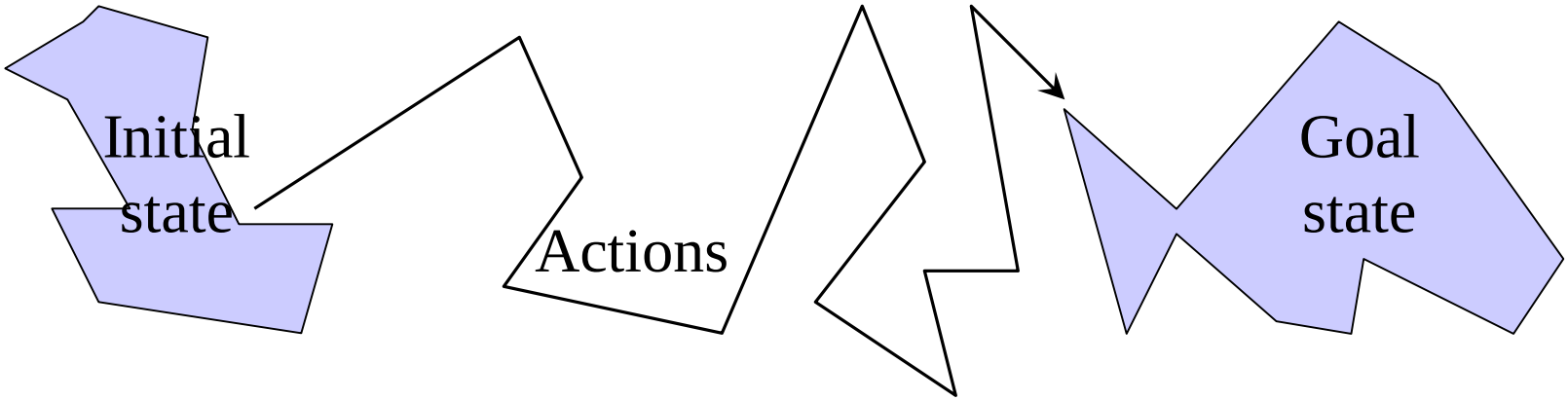
Class #3

Problem Solving as Search

Building goal-based agents

To build a goal-based agent we need to answer the following questions:

- What is the goal to be achieved?
- What are the actions?
- What *relevant* information is necessary to encode in order to describe the state of the world, describe the available transitions, and solve the problem?



What is the goal to be achieved?

- Could describe a situation we want to achieve, a set of properties that we want to hold, etc.
- Requires defining a **"goal test"** so that we know what it means to have achieved/satisfied our goal.

What are the actions?

- Characterize the **primitive actions** or events that are available for making changes in the world in order to achieve a goal.
- **Deterministic** world: no uncertainty in an action's effects. Given an action (a.k.a. **operator** or move) and a description of the **current world state**, the action completely specifies
  - whether that action *can* be applied to the current world (i.e., is it applicable and legal), and
  - what the exact state of the world will be after the action is performed in the current world (i.e., no need for "history" information to compute what the new world looks like).

Representing actions

- Note also that actions in this framework can all be considered as **discrete events** that occur at an **instant of time**.
  - For example, if "Mary is in class" and then performs the action "go home," then in the next situation she is "at home." There is no representation of a point in time where she is neither in class nor at home (i.e., in the state of "going home").
- The number of actions / operators depends on the **representation** used in describing a state.
  - In the 8-puzzle, we could specify 4 possible moves for each of the 8 tiles, resulting in a total of **4*8=32 operators**.
  - On the other hand, we could specify four moves for the "blank" square and we would only need **4 operators**.
- Representational shift can greatly simplify a problem!

Representing states

- What information is necessary to encode about the world to sufficiently describe all relevant aspects to solving the goal? That is, what knowledge needs to be represented in a state description to adequately describe the current state or situation of the world?
- The **size of a problem** is usually described in terms of the **number of states** that are possible.
  - Tic-Tac-Toe has about 3^9 states.
  - Checkers has about 10^{40} states.
  - Rubik's Cube has about 10^{19} states.
  - Chess has about 10^{120} states in a typical game.

Closed World Assumption

- We will generally use the **Closed World Assumption**.
- All necessary information about a problem domain is available in each percept so that each state is a complete description of the world.
- There is no incomplete information at any point in time.

Some example problems

- Toy problems and micro-worlds
  - 8-Puzzle
  - Missionaries and Cannibals
  - Cryptarithmetic
  - Remove 5 Sticks
  - Water Jug Problem
- Real-world problems

Cryptarithmetic Problem

A. Digit ranges from 0 to 9 only.

B. Each Variable should have unique and distinct value.

C. Each Letter, Symbol represents only one digit throughout the problem.

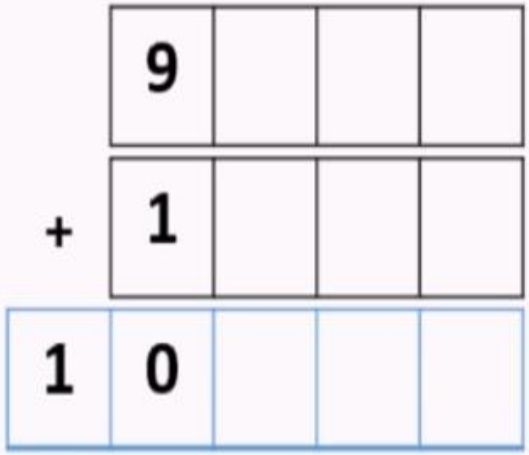
D. You have to find the value of each letter in the Crypt arithmetic.

