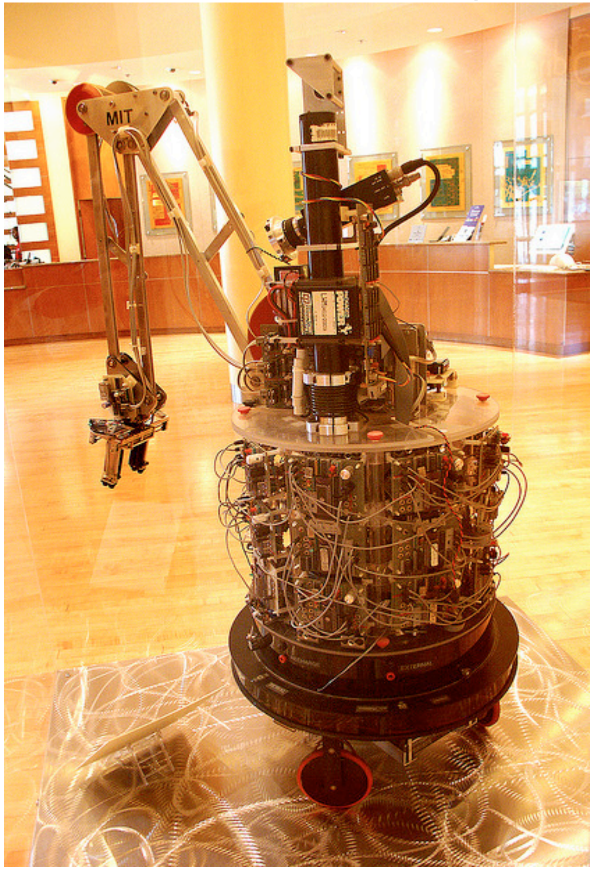
Genetic Algorithm Example:

Evolving a Control Program for a Virtual “Robot”

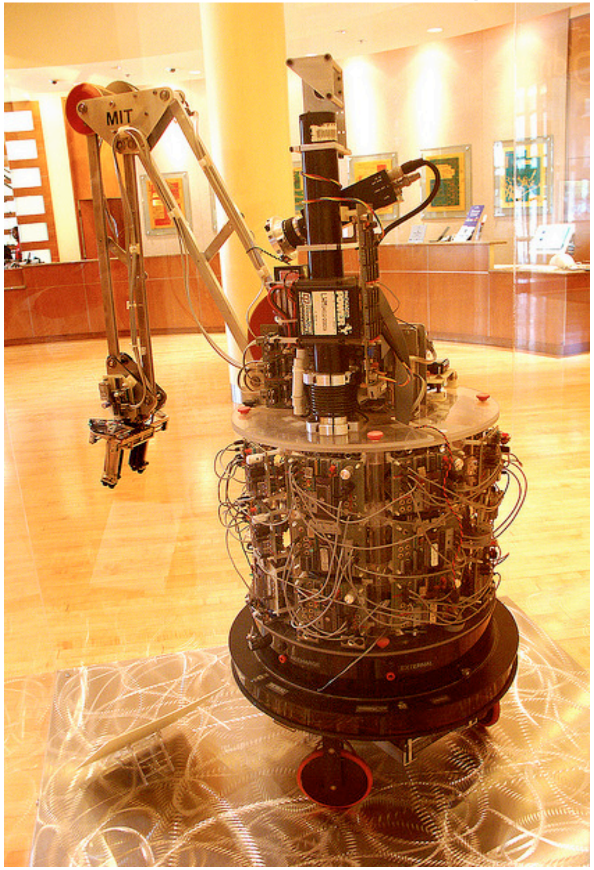
Herbert:

The Soda Can Collecting Robot
(Connell, Brooks, Ning, 1988)



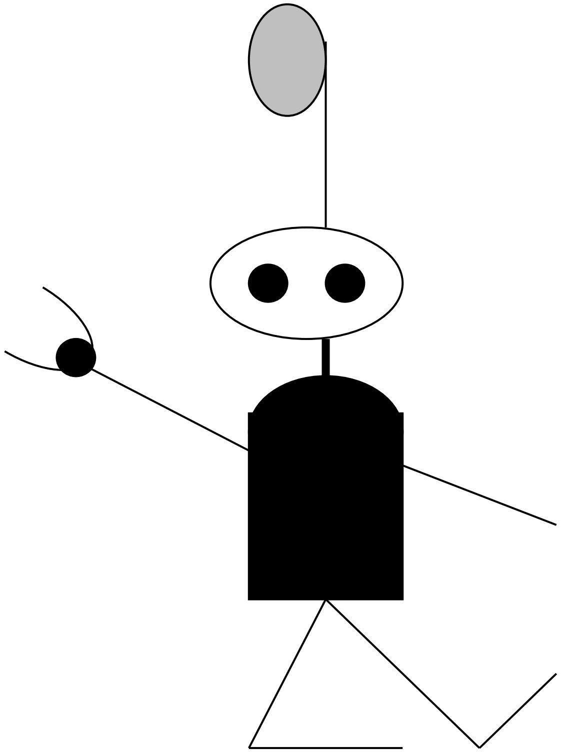
http://cyberneticzoo.com/?p=5516

Herbert:
The Soda Can Collecting Robot
(Connell, Brooks, Ning, 1988)

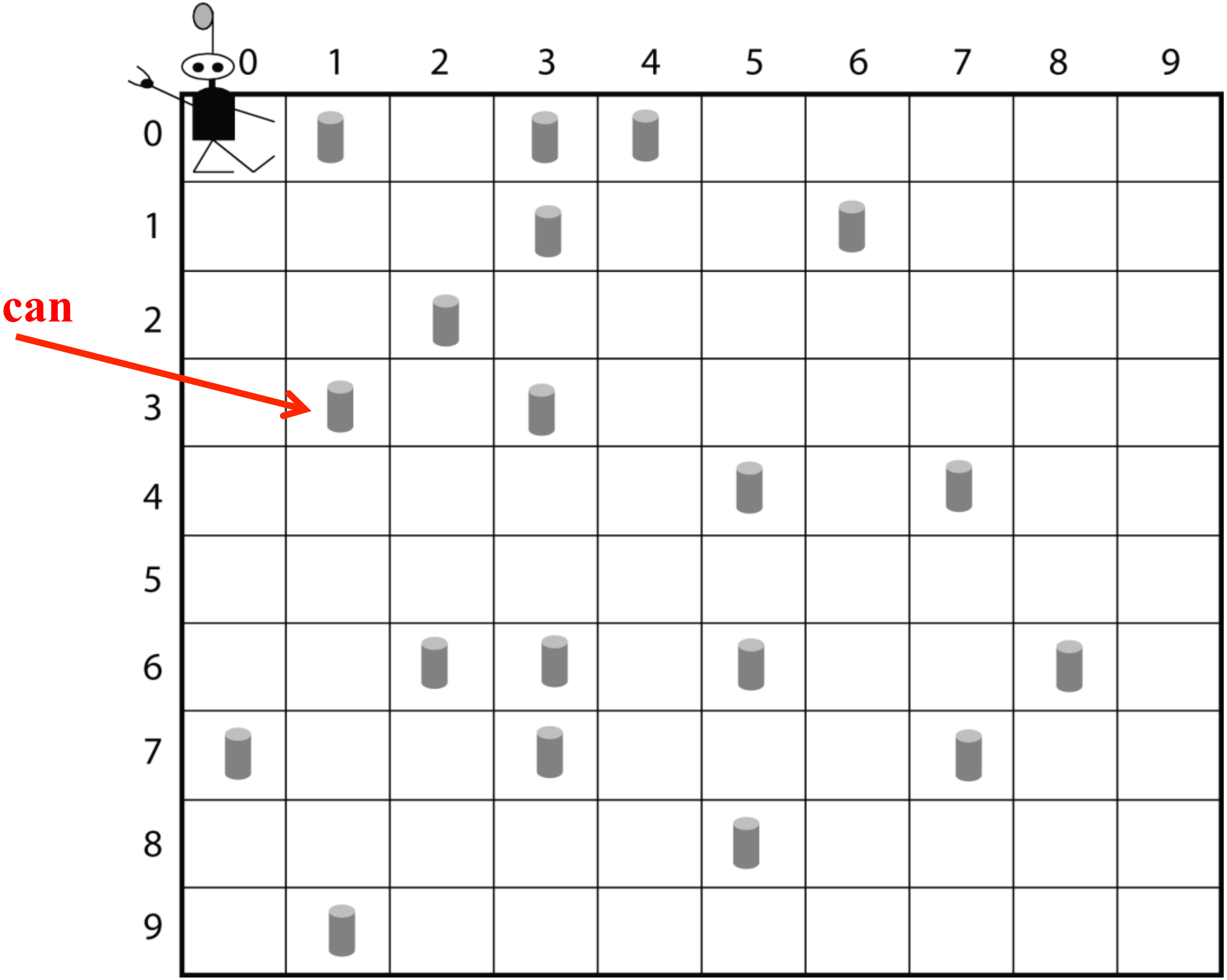


http://cyberneticzoo.com/?p=5516

Robby:
The Virtual Soda Can Collecting Robot
(Mitchell, 2009)



Robby’s World



Soda can

What Robby Can See and Do

Input:

Contents of **N**orth, **S**outh, **E**ast, **W**est,
Current

Possible actions:

