

# Exercises: Artificial Intelligence

The farmer, fox, goose and grain

# Representation

- States of the form  $[\mathcal{L}|\mathcal{R}]$ , where:
  - $\mathcal{L}$ : *Items on left bank*
  - $\mathcal{R}$ : *Items on right bank*
- $\mathcal{L}$  and  $\mathcal{R}$  contain:
  - Fa: *Farmer*
  - Fo: *Fox*
  - Go: *Goose*
  - Gr: *Grain*

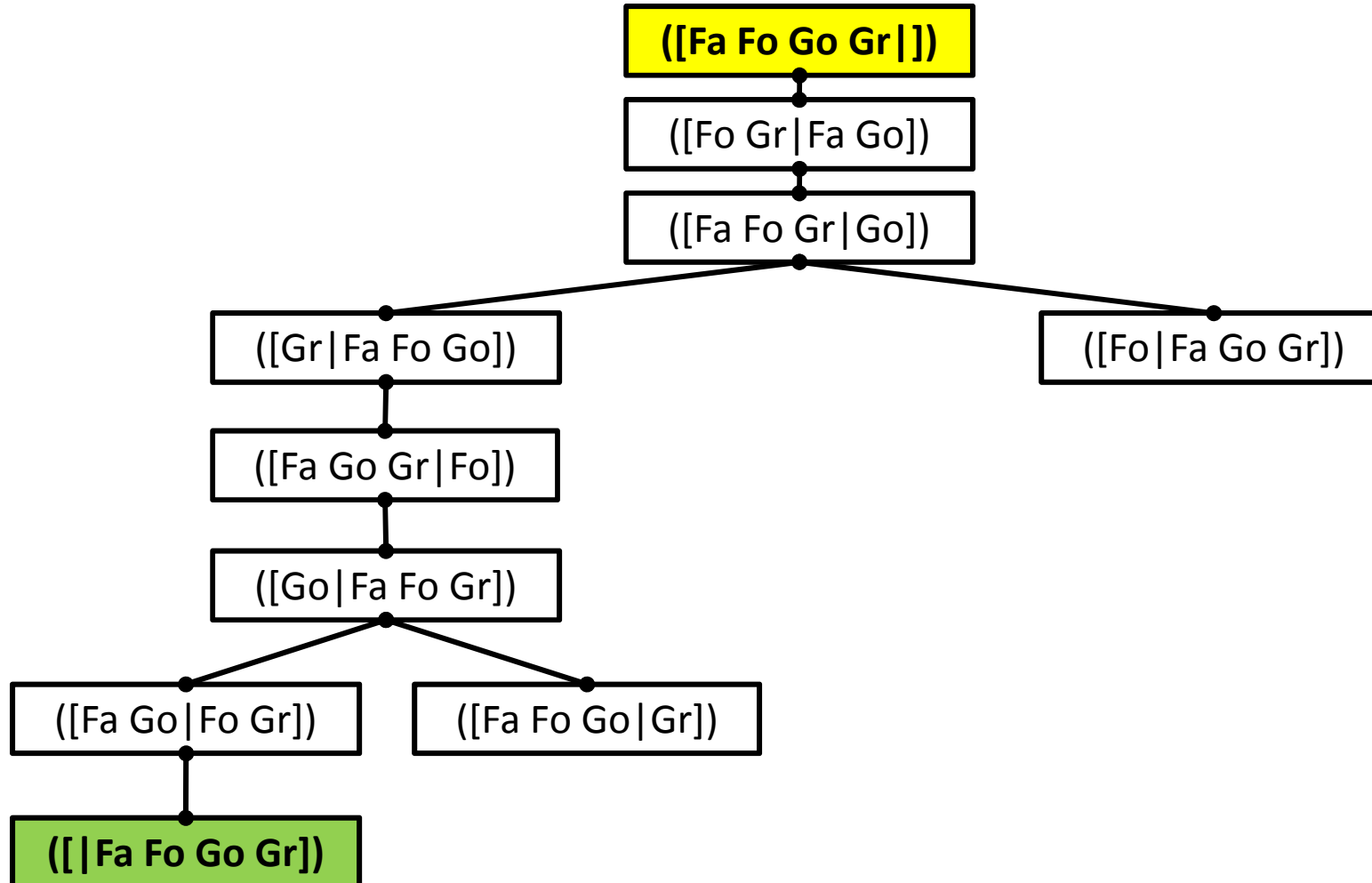
# Representation

- Start: [Fa Fo Go Gr |]
- Goal: [| Fa Fo Go Gr]
- Rules:
  - $R_1: [Fa \mathcal{X} | \mathcal{Y}] \longrightarrow [\mathcal{X} | Fa \mathcal{Y}]$
  - $R_2: [\mathcal{X} | Fa \mathcal{Y}] \longrightarrow [Fa \mathcal{X} | \mathcal{Y}]$
  - $R_3: [Fa z \mathcal{X} | \mathcal{Y}] \longrightarrow [\mathcal{X} | Fa z \mathcal{Y}]$
  - $R_4: [\mathcal{X} | Fa z \mathcal{Y}] \longrightarrow [Fa z \mathcal{X} | \mathcal{Y}]$
  - No combination (Fo,Go) or (Go,Gr) on either bank, without the farmer.

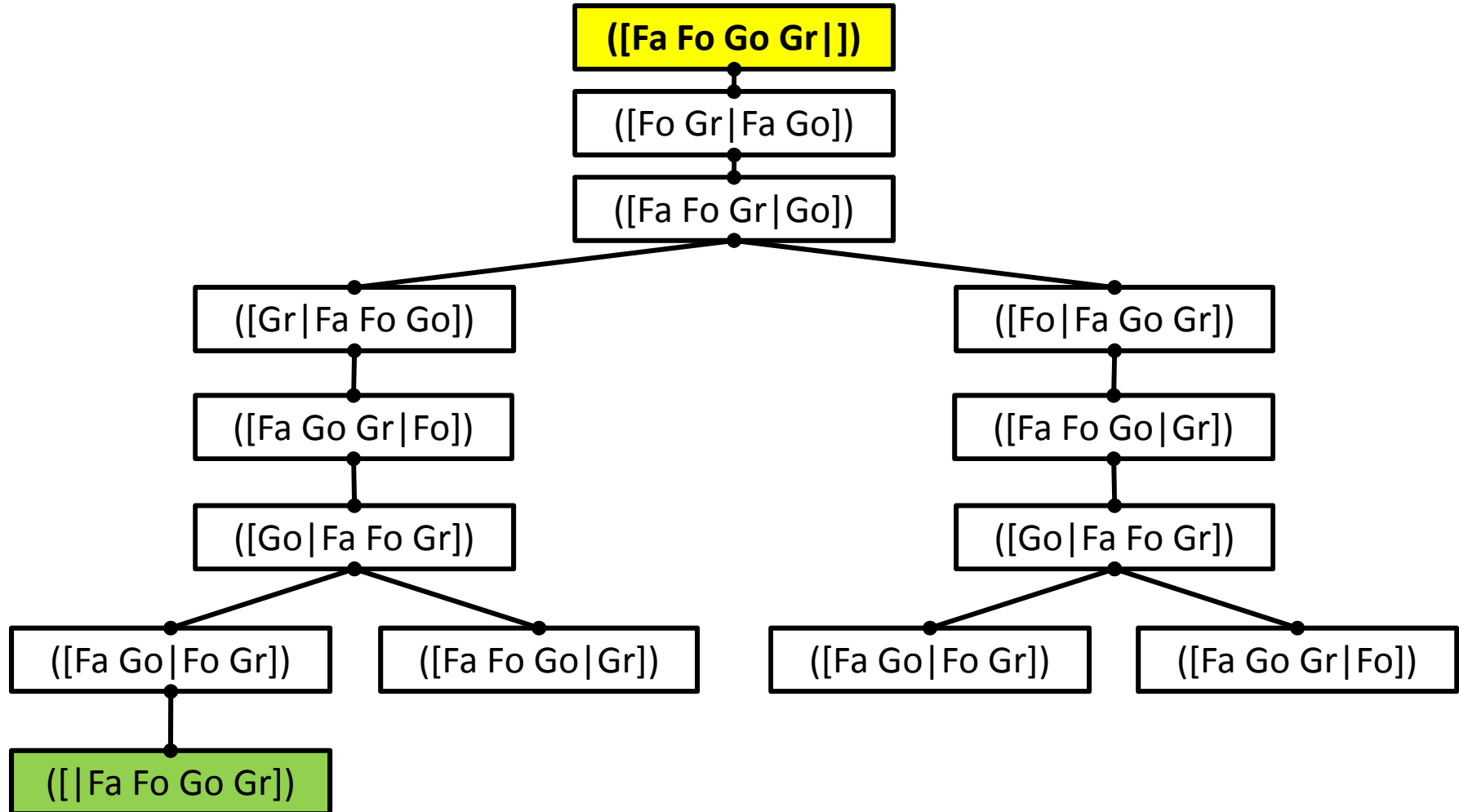
# Depth-first search (queues)

