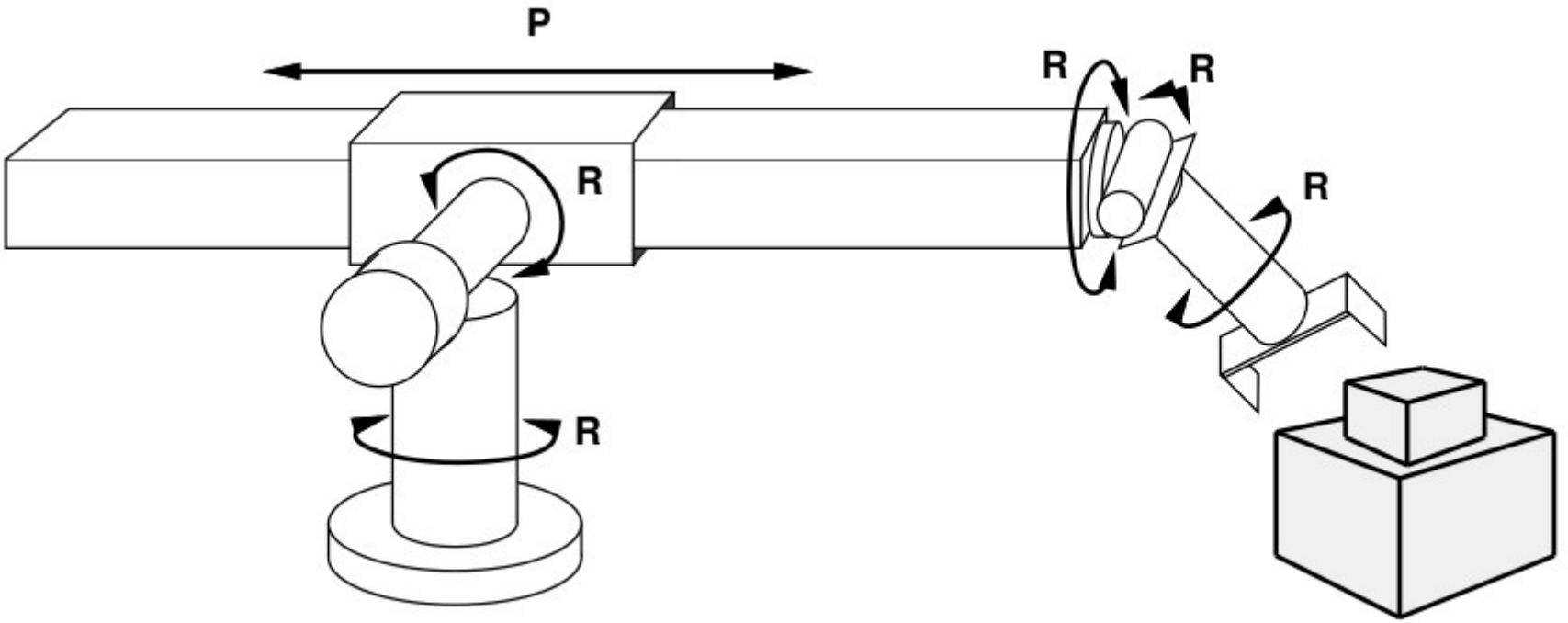
policy = which action to take in a given state