`Move N`

`Move S`

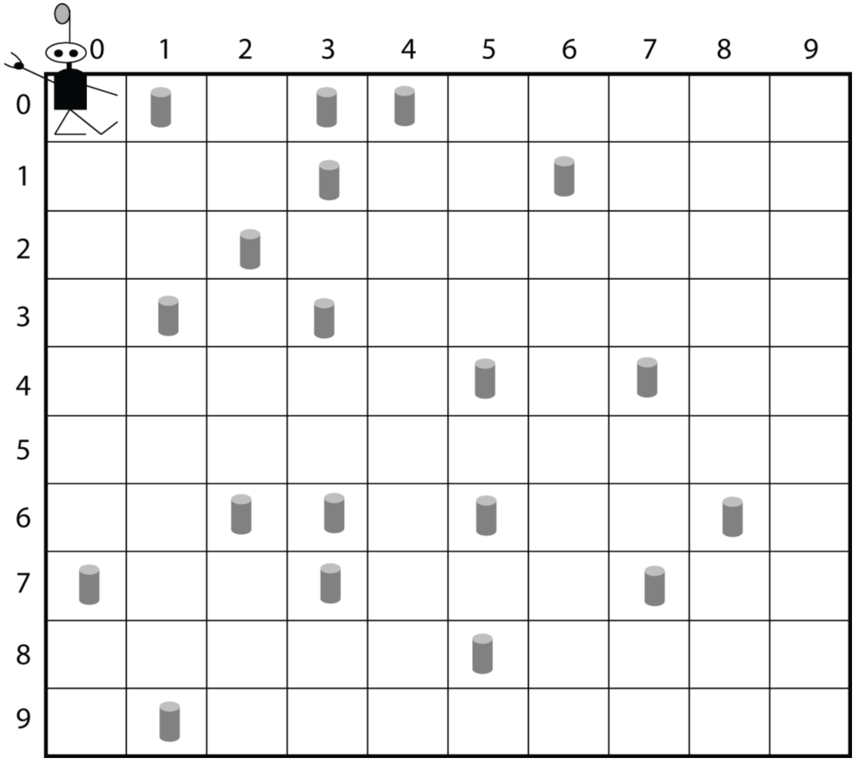
`Move E`

`Move W`

`Move random`

`Stay put`

`Try to pick up can`



Rewards/Penalties (points):

Picks up can: **10**

Tries to pick up can on empty site: **-1**

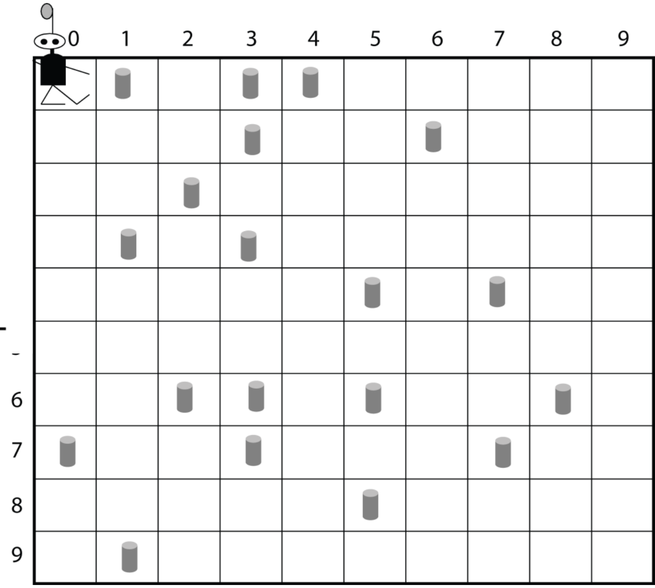
Crashes into wall: **-5**

Robby's Score: Sum of rewards/penalties

Goal: Use a genetic algorithm to evolve a control program (i.e., *strategy*) for Robby.

One Example Strategy

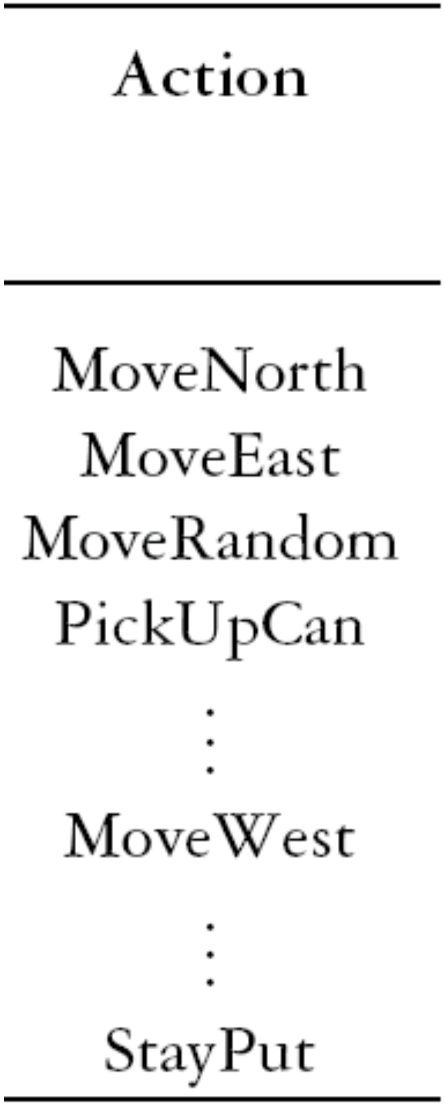
	Situation					Action
	North	*South*	*East*	*West*	*Current Site*	
1	Empty	Empty	Empty	Empty	Empty	MoveNorth
2	Empty	Empty	Empty	Empty	Can	MoveEast
3	Empty	Empty	Empty	Empty	Wall	MoveRandom
4	Empty	Empty	Empty	Can	Empty	PickUpCan
.	⋮	⋮	⋮	⋮	⋮	⋮
.	Wall	Empty	Can	Wall	Empty	MoveWest
.	⋮	⋮	⋮	⋮	⋮	⋮
243	Wall	Wall	Wall	Wall	Wall	StayPut



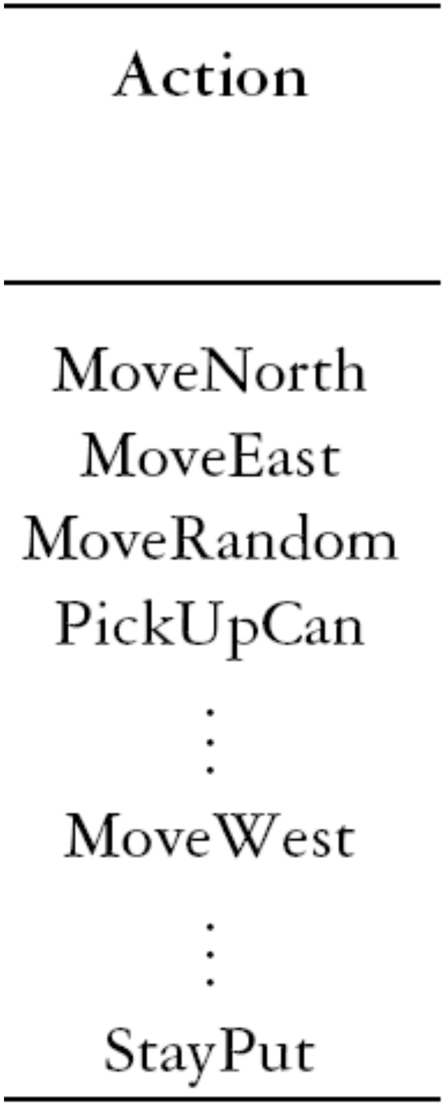
Encoding a Strategy

	Situation				
	North	*South*	*East*	*West*	*Current Site*
1	Empty	Empty	Empty	Empty	Empty
2	Empty	Empty	Empty	Empty	Can
3	Empty	Empty	Empty	Empty	Wall
4	Empty	Empty	Empty	Can	Empty
.	⋮	⋮	⋮	⋮	⋮
.	Wall	Empty	Can	Wall	Empty
.	⋮	⋮	⋮	⋮	⋮
243	Wall	Wall	Wall	Wall	Wall

	Action
1	MoveNorth
2	MoveEast
3	MoveRandom
4	PickUpCan
.	⋮
.	
.	MoveWest
	⋮
243	StayPut

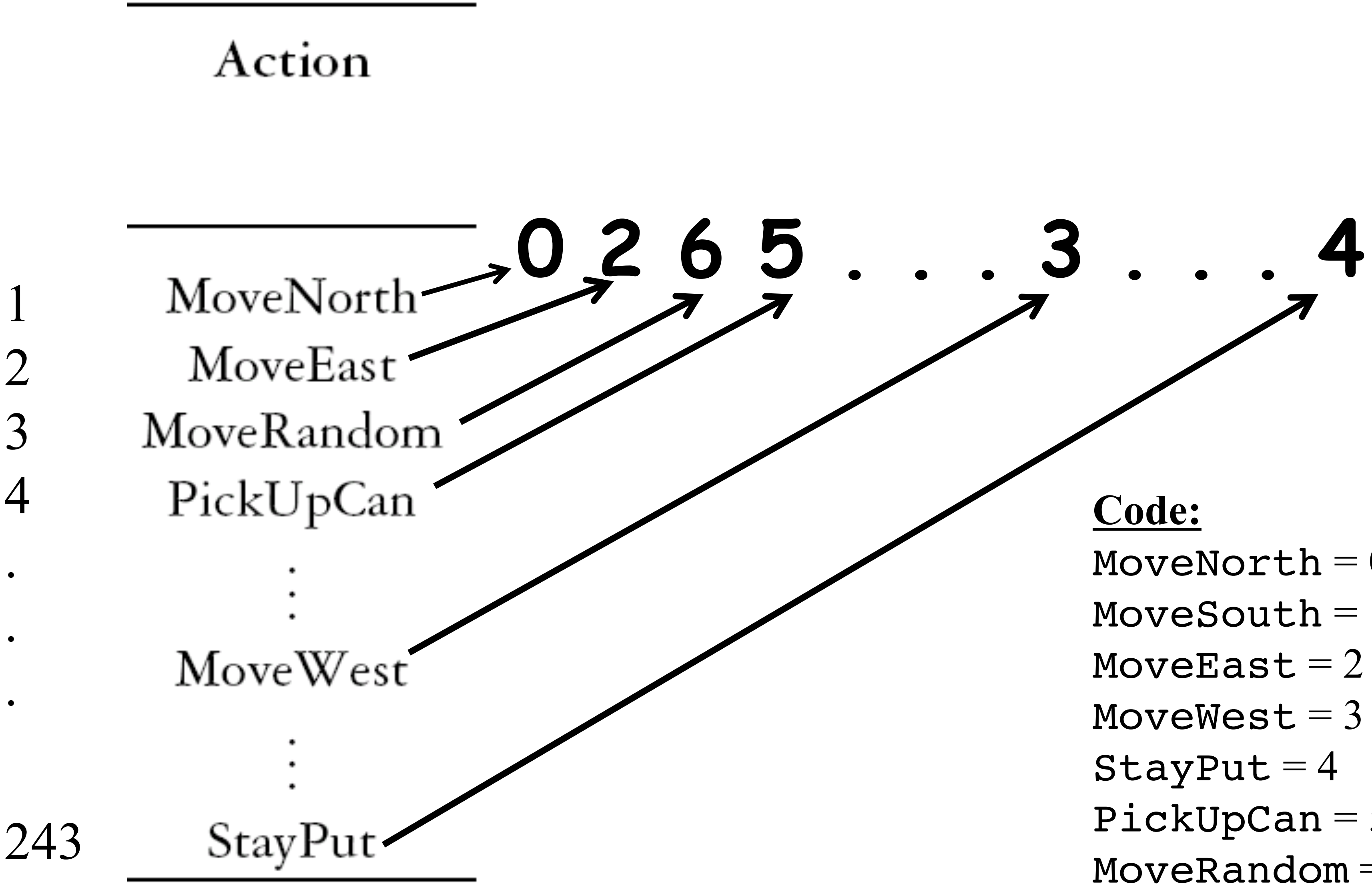


	Action
1	MoveNorth
2	MoveEast
3	MoveRandom
4	PickUpCan
.	⋮
.	MoveWest
.	⋮
243	StayPut