E. The Numerical base, unless specifically stated , is 10.

F. Numbers must not begin with zero i.e. 0123 (wrong) , 123 (correct).

G. After replacing letters by their digits, the resulting arithmetic operations must be correct.

Cryptarithmetic Problem



	C4	C3	C2	C1	
		S	E	N	D
+		M	O	R	E
	M	O	N	E	Y

C3+S+M = 10+O

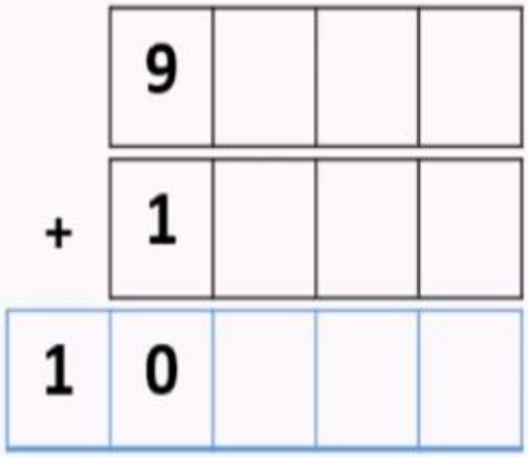
0+S+1 = 10+O

S = 9+O

C3 → 0, 1

Character	Code
S	9
E	
N	
D	
M	1
O	0
R	
Y	

Cryptarithmetic Problem



1+E = N

C4 C3 C2 C1

S E N D

\+ M O R E

M O N E Y

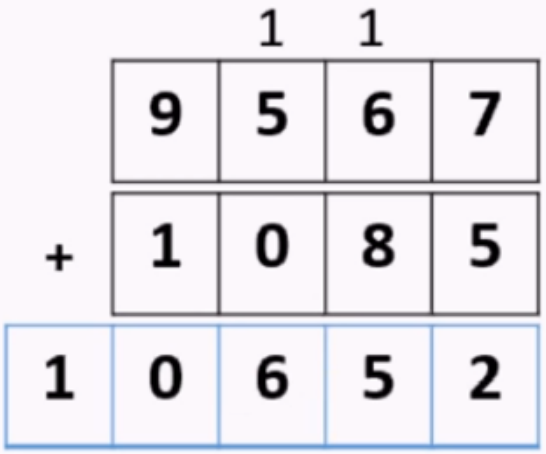
C1+N+R = 10+E

C1+1+E+R = 10+E

C1+R=9

Character	Code
S	9
E	
N	
D	
M	1
O	0
R	
Y	

Cryptarithmetic Problem



1+E = N

C4 C3 C2 C1

S E N D

\+ M O R E

M O N E Y

D+E = 10+Y

If Y=2

D= 5+2

D=7

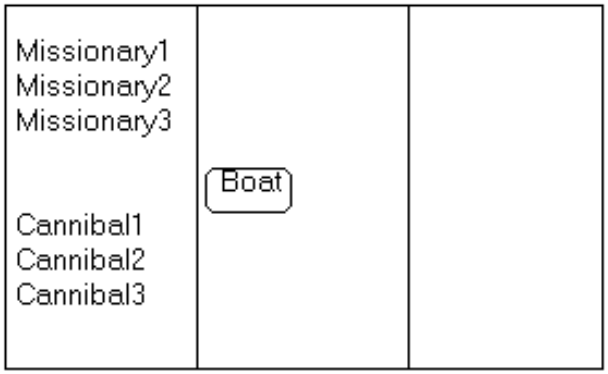
Character	Code
S	9
E	5
N	6
D	7
M	1
O	0
R	8
Y	2

7 6 5 4 3 2

Missionaries and Cannibals

There are 3 missionaries, 3 cannibals, and 1 boat that can carry up to two people on one side of a river.