Example: vacuum world state space graph



Example: robotic assembly



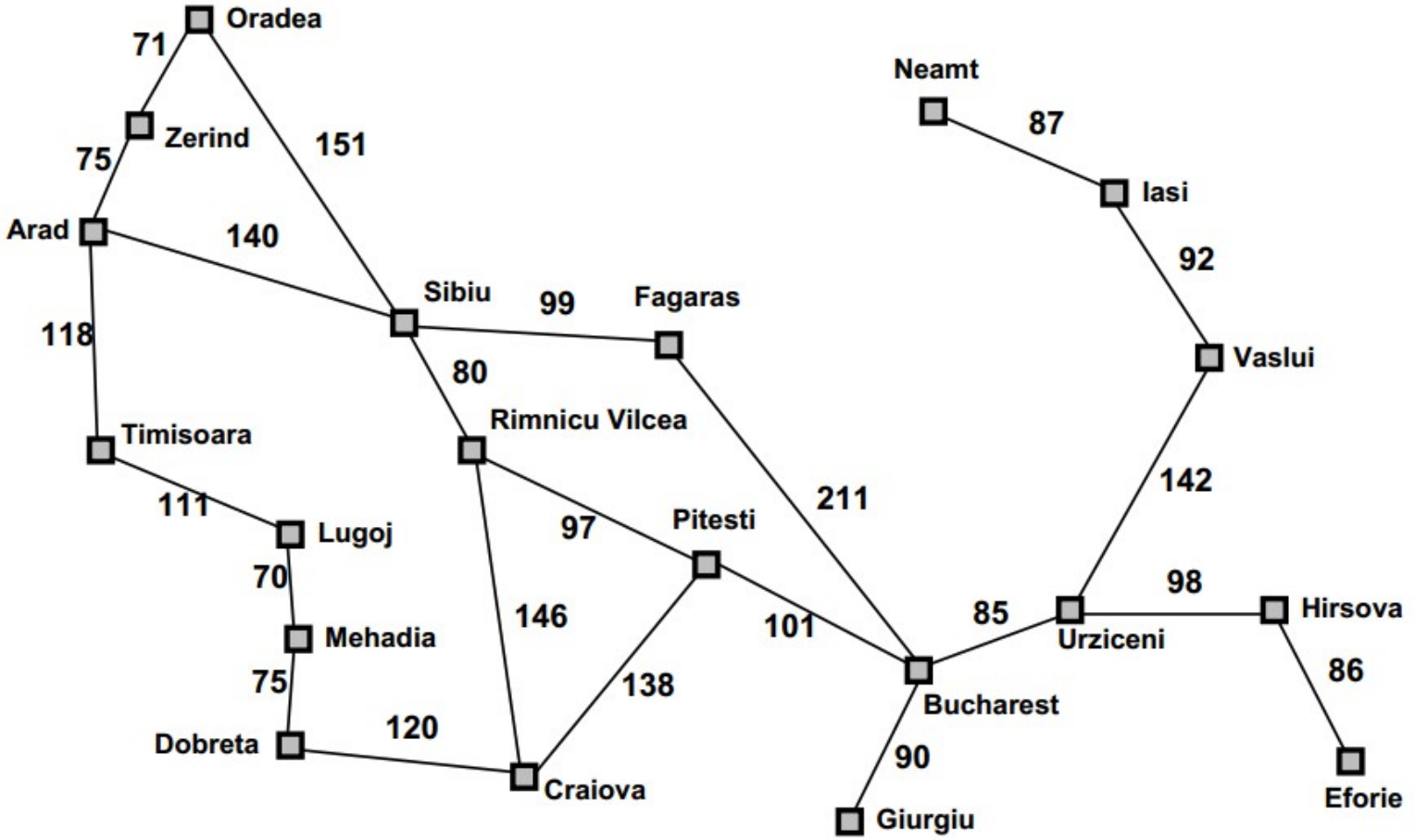
efficient search

we could always do exhaustive search
and be guaranteed to find out goal

the challenge is finding the goal
as efficiently as possible
(i.e. looking the fewest places)

(and we may need to consider the costs
associated with traveling to that state)

Romania with step costs in km



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

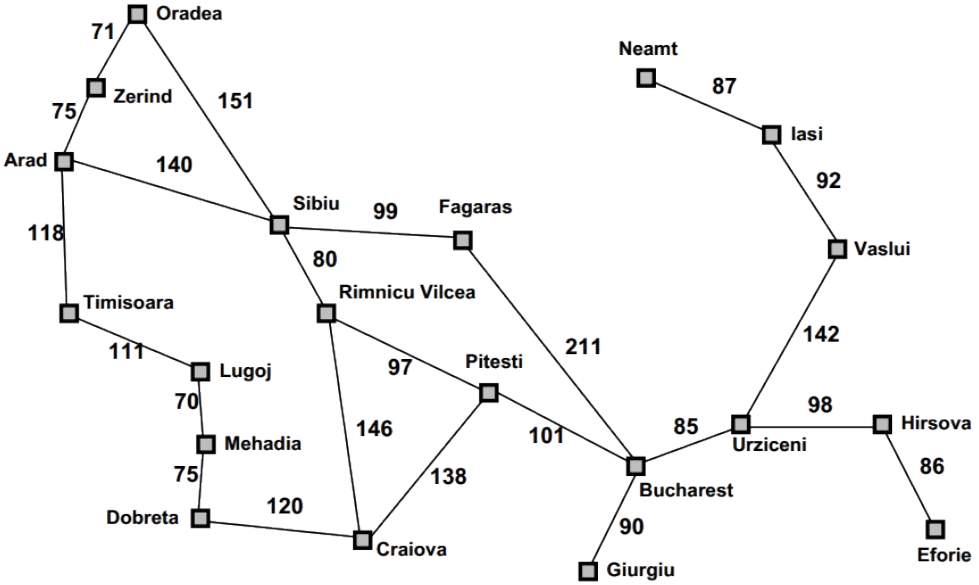
start state: you're in Arad

goal state: you want to get to Bucharest

Find a (any) path from Arad to Bucharest

1) frontier: Arad
2) frontier: Timisoara, Zerind, Sibiu
3) frontier: Zerind, Sibiu, Lugoj, Arad
4) frontier: Sibiu, Lugoj, Arad, Oradea, Arad

Hmmm… what's wrong?