	Action
1	MoveNorth
2	MoveEast
3	MoveRandom
4	PickUpCan
.	⋮
.	MoveWest
.	
	⋮
243	StayPut

Code:
MoveNorth = 0
MoveSouth = 1
MoveEast = 2
MoveWest = 3
StayPut = 4
PickUpCan = 5
MoveRandom = 6



	Action
1	MoveNorth
2	MoveEast
3	MoveRandom
4	PickUpCan
.	⋮
.	MoveWest
.	⋮
243	StayPut

0 2 6 5 . . . 3 . . . 4

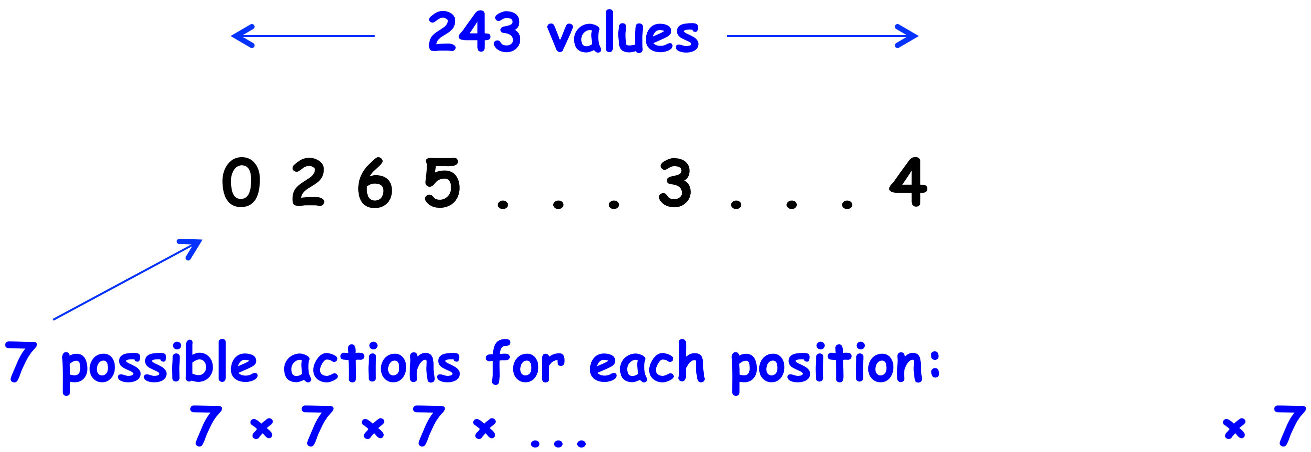
Code:

`MoveNorth` = 0
`MoveSouth` = 1
`MoveEast` = 2
`MoveWest` = 3
`StayPut` = 4
`PickUpCan` = 5
`MoveRandom` = 6

Question: How many possible strategies are there in our representation?

0 2 6 5 . . . 3 . . . 4

Question: How many possible strategies are there in our representation?



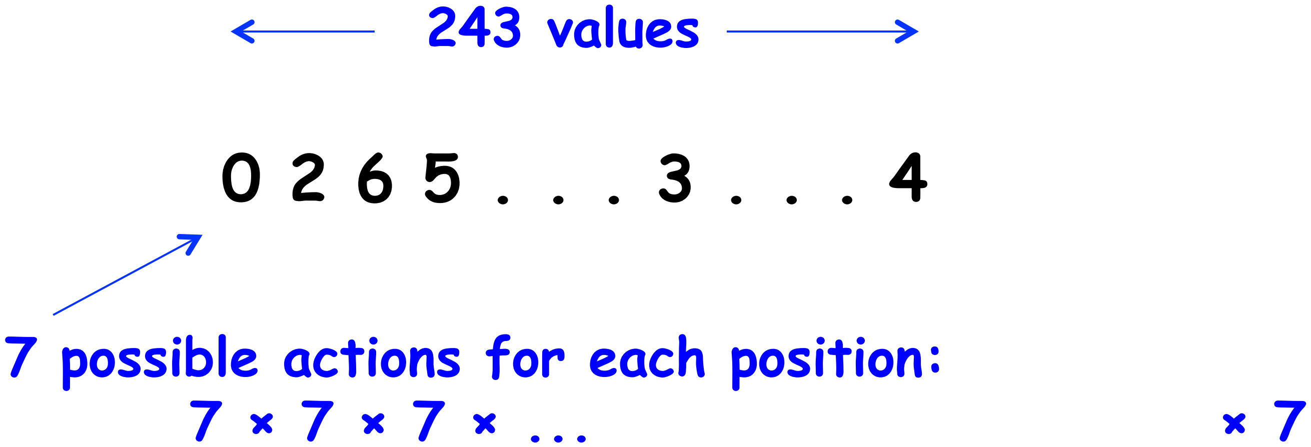
← 243 values →

0 2 6 5 . . . 3 . . . 4

7 possible actions for each position:

7 × 7 × 7 × ... × 7

Question: How many possible strategies are there in our representation?



243 values

0 2 6 5 . . . 3 . . . 4

7 possible actions for each position:

$7 \times 7 \times 7 \times \ldots \times 7$

Goal: Have GA search intelligently in this vast space for a good strategy

Genetic algorithm for evolving strategies

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy, calculate fitness (average reward minus penalties earned on random environments)

3. The strategies pair up and create offspring via crossover with random mutations — the fitter the parents, the more offspring they create.

4. Keep going back to step 2 until a good-enough strategy is found (or for a set number of generations)

Robby’s fitness function

```
Calculate_Fitness (Robby) {
  Total_Reward = 0 ;
  Average_Reward = 0 `
  For i = 1 to NUM_ENVIRONMENTS {
    generate_random_environment( ); /* .5 probability
                                     * to place can at
                                     * each site */
    For j = 1 to NUM_MOVES_PER_ENVIRONMENT {
      Total_Reward = Total_Reward + perform_action(Robby);
    }
  }

  Fitness = Total_Reward / NUM_ENVIRONMENTS;
  return(Fitness);
}
```

Genetic algorithm for evolving strategies for Robby

1. Generate 200 random strategies (i.e., programs for controlling Robby)

Random Initial Population

Individual 1:
2330032342163034353054600610256251511416226043565433406651151 4
1565022064064205100664321616152165202236443336334601332650300 0
4062205024316500611130514666423240124563334552412614344136102 0
150630642551654043264463156164510543665346310551646005164

Individual 2:
164113431210253603403612414312011042354625253042020445164336 65
610353221531051314406221206146314321546102565236444220253403 45
305020056206340263310024534164301516312100122144006640126652 46
351650154123113132453304433212634555005314213064423311000

Individual 3:
204233444024112261321364526324642122061221222526606261444361 25
325126640613353401534111102061642266531455225402340511550313 02
220200654451250622066314261355320100004000316401301541601620 06
134440626160505641421553133236021503355131253632642630551

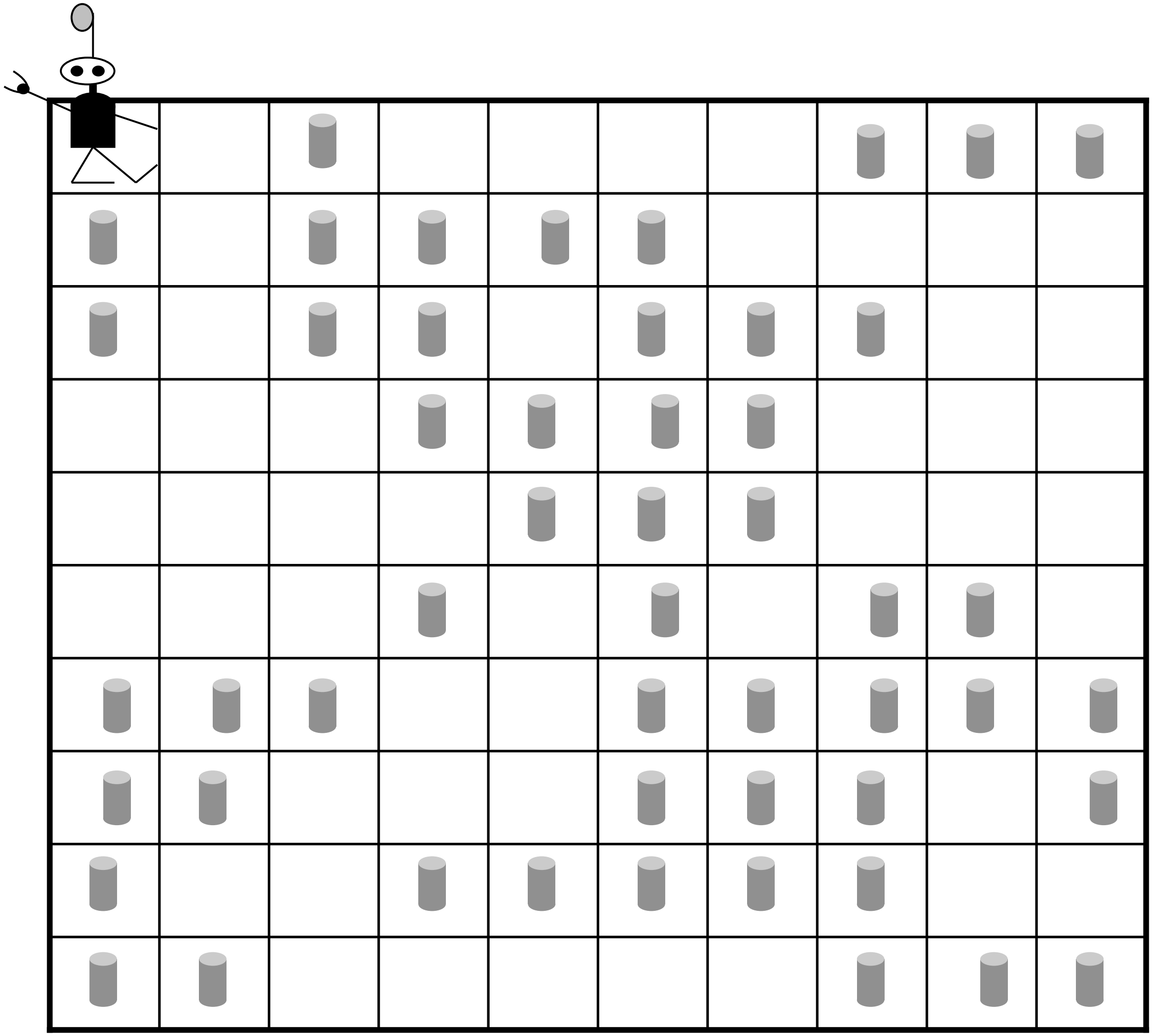
.
.
.