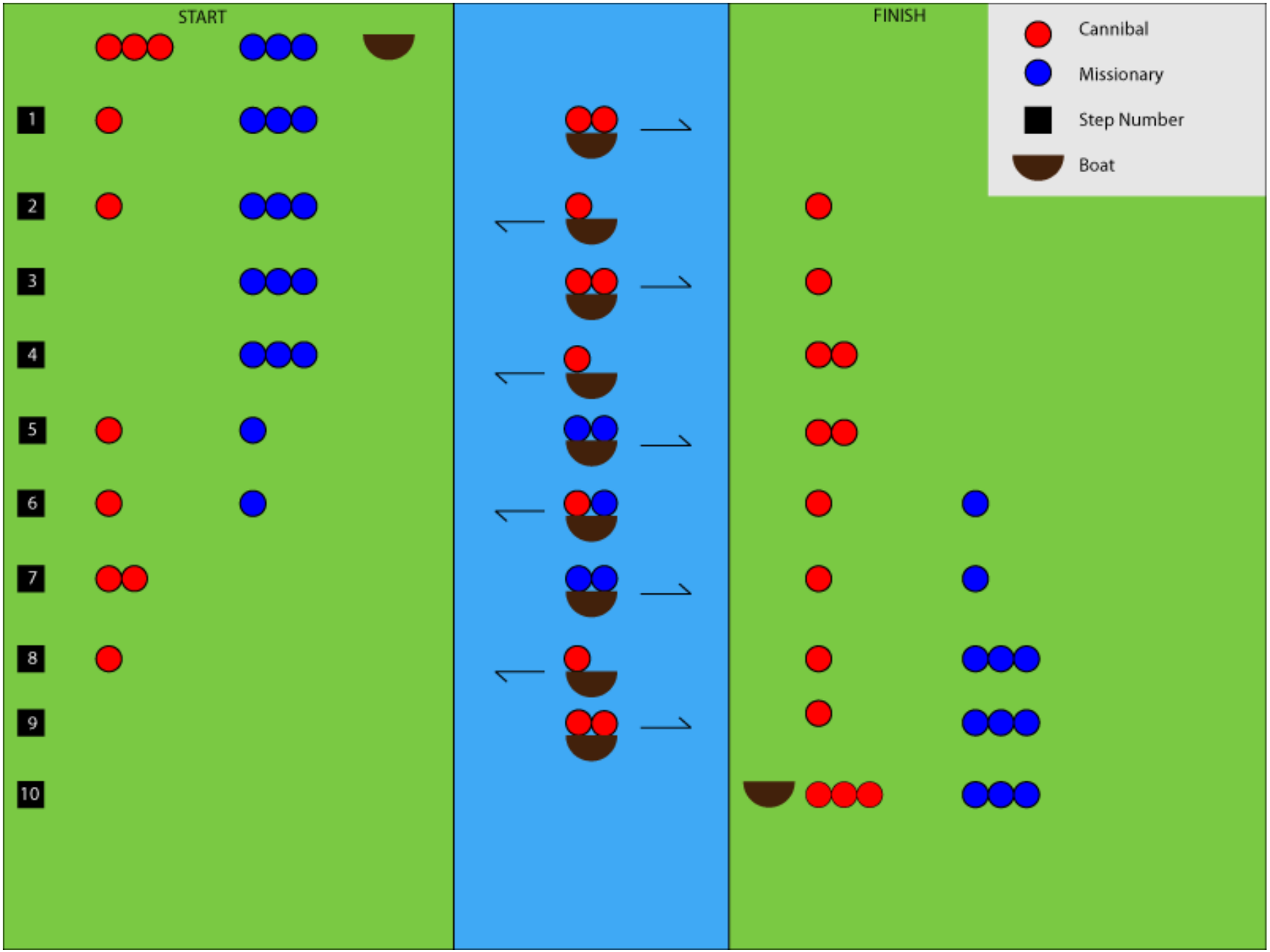
- **Goal**: Move all the missionaries and cannibals across the river.
- **Constraint:** Missionaries can never be outnumbered by cannibals on either side of river, or else the missionaries are killed.
- **State:** configuration of missionaries and cannibals and boat on each side of river.
- **Operators:** Move boat containing some set of occupants across the river (in either direction) to the other side.



3 Missionaries and 3 Cannibals wish to cross the river. They have a boat that will carry two people. Everyone can navigatethe boat. If at any time the Cannibals outnumber the missionaries on either bank of the river, they will eat the Missionaries. Find the smallest number of crossings that will allow everyone to cross the river safely.
The problem can be solved in 11 moves. But people rarely get the optimal solution, because the MC problem contains a 'tricky' state at the end, where two people move back across the river.

Missionaries and Cannibals Solution

	Near side		***Far side***	
0 Initial setup:	MMMCCC	B		-
1 Two cannibals cross over:	MMMC		B	CC
2 One comes back:	MMMCC	B		C
3 Two cannibals go over again:	MMM		B	CCC
4 One comes back:	MMMC	B		CC
5 Two missionaries cross:	MC		B	MMCC
6 A missionary & cannibal return:	MMCC	B		MC
7 Two missionaries cross again:	CC		B	MMMC
8 A cannibal returns:	CCC	B		MMM
9 Two cannibals cross:	C		B	MMMCC
10 One returns:	CC	B		MMMC
11 And brings over the third:	-		B	MMMCCC


START
FINISH
Cannibal
Missionary
Step Number
Boat
1
2
3
4
5
6
7
8
9
10

Some more real-world problems

- Route finding
- Touring (traveling salesman)
- Logistics
- VLSI layout
- Robot navigation
- Learning