function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do

if the frontier is empty **then return** failure

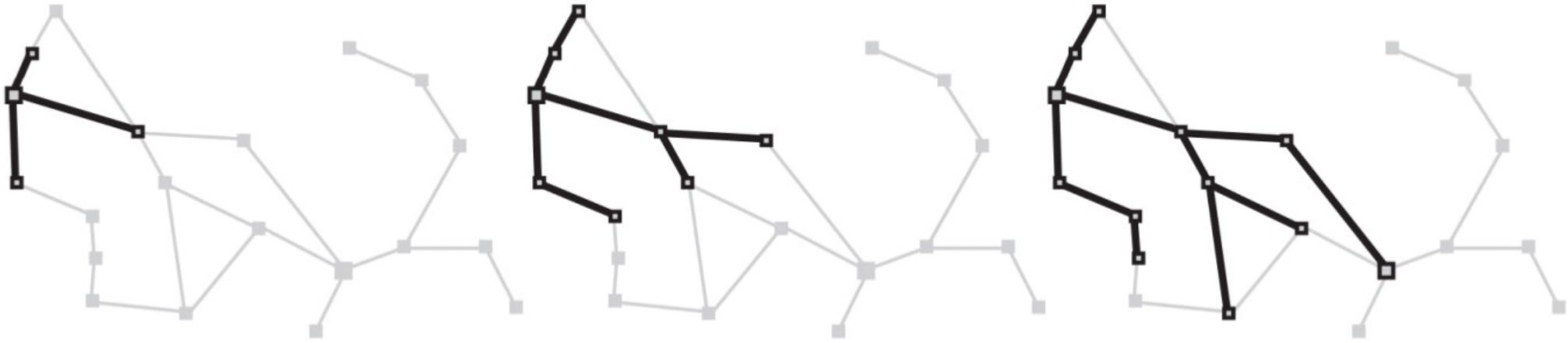
choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the corresponding solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to the frontier

only if not in the frontier or explored set



function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do

if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the corresponding solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to the frontier

only if not in the frontier or explored set

"search strategy"

we choose FIFO before (but why???)

all search algorithms are basically the same (look 'til you've found it!)

… but they differ in search strategies (where should I look next)

So really… what search strategy is best?

We evaluate them on:

1) Completeness *(does it solve the problem?)*

2) Optimality *(is the solution the lowest cost?)*

3) Time Complexity *(how many places did you look?)*

4) Space Complexity *(how much info did you have to store?)*

evaluation metrics for graphs/trees:

Time Complexity: # of nodes visited during search

Space Complexity: # of nodes stored in memory

(these are both worst case measures… usually)

for graphs/trees these are measured via:

branching factor (b)
(max # of successors/node)

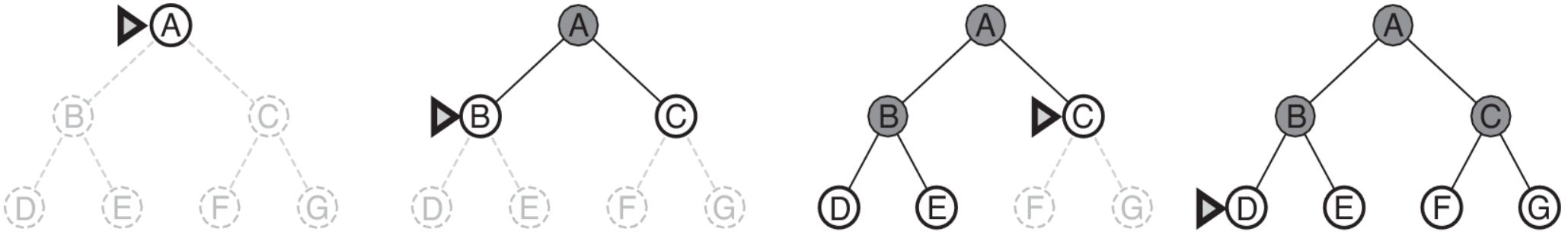
depth (d)
(shallowest goal node)

max path length (m)
(from start state to a terminal state)

search strategies… finally!

breadth-first search

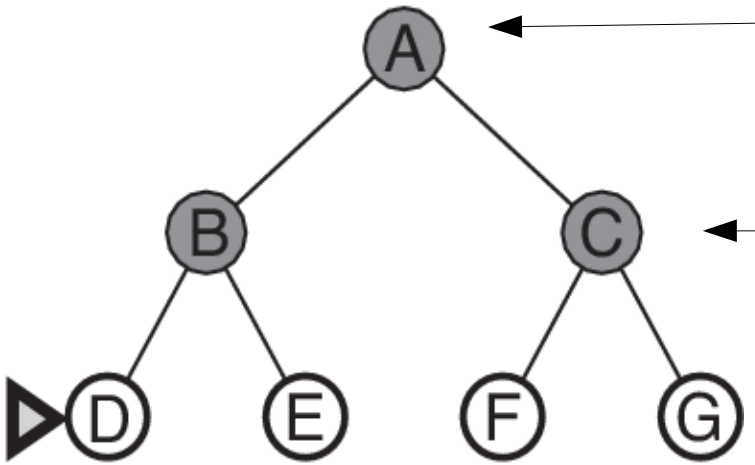
(FIFO)



frontier: A
frontier: B, C
frontier: C, D, E
frontier: D, E, F, G

b = 2

nodes visited:



$1 = 2^0$

$2 = 2^1$

$4 = 2^2$

$8 = 2^3$

… …

$? = 2^d$

space complexity generally:

b^d

time complexity generally:

b^d