Individual 200:
346325251360010122256121060433011352051553201306560053222350 43
324250641242552655346353455230533266120106321245544234406136 54
302462401606630164646411030265400063341261503522621060636242 60
550616616344255124354464110023463330440102533212142402251

Genetic algorithm for evolving strategies for Robby

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy in the population, calculate fitness (average reward minus penalties earned on random environments)



Fitness =
Average final score from
N moves on each of M
random environments

Genetic algorithm for evolving strategies for Robby

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy in the population, calculate fitness (average reward minus penalties earned on random environments)

3. Strategies are selected according to fitness to become parents. (See code for choice of selection methods.)

Genetic algorithm for evolving strategies for Robby

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy in the population, calculate fitness (average reward minus penalties earned on random environments)

3. Strategies are selected according to fitness to become parents. (See code for choice of selection methods.)

4. The parents pair up and create offspring via crossover with random mutations.

Parent 1:

16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
30502005620634026331002453416430151631210012214400664012665246
35165015412311313245330443321263455500531421306442331 1000

Parent 2:

204233444024112261321364526324642122061221222526606261444 36125
325126640613353401534111102061642266531455225402340511 55031302
220200654451250622066314261355320100004000316401301541 60162006
134440626160505641421553133236021503355131253632642630551

Parent 1:

```
164113431210253603403612414312011042354625253042020445164336656
61035322153105131440622120614631432154610256523644422025340345
305020056206340263310024534164301516312100122144006640126652 46
35165015412311313245330443321263455500531421306442331100 0
```

Parent 2:

```
2042334440241122613213645263246421220612212225266062614443612 5
3251266406133534015341111020616422665314552254023405115503130 2
2202006544512506220663142613553201000040003164013015416016200 6
1344406261605056414215531332360215033551312536326426305 51
```

Parent 1:

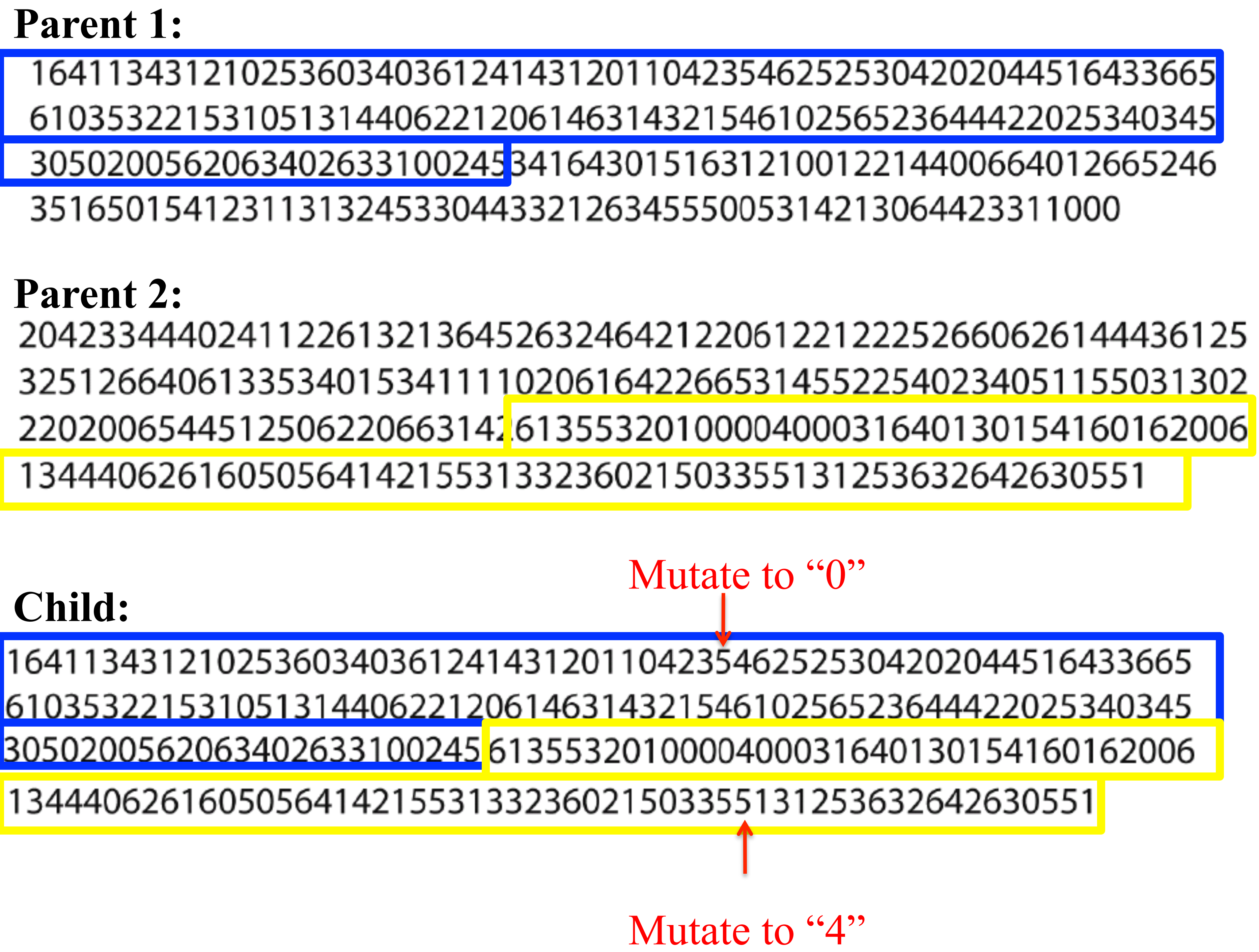
```
16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
30502005620634026331002453416430151631210012214400664012665246
351650154123113132453304433212634555005314213064423311000
```

Parent 2:

```
20423344402411226132136452632464212206122122252660626144436125
32512664061335340153411110206164226653145522540234051155031302
22020065445125062206631426135532010000400031640130154160162006
134440626160505641421553133236021503355131253632642630551
```

Child:

```
16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
3050200562063402633100245 6135532010000400031640130154160162006
134440626160505641421553133236021503355131253632642630551
```



Parent 1:

1641134312102536034036124143120110423546252530420204451643366
5

Parent 1:

16411343121025360340361241431201104235462525304202044516433665
61035322153105131440622120614631432154610256523644422025340345
30502005620634026331002453416430151631210012214400664012665246
351650154123113132453304433212634555005314213064423311000

Genetic algorithm for evolving strategies for Robby

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy in the population, calculate fitness (average reward minus penalties earned on random environments)

3. Strategies are selected according to fitness to become parents. (See code for choice of selection methods.)

4. The parents pair up and create offspring via crossover with random mutations.

5. The offspring are placed in the new population and the old population dies.

Genetic algorithm for evolving strategies for Robby

1. Generate 200 random strategies (i.e., programs for controlling Robby)

2. For each strategy in the population, calculate fitness (average reward minus penalties earned on random environments)

3. Strategies are selected according to fitness to become parents. (See code for choice of selection methods.)

4. The parents pair up and create offspring via crossover with random mutations.

5. The offspring are placed in the new population and the old population dies.

6. Keep going back to step 2 until a good-enough strategy is found!

My hand-designed strategy:

“If there is a can in the current site, pick it up.”

“Otherwise, if there is a can in one of the adjacent sites, move to that site.”

“Otherwise, choose a random direction to move in, avoiding walls.”

My hand-designed strategy:

“If there is a can in the current site, pick it up.”

“Otherwise, if there is a can in one of the adjacent sites, move to that site.”

“Otherwise, choose a random direction to move in, avoiding walls.”

Average fitness of this strategy:

346

(out of max possible ≈ 500)

My hand-designed strategy:

“If there is a can in the current site, pick it up.”