Knowledge representation issues

- What's in a state ?
  - Is the color of the boat relevant to solving the Missionaries and Cannibals problem? Is sunspot activity relevant to predicting the stock market? What to represent is a very hard problem that is usually left to the system designer to specify.
- What **level of abstraction** or detail to describe the world.
  - Too fine-grained and we'll "miss the forest for the trees." Too coarse-grained and we'll miss critical details for solving the problem.
- The number of states depends on the representation and level of abstraction chosen.
  - In the Remove-5-Sticks problem, if we represent the individual sticks, then there are 17-choose-5 possible ways of removing 5 sticks. On the other hand, if we represent the "squares" defined by 4 sticks, then there are 6 squares initially and we must remove 3 squares, so only 6-choose-3 ways of removing 3 squares.

Formalizing search in a state space

- A state space is a **graph** (V, E) where V is a set of **nodes** and E is a set of **arcs**, and each arc is directed from a node to another node
- **Each node is a data structure** that contains **a state description** plus other information such as the parent of the node, the name of the operator that generated the node from that parent, and other bookkeeping data
- Each **arc** corresponds to an **instance of one of the operators**. When the operator is applied to the state associated with the arc's source node, then the resulting state is the state associated with the arc's destination node.

Formalizing search II

- Each arc has a fixed, positive **cost** associated with it corresponding to the cost of the operator.
- Each node has a set of **successor nodes** corresponding to all of the legal operators that can be applied at the source node's state.
  - The process of expanding a node means to generate all of the successor nodes and add them and their associated arcs to the state-space graph
- One or more nodes are designated as **start nodes.**
- A **goal test** predicate is applied to a state to determine if its associated node is a goal node.

Water Jug Problem

Given a full 5-gallon jug and an empty 2-gallon jug, the goal is to fill the 2-gallon jug with exactly one gallon of water.

- State = (x,y), where x is the number of gallons of water in the 5-gallon jug and y is # of gallons in the 2-gallon jug
- Initial State = (5,0)
- Goal State = (*,1), where * means any amount