- $S = (\langle [Fa\ Fo\ Go\ Gr\ |] \rangle)$
- $Q_1 = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go] \rangle)$
- $Q_2 = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go] \rangle)$
- $Q_3 = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Fo\ |Fa\ Go\ Gr] \rangle)$
- $Q_4 = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go][Fa\ Go\ Gr\ |Fo] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Fo\ |Fa\ Go\ Gr] \rangle)$
- $Q_5 = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go][Fa\ Go\ Gr\ |Fo][Go\ |Fa\ Fo\ Gr] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Fo\ |Fa\ Go\ Gr] \rangle)$
- $Q_6 = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go][Fa\ Go\ Gr\ |Fo][Go\ |Fa\ Fo\ Gr][Fa\ Go\ |Fo\ Gr] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go][Fa\ Go\ Gr\ |Fo][Go\ |Fa\ Fo\ Gr][Fa\ Fo\ Go\ |Gr] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Fo\ |Fa\ Go\ Gr] \rangle)$
- $G = (\langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go][Fa\ Go\ Gr\ |Fo][Go\ |Fa\ Fo\ Gr][Fa\ Go\ |Fo\ Gr][|Fa\ Fo\ Go\ Gr] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Gr\ |Fa\ Fo\ Go][Fa\ Go\ Gr\ |Fo][Go\ |Fa\ Fo\ Gr][Fa\ Fo\ Go\ |Gr] \rangle, \langle [Fa\ Fo\ Go\ Gr\ |][Fo\ Gr\ |Fa\ Go][Fa\ Fo\ Gr\ |Go][Fo\ |Fa\ Go\ Gr] \rangle)$

# Depth-first search (search tree)



# Breadth-first search (search tree)



# Exercises: Artificial Intelligence

Bidirectional Search

Bidirectional Search

# **PROBLEM 1: BREADTH-FIRST?**

# Other methods than 2 x breadth-first

- Bidirectional search is complete for each combination with at least one complete search-strategy.
  - 2 x Breadth-first
  - 2 x Depth-first
  - Breadth-first and Depth-first
- Not each combination benefits from searching at both ends.