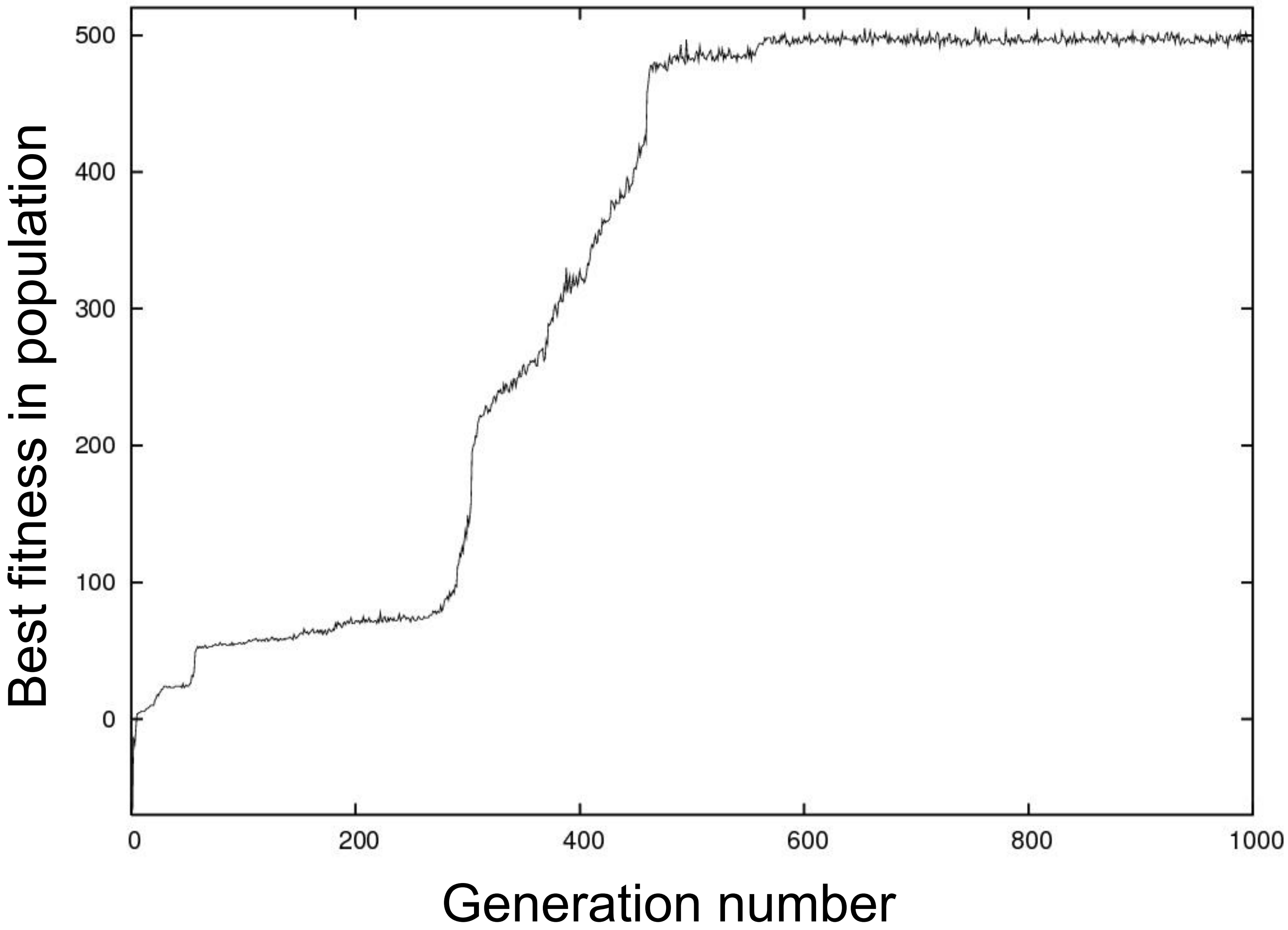
“Otherwise, if there is a can in one of the adjacent sites, move to that site.”

“Otherwise, choose a random direction to move in, avoiding walls.”

Average fitness of this strategy: **346**
(out of max possible ≈ 500)

Average fitness of GA evolved strategy: **486**
(out of max possible ≈ 500)