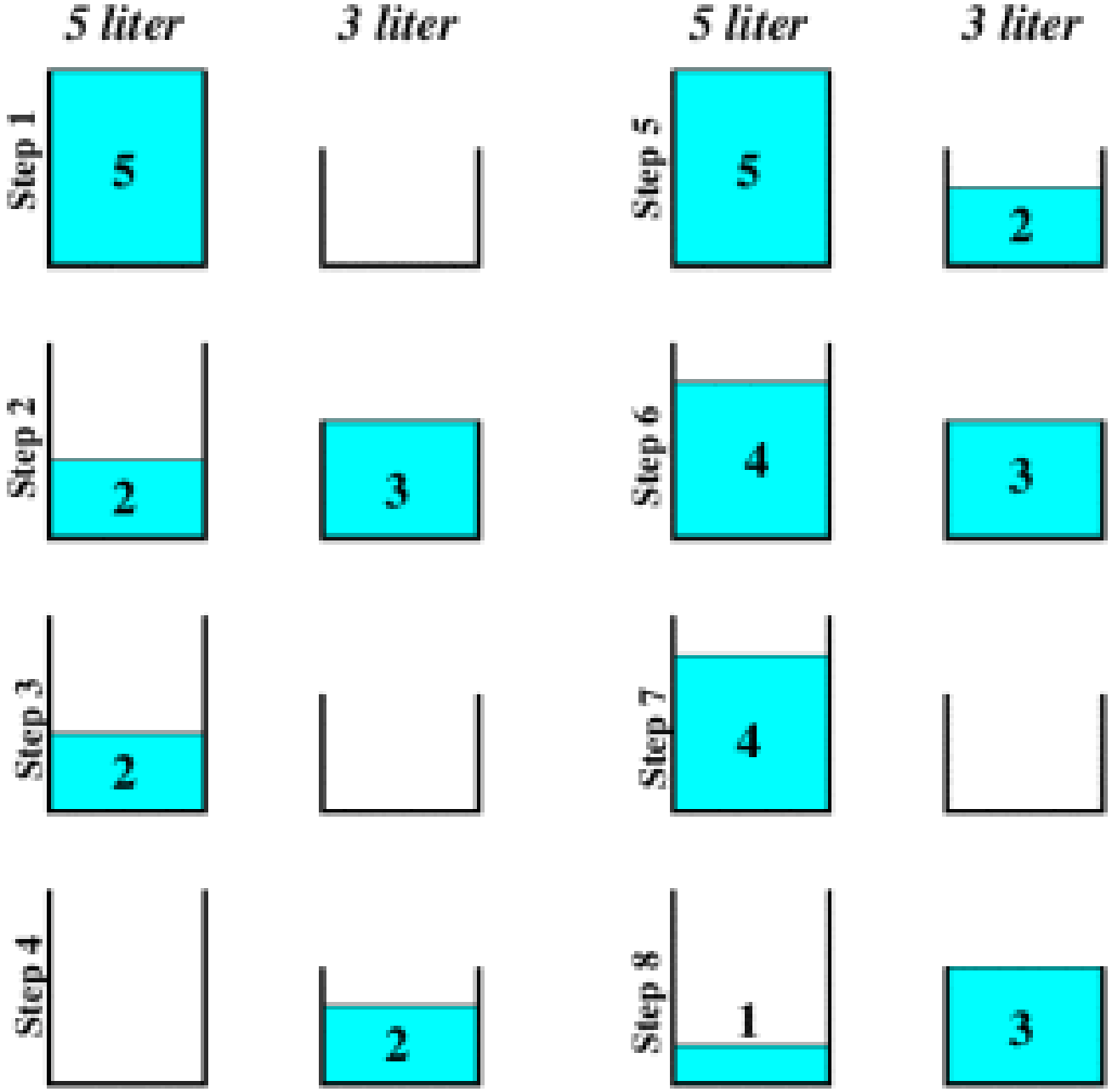
Operator table

Name	Cond.	Transition	Effect
Empty5	–	(x,y)→(0,y)	Empty 5-gal. jug
Empty2	–	(x,y)→(x,0)	Empty 2-gal. jug
2to5	$x \leq 3$	(x,2)→(x+2,0)	Pour 2-gal. into 5-gal.
5to2	$x \geq 2$	(x,0)→(x-2,2)	Pour 5-gal. into 2-gal.
5to2part	$y < 2$	(1,y)→(0,y+1)	Pour partial 5-gal. into 2-gal.

Water Jug Problem

- To solve this we have to make some assumptions not mentioned in the problem. They are
- 1. We can fill a jug from the pump.
- 2. we can pour water out of a jug to the ground.
- 3. We can pour water from one jug to another.
- 4. There is no measuring device available.

0-3
3-0
3-3
4-2
0-2
2-0


5 liter
3 liter
5 liter
3 liter
Step 1
5
Step 2
2
3
Step 3
2
Step 4
2
Step 5
5
2
Step 6
4
3
Step 7
4
Step 8
1
3

1.	(X, Y)	if	$X < 4$	$\rightarrow (4, Y)$	Fill the 4-gallon jug
2.	(X, Y)	if	$Y < 3$	$\rightarrow (X, 3)$	Fill the 3-gallon jug
3.	(X, Y)	if	$X = d \ \& \ d > 0$	$\rightarrow (X\text{-}d, Y)$	Pour some water out of the 4-gallon jug
4.	(X, Y)	if	$Y = d \ \& \ d > 0$	$\rightarrow (X, Y - d)$	Pour some water out of 3-gallon jug
5.	(X, Y)	if	$X > 0$	$\rightarrow (0, Y)$	Empty the 4-gallon jug on the ground
6.	(X, Y)	if	$Y > 0$	$\rightarrow (X, 0)$	Empty the 3-gallon jug on the ground
7.	(X, Y)	if	$X + Y \leq 4$ and		Pour water from the 3-gallon jug into the
	$Y > 0$	$\rightarrow$	$4, (Y - (4 - X))$		4-gallon jug until the gallon jug is full.
8.	(X, Y)	if	$X + Y \geq 3$ and		Pour water from the 4-gallon jug into the
	$X > 0$	$\rightarrow$	$(X - (3 - Y), 3))$		3-gallon jug until the 3-gallon jug is full.
9.	(X, Y)	if	$X + Y \leq 4$ and		Pour all the water from the 3-gallon jug
	$Y > 0$	$\rightarrow$	$(X + Y, 0)$		into the 4-gallon jug
10.	(X, Y)	if	$X + Y \leq 3$ and		Pour all the water from the 4-gallon jug
	$X > 0$	$\rightarrow$	$(0, X + Y)$		into the 3-gallon jug
11.	$(0, 2)$	$\rightarrow$	$(2, 0)$		Pour the 2-gallons water from 3-gallon jug into the 4;gallon jug
12.	$(2, Y)$	$\rightarrow$	$(0, Y)$		Empty the 2-gallons in the 4-gallon jug on the ground.