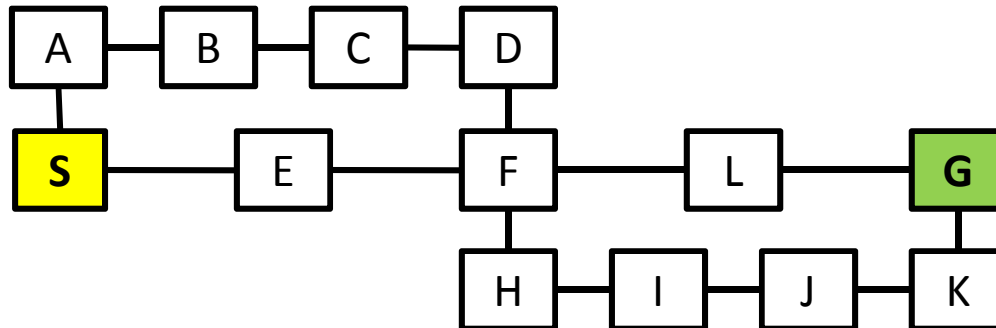
# 2 x Depth-first

- Forward:

- ( $\langle S \rangle$ )  $\rightarrow$  ( $\langle SA \rangle, \langle SE \rangle$ )  $\rightarrow$  ( $\langle SAB \rangle, \langle SE \rangle$ )  $\rightarrow$  ( $\langle SABCD \rangle, \langle SE \rangle$ )  
 $\rightarrow$  ( $\langle SABCDF \rangle, \langle SE \rangle$ )

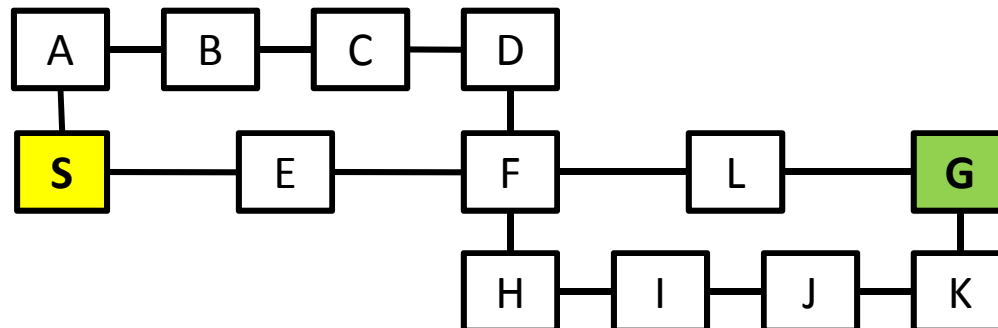
- Backward:

- ( $\langle G \rangle$ )  $\rightarrow$  ( $\langle GK \rangle, \langle GL \rangle$ )  $\rightarrow$  ( $\langle GKJ \rangle, \langle GL \rangle$ )  $\rightarrow$  ( $\langle GKJI \rangle, \langle GL \rangle$ )  
 $\rightarrow$  ( $\langle GKJIH \rangle, \langle GL \rangle$ )  $\rightarrow$  ( $\langle **GKJIHF** \rangle, \langle **GL** \rangle$ )



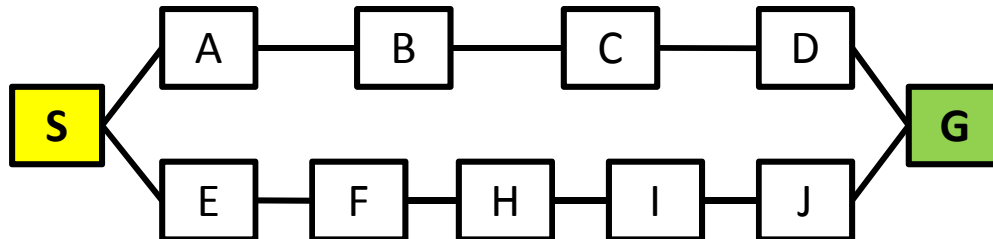
# 2 x Breadth-first

- Forward:
  - $(\langle S \rangle) \rightarrow (\langle SA \rangle, \langle SE \rangle) \rightarrow (\langle SE \rangle, \langle SAB \rangle) \rightarrow (\langle \mathbf{SAB} \rangle, \langle \mathbf{SEF} \rangle)$
- Backward:
  - $(\langle G \rangle) \rightarrow (\langle GK \rangle, \langle GL \rangle) \rightarrow (\langle GL \rangle, \langle GKJ \rangle) \rightarrow (\langle \mathbf{GKJ} \rangle, \langle \mathbf{GLF} \rangle)$