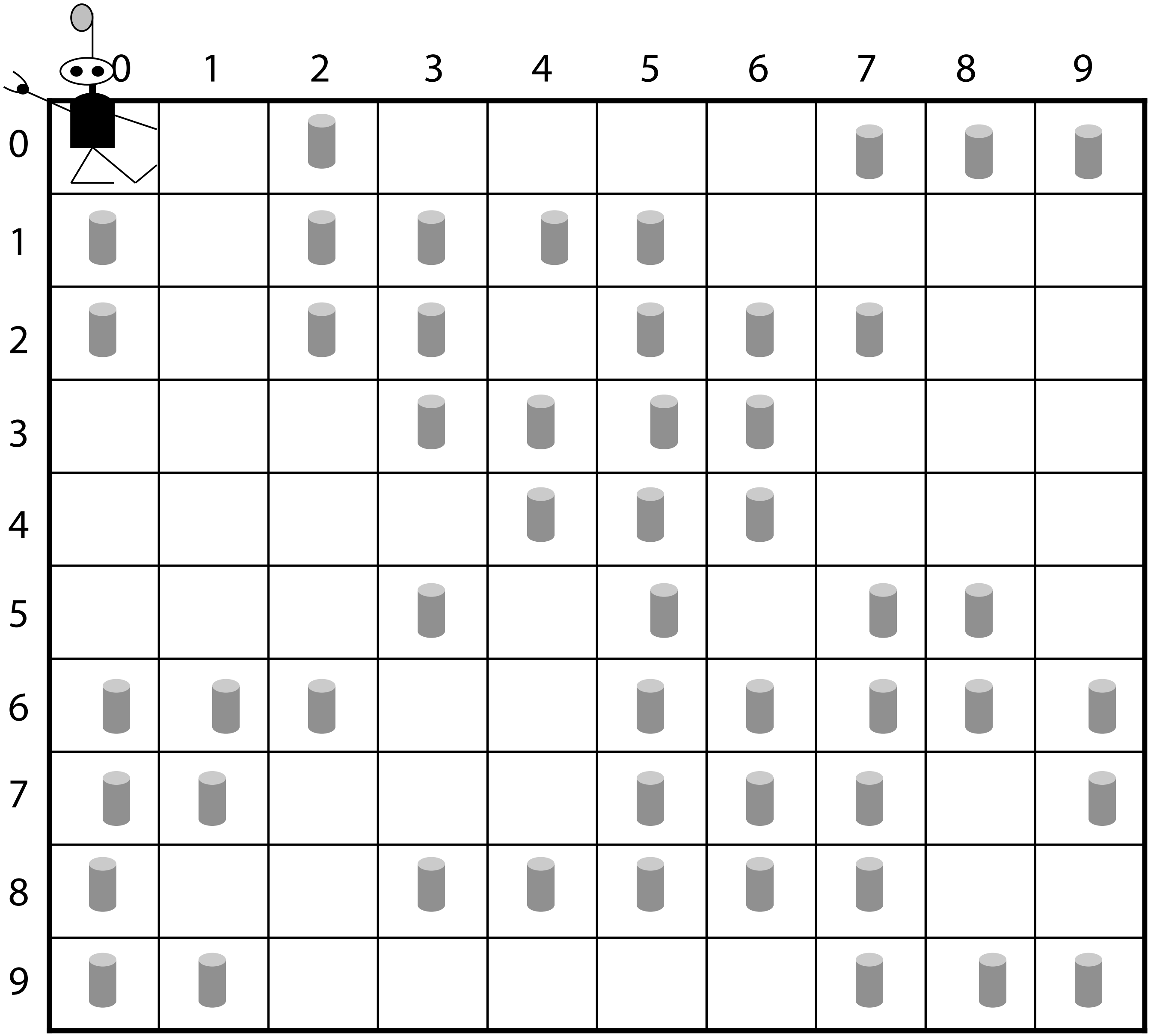
One Run of the Genetic Algorithm



Generation 1

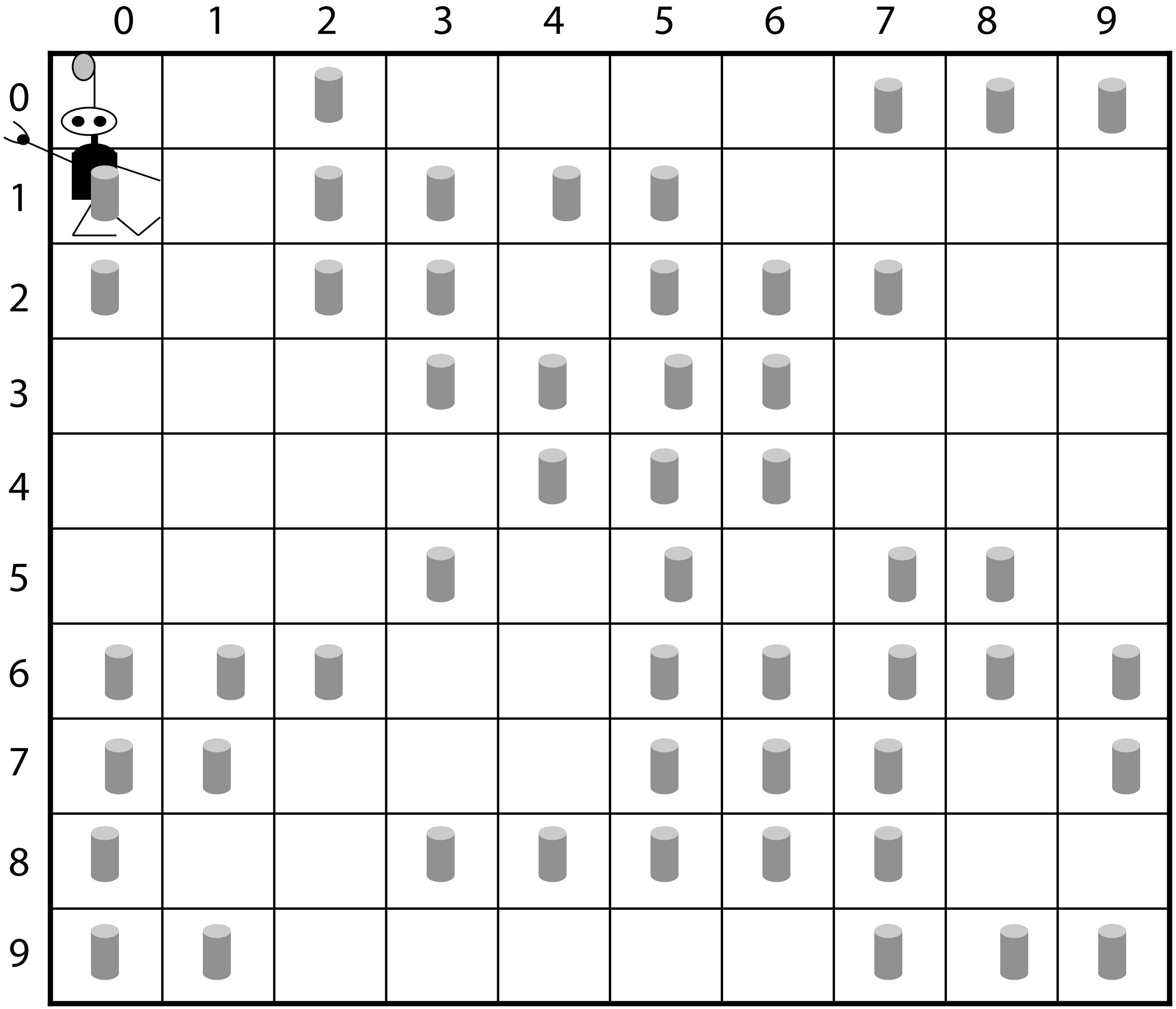
Best fitness = -81

Time: 0 Score: 0

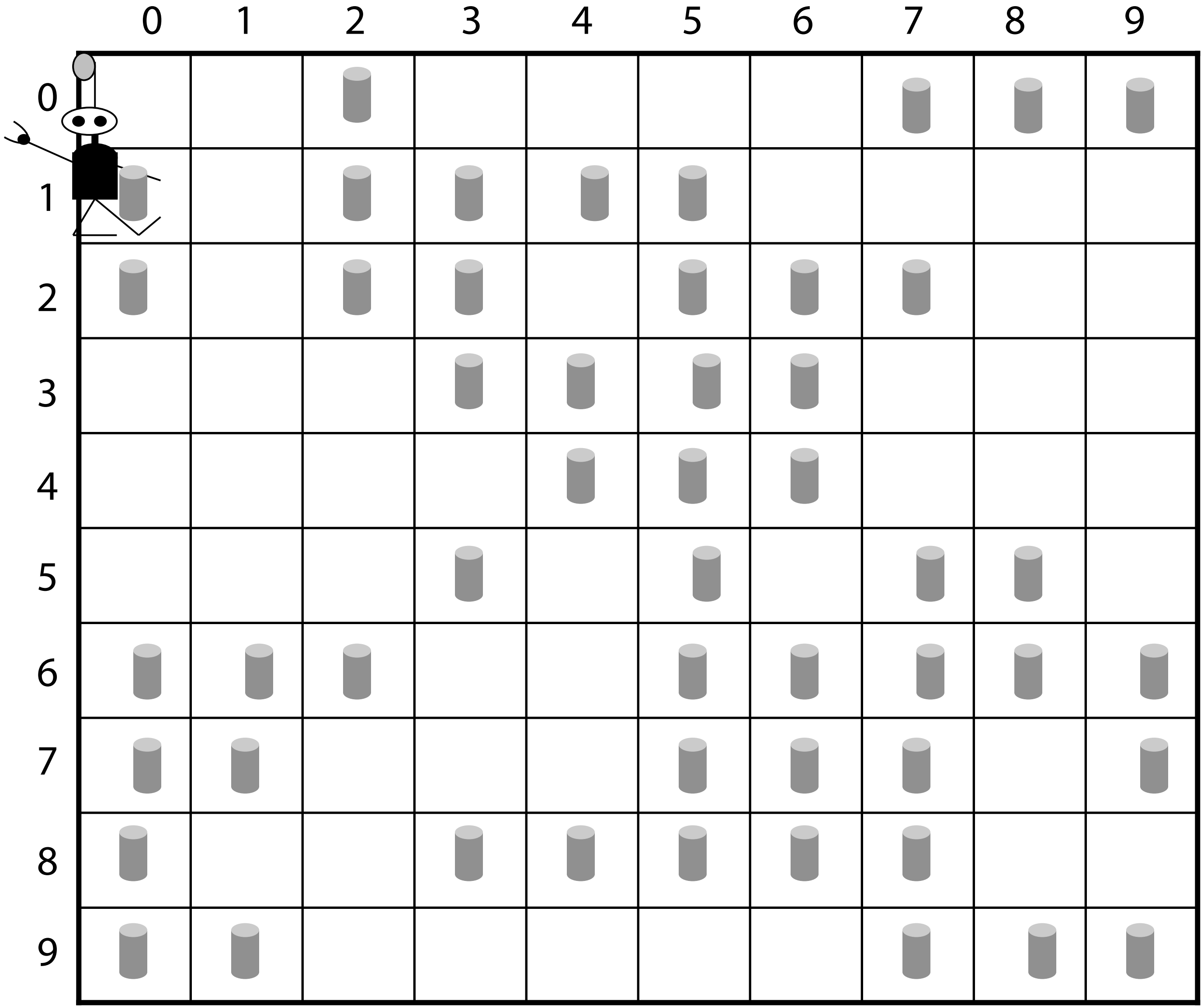


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 1 Score: 0

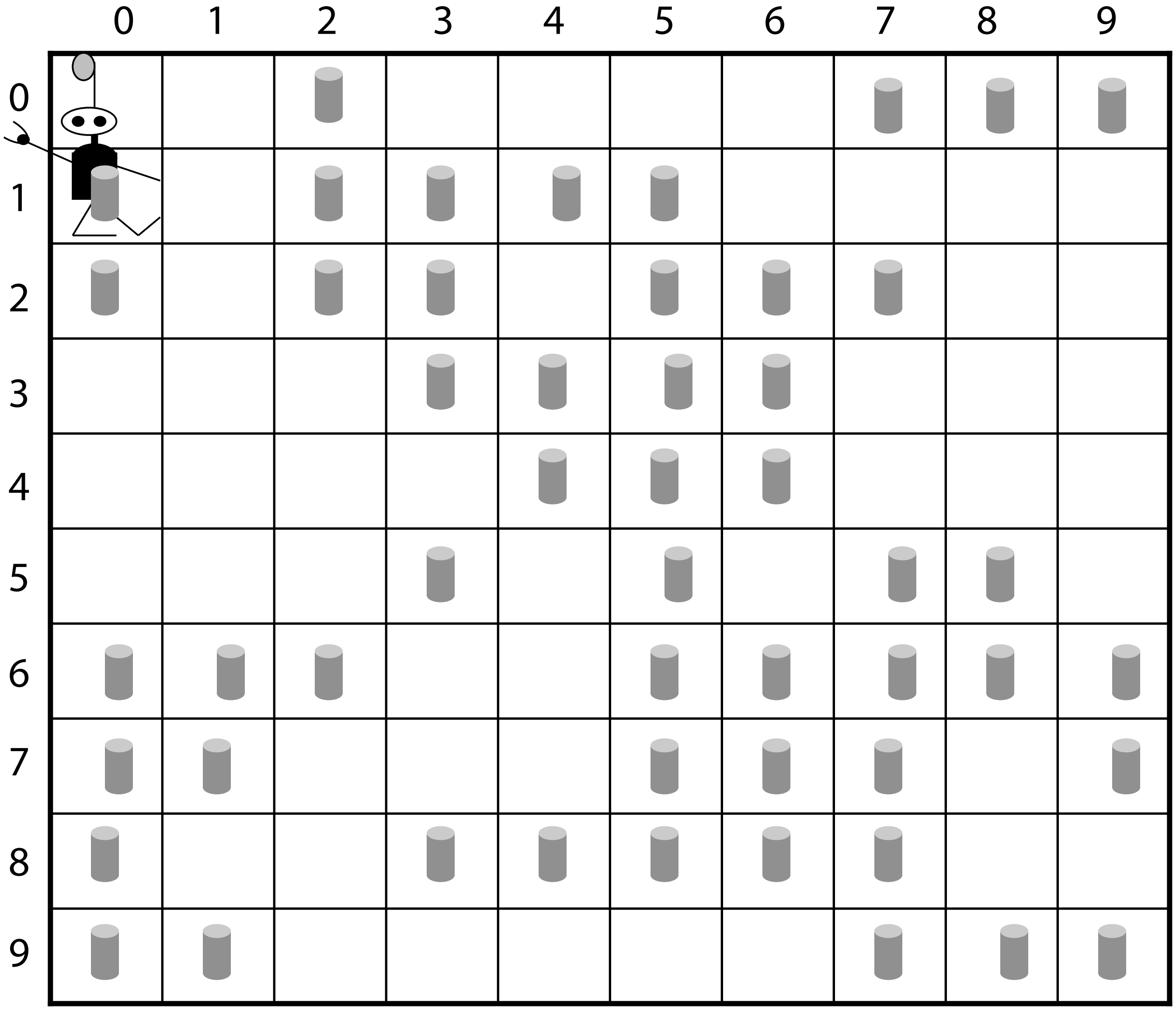