Fig. 2.3. *Production rules (operators) for the water jug problem.*

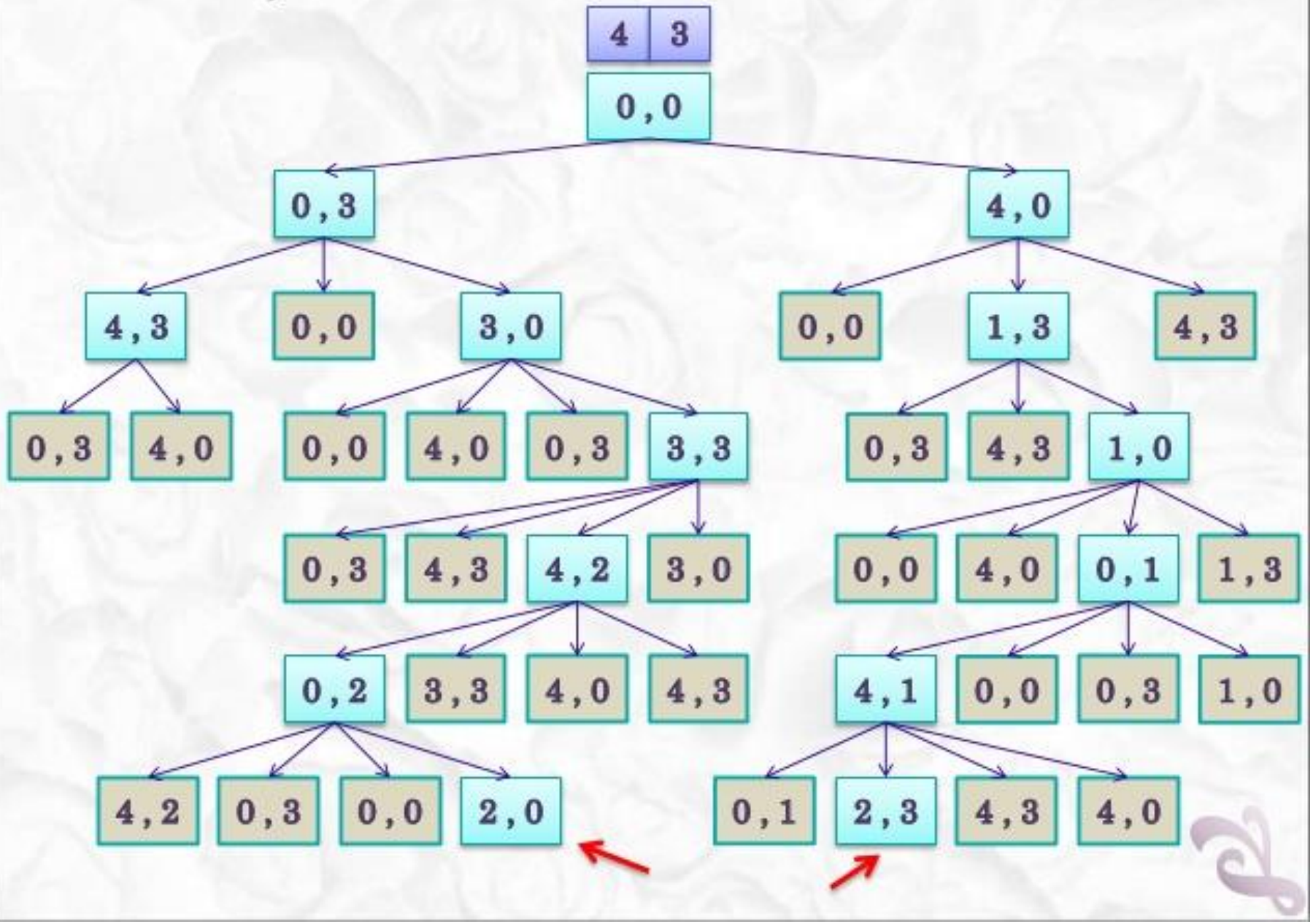
Water in 4-gallon jug (X)	Water in 3-gallon jug (Y)	Rule applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

Fig. 2.4 (*a*). *A solution to water jug problem.*

X	Y	Rule applied (Control strategy)
0	0	
4	0	I-
1	3	8
1	0	6
0	1	10
4	1	1
2	3	8

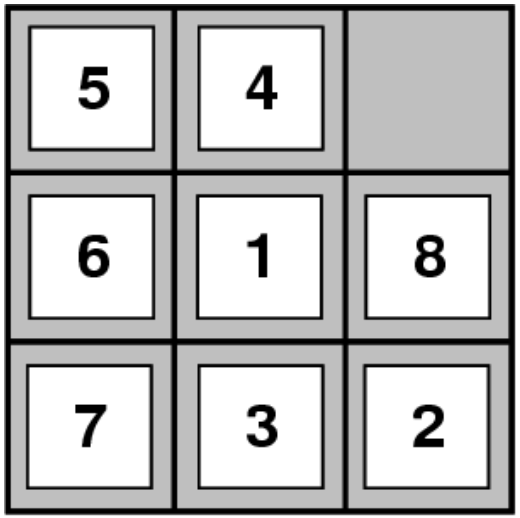
Fig. 2.4 (*b*). *2nd solution to water jug problem.*