# Breadth-first and Depth-first

- Forward (Breadth-first):
  - ( $\langle S \rangle$ )  $\rightarrow$  ( $\langle SA \rangle, \langle SE \rangle$ )  $\rightarrow$  ( $\langle SE \rangle, \langle SAB \rangle$ )  $\rightarrow$  ( $\langle SAB \rangle, \langle SEF \rangle$ )  
 $\rightarrow$  ( **$\langle SE\underline{F} \rangle, \langle SABC \rangle$** )
- Backward (Depth-first):
  - ( $\langle G \rangle$ )  $\rightarrow$  ( $\langle GJ \rangle, \langle GD \rangle$ )  $\rightarrow$  ( $\langle GJI \rangle, \langle GD \rangle$ )  $\rightarrow$  ( $\langle GJIH \rangle, \langle GD \rangle$ )  
 $\rightarrow$  ( **$\langle GJIH\underline{F} \rangle, \langle GD \rangle$** )

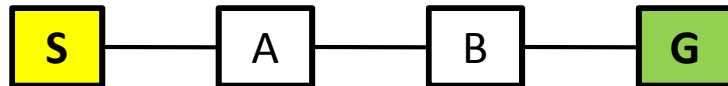


Bidirectional Search

## **PROBLEM 2: SHARED-STATE CHECK?**

# Replace shared-state check

- When only checking identical end-states, paths can cross each other unnoticed.
- Forward:
  - $(\langle S \rangle) \rightarrow (\langle SA \rangle) \rightarrow (\langle SAB \rangle) \rightarrow (\langle SABG \rangle)$
- Backward:
  - $(\langle G \rangle) \rightarrow (\langle GB \rangle) \rightarrow (\langle GBA \rangle) \rightarrow (\langle GBAS \rangle)$



# Exercises: Artificial Intelligence

Beam Search

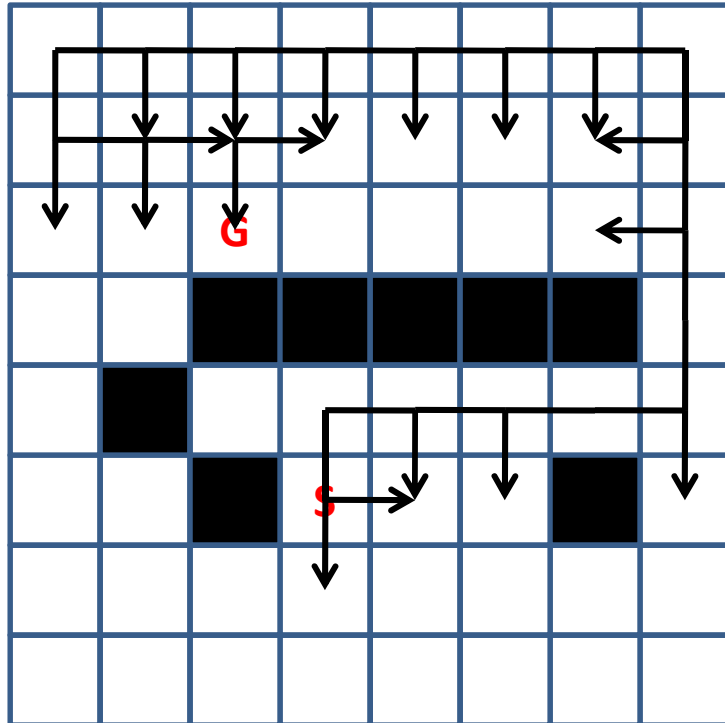
# Beam Search

- ***Input:***
  - **QUEUE:** Path only containing root
  - **WIDTH:** Number
- ***Algorithm:***
  - **WHILE** (QUEUE not empty && goal not reached) **DO**
    - Remove ***all paths*** from QUEUE
    - Create paths to all children (of all paths)
    - Reject paths with loops
    - ***Sort new paths (according to heuristic)***
    - ***(Optimization: Remove paths without successor)***
    - Add WIDTH ***best paths*** to QUEUE
  - **IF** goal reached
    - **THEN** success
    - **ELSE** failure