Time: 2 Score: -5

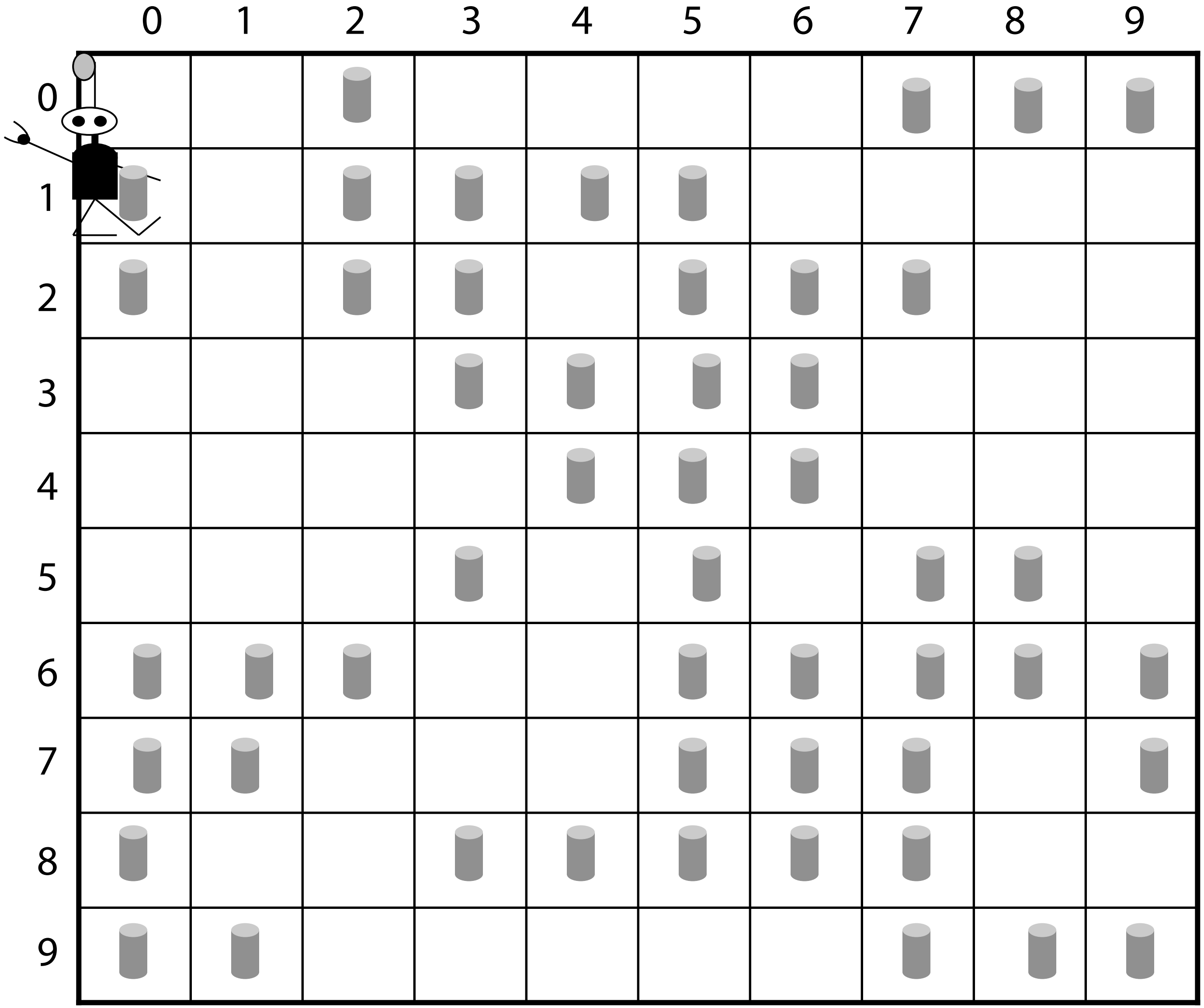


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 2 Score: -5

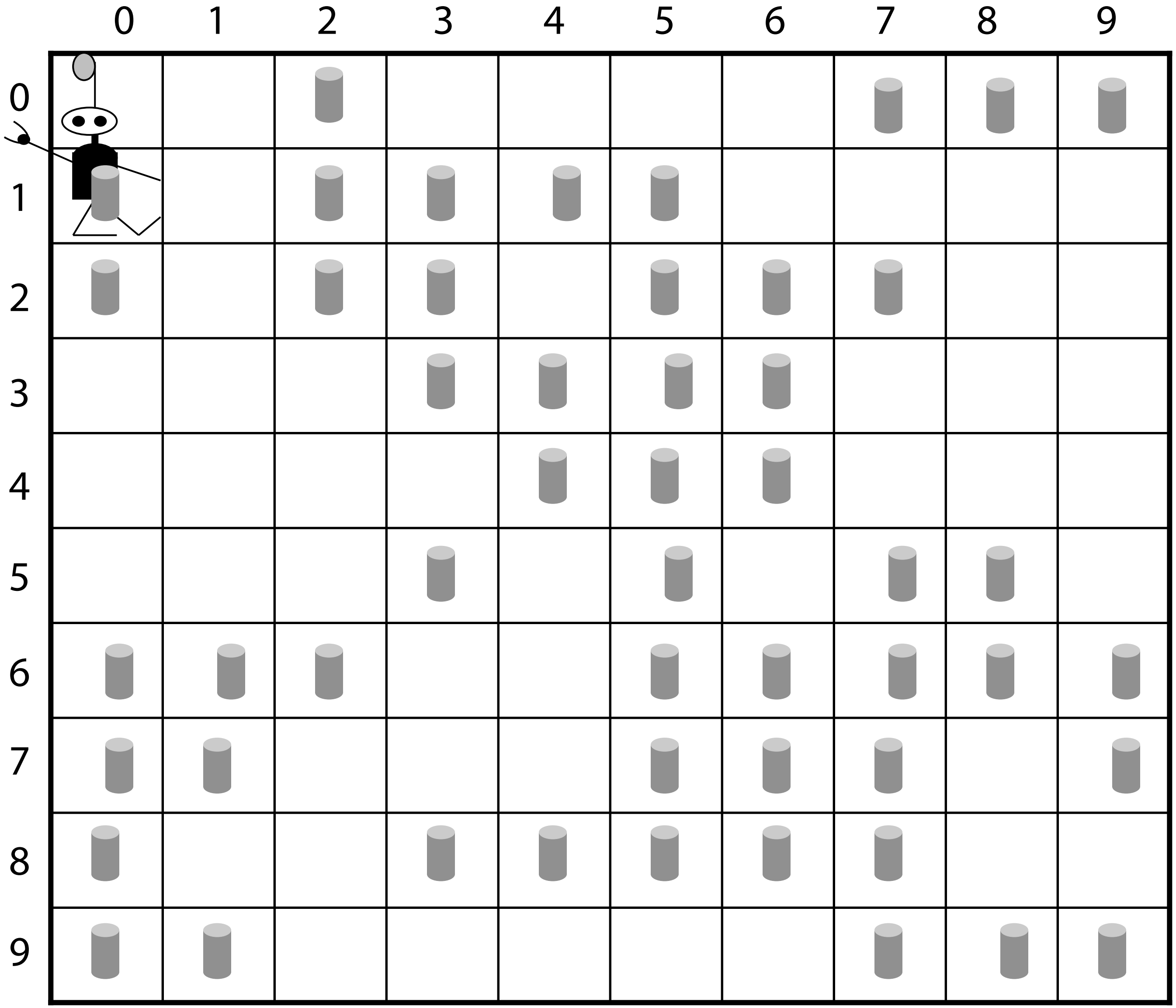


Time: 3   Score: -10

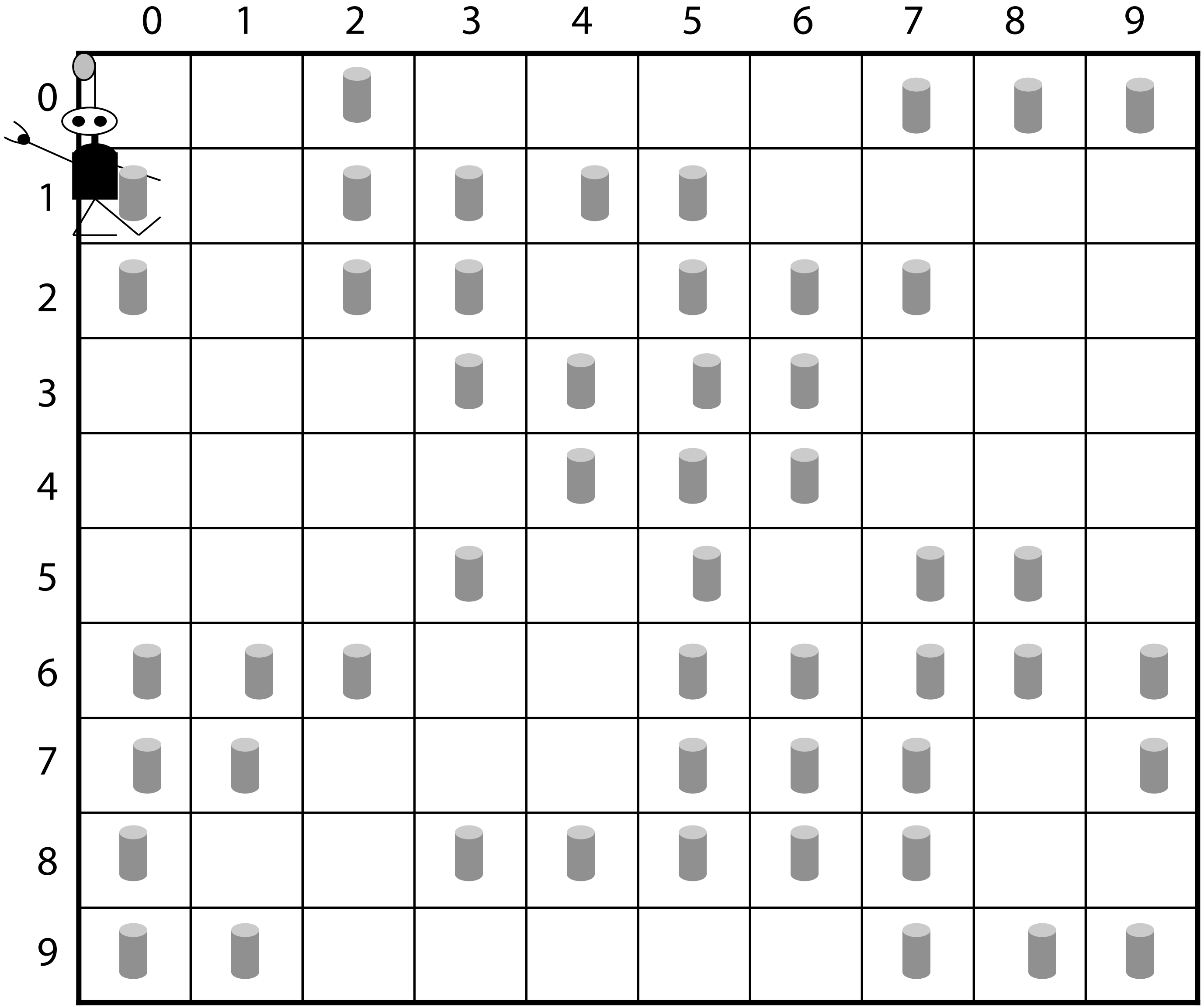


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

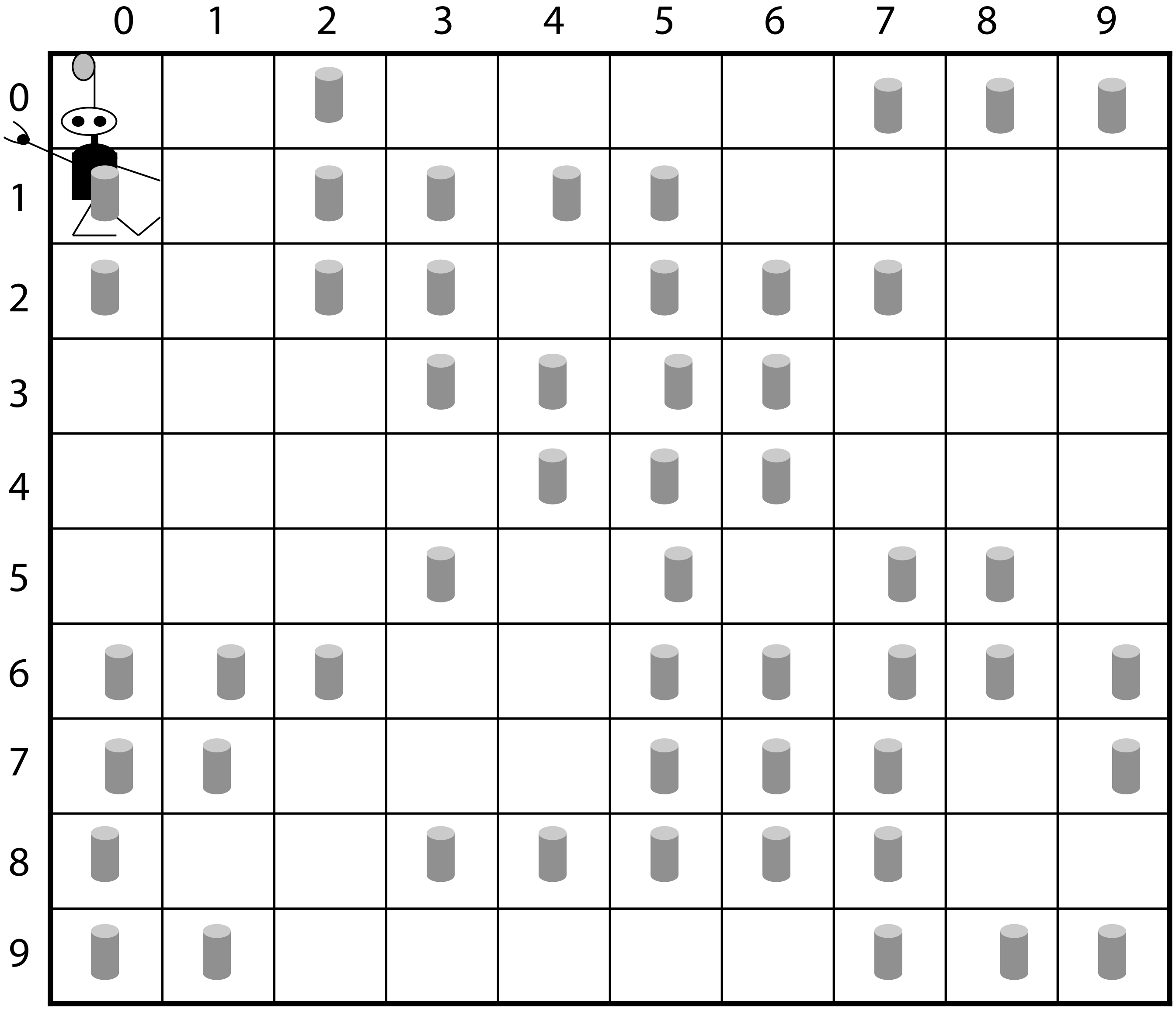