THE WATER JUGS PROBLEM – SEARCH TREE

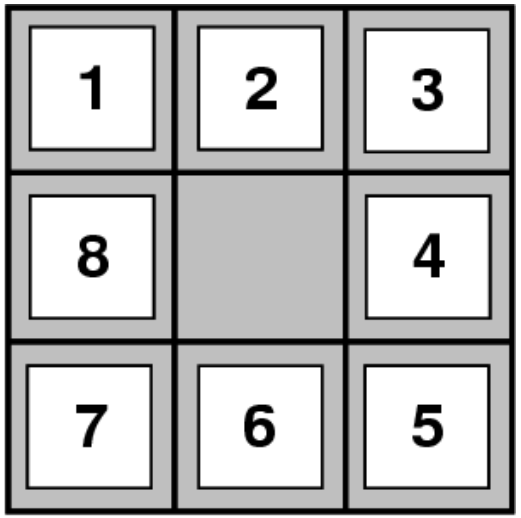


8-Puzzle

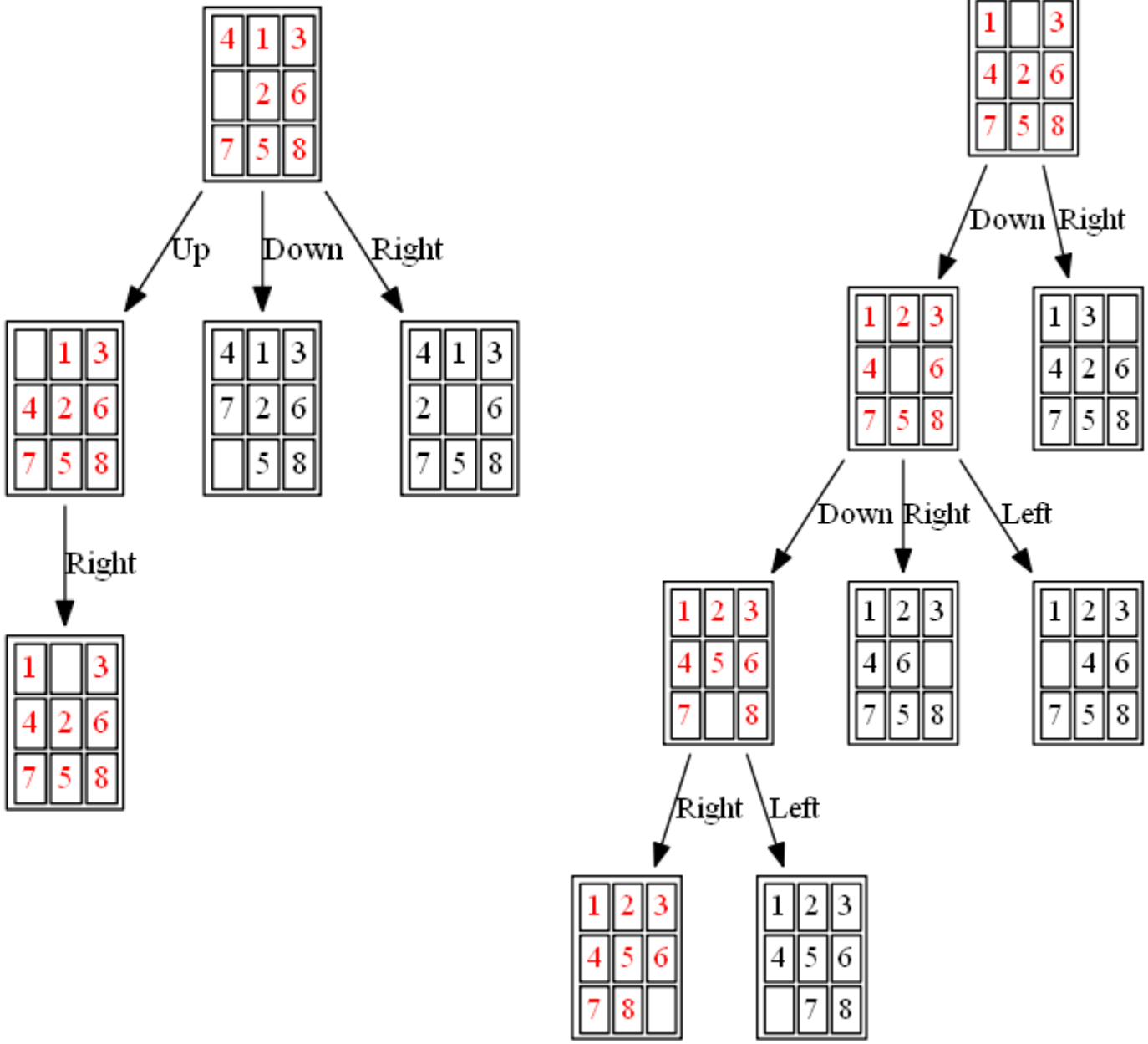
Given an initial configuration of 8 numbered tiles on a 3 x 3 board, move the tiles in such a way so as to produce a desired goal configuration of the tiles.