# Exercises: Artificial Intelligence

Path Search

# Depth-first Search



17	16	15	14	13	12	11	10
18	19	20					9
		G					8
							7
							6
							5
							4
							3
							2
							1
							0
							0

A 10x8 grid with numbered cells (10-20) and obstacles. The start cell 'S' is at (6,3) and the goal cell 'G' is at (3,2). Black obstacles are at (3,2), (3,3), (3,4), (3,5), (3,6), (3,7), (4,1), (5,2), (5,3), (5,4), (5,5), (5,6), (5,7), (6,3), and (6,7).

# Heuristic: Manhattan Distance

4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	0	1	2	3	4	5
3	2						6
4		2	3	4	5	6	7
5	4		4	5	6		8
6	5	4	5	6	7	8	9
7	6	5	6	7	8	9	10

# Hill-climbing I Search

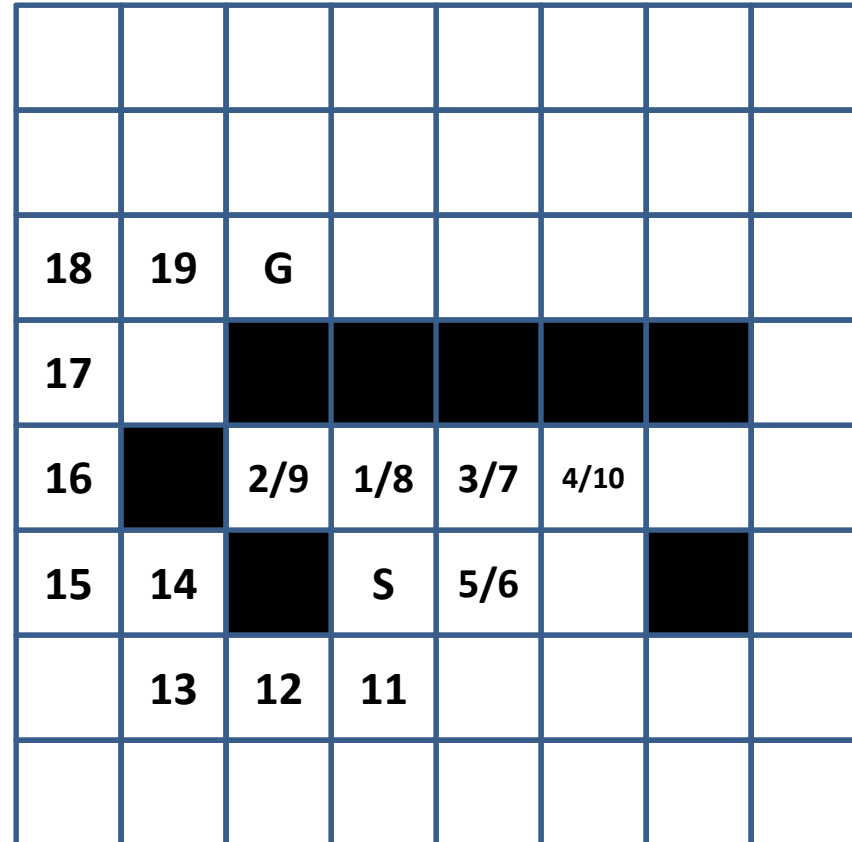
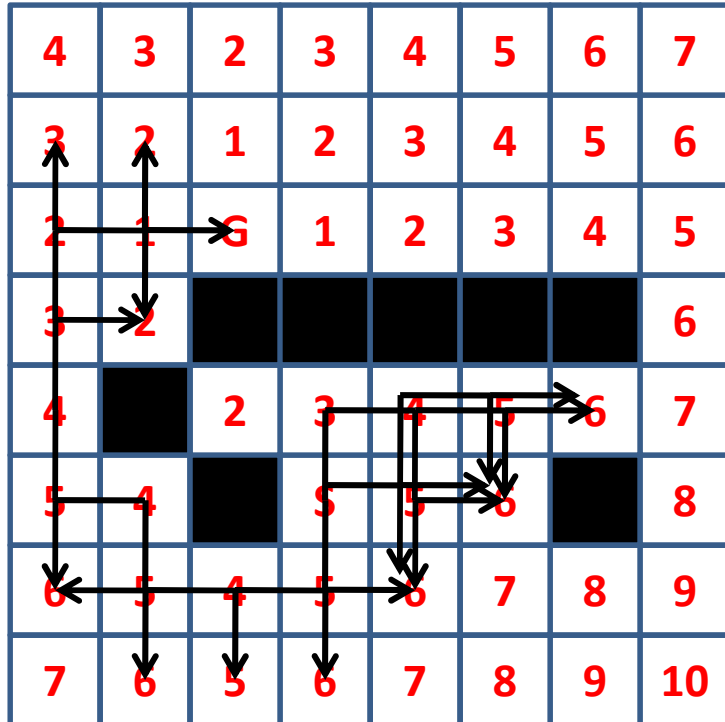
4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	G	1	2	3	4	5
3	2						6
4		2	3	4	5	6	7
5	4		5	6			8
6	5	4	5	6	7	8	9
7	6	5	6	7	8	9	10

Diagram illustrating a grid with values and search paths. The grid contains numbers 1-10 and a goal 'G'. Blacked-out cells represent obstacles. Arrows indicate the search path starting from 'G' and moving towards higher values.

		G	12	11	10	9	8	
							7	
			2	1	3	4	5	6
				S				

Diagram illustrating a grid with values and search paths. The grid contains numbers 1-12 and a goal 'G'. Blacked-out cells represent obstacles. Arrows indicate the search path starting from 'S' and moving towards higher values.

# Greedy Search



# Exercises: Artificial Intelligence

Water Jugs

# Representation

- States of the form  $[x,y]$ , where:
  - $x$ : *contents of 4 liter jug*
  - $y$ : *contents of 3 liter jug*
- Start:  $[0,0]$
- Goal:  $[2,0]$

# Representation

- Rules:

- Fill x:  $[x,y] \wedge x < 4 \longrightarrow [4,y]$

- Fill y:  $[x,y] \wedge y < 3 \longrightarrow [x,3]$

- Empty x:  $[x,y] \wedge x > 0 \longrightarrow [0,y]$

- Empty y:  $[x,y] \wedge y > 0 \longrightarrow [x,0]$

- Fill x with y:  $[x,y] \wedge x+y > 4 \wedge y > 0 \longrightarrow [4,(x+y-4)]$

- Fill x with y:  $[x,y] \wedge x+y \leq 4 \wedge y > 0 \longrightarrow [(x+y),0]$

- Fill y with x:  $[x,y] \wedge x+y > 3 \wedge x > 0 \longrightarrow [(x+y-3),3]$

- Fill y with x:  $[x,y] \wedge x+y \leq 3 \wedge x > 0 \longrightarrow [0,(x+y)]$

# Heuristic

- $H([x,y]) = f(x) + f(y)$
- $f(x)$  is defined as follows:

x	0	1	2	3	4
f(x)	2	1	0	1	3

- We need a jug filled with 2 liter.
- To obtain a jug filled with 2 liter we need a jug filled with either 1 or 3 liter.
- We consider an empty jug better than a jug filled with 4 liter.

# Hill-climbing II Search