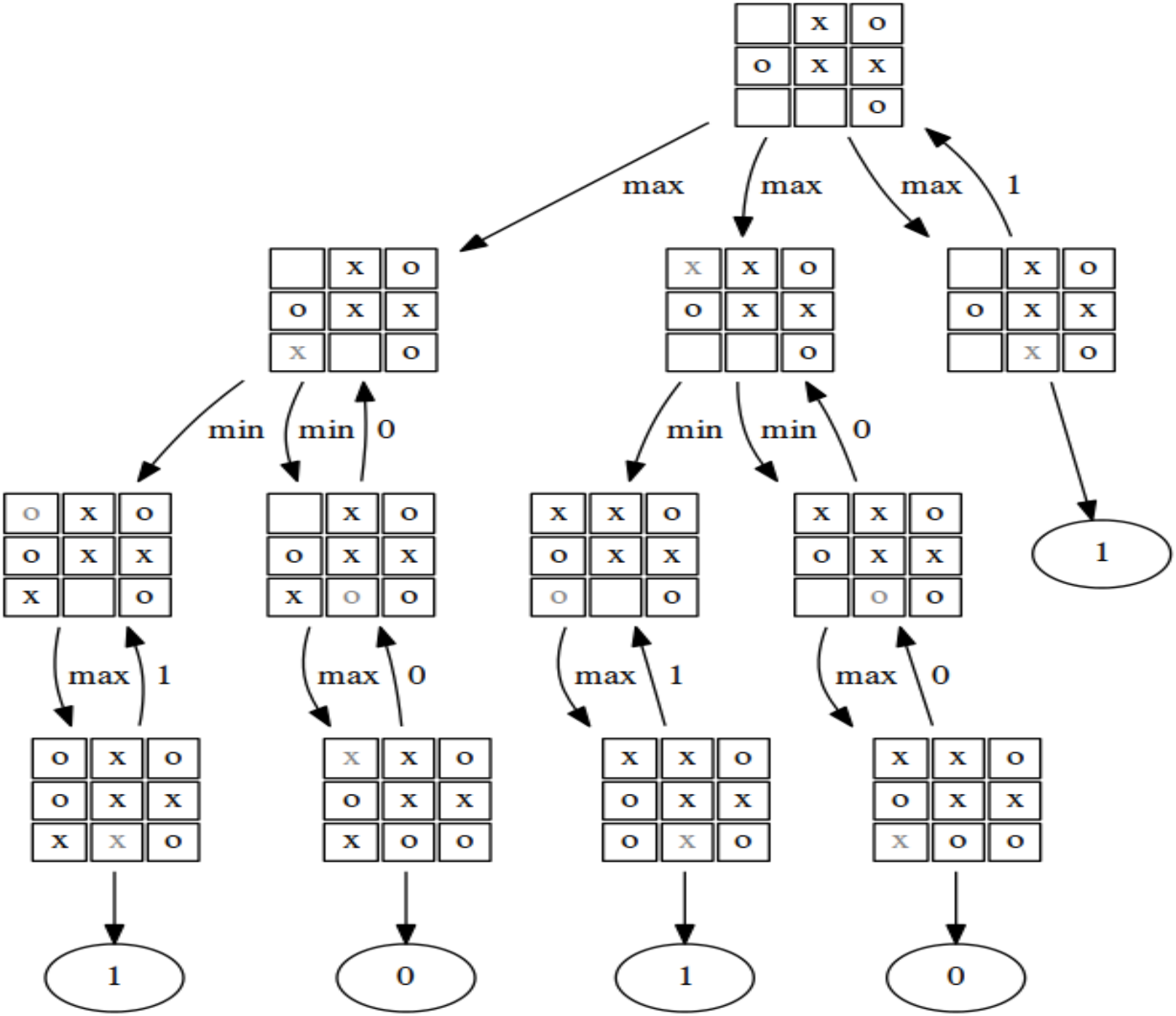


Start State



Goal State


Up
Down
Right
Right
Down
Right
Down
Right
Left
Right
Left


max
max
max
1
min
min
0
min
min
0
max
1
max
0
max
1
max
0
1
0
1
0

Formalizing search III

- A **solution** is a sequence of operators that is associated with a path in a state space from a start node to a goal node.
- The **cost of a solution** is the sum of the arc costs on the solution path.
  - If all arcs have the same (unit) cost, then the solution cost is just the length of the solution (number of steps / state transitions)

Key procedures to be defined in State Space Search

- EXPAND
  - Generate all successor nodes of a given node
- GOAL-TEST
  - Test if state satisfies all goal conditions
- QUEUEING-FUNCTION
  - Used to maintain a ranked list of nodes that are candidates for expansion

Some issues

- Search process constructs a search tree, where
  - **root** is the initial state and
  - **leaf nodes** are nodes
    - not yet expanded (i.e., they are in the list "nodes") or
    - having no successors (i.e., they're "deadends" because no operators were applicable and yet they are not goals)
- Search tree may be infinite because of loops even if state space is small
- Return a path or a node depending on problem.
  - E.g., in cryptarithmetic return a node; in 8-puzzle return a path
- Changing definition of the QUEUEING-FUNCTION leads to different search strategies

Evaluating search strategies

- **Completeness**
  - Guarantees finding a solution whenever one exists
- **Time complexity**
  - How long (worst or average case) does it take to find a solution? Usually measured in terms of the number of nodes expanded
- **Space complexity**
  - How much space is used by the algorithm? Usually measured in terms of the maximum size of the “nodes” list during the search
- **Optimality/Admissibility**
  - If a solution is found, is it guaranteed to be an optimal one? That is, is it the one with minimum cost?