Time: 3 Score: -10



Time: 4 Score: -15



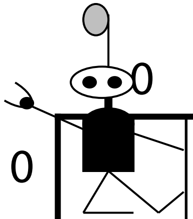
Time: 4   Score: -15



Generation 14

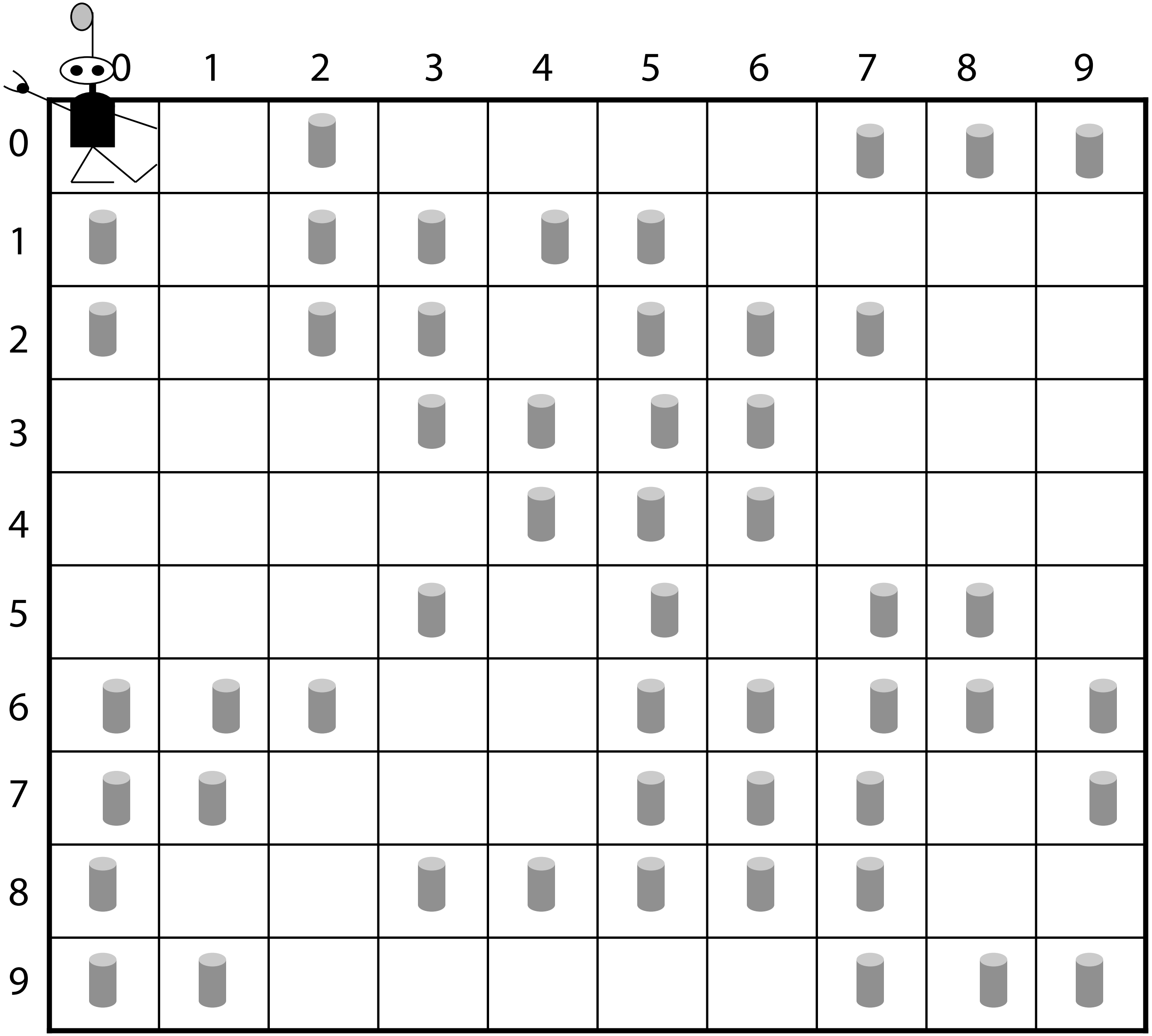
Best fitness = 1

Time: 0 Score: 0



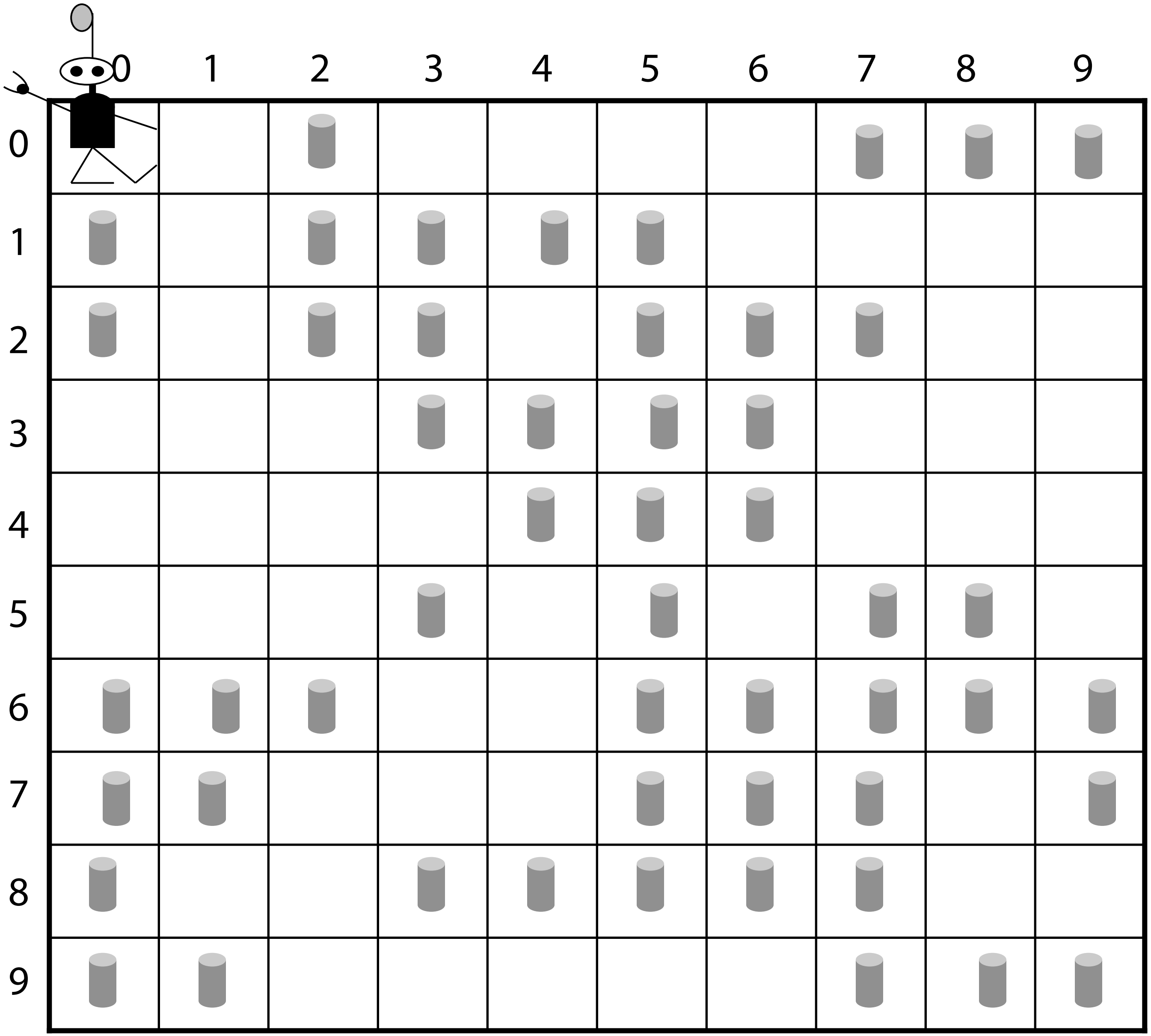
	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 1 Score: 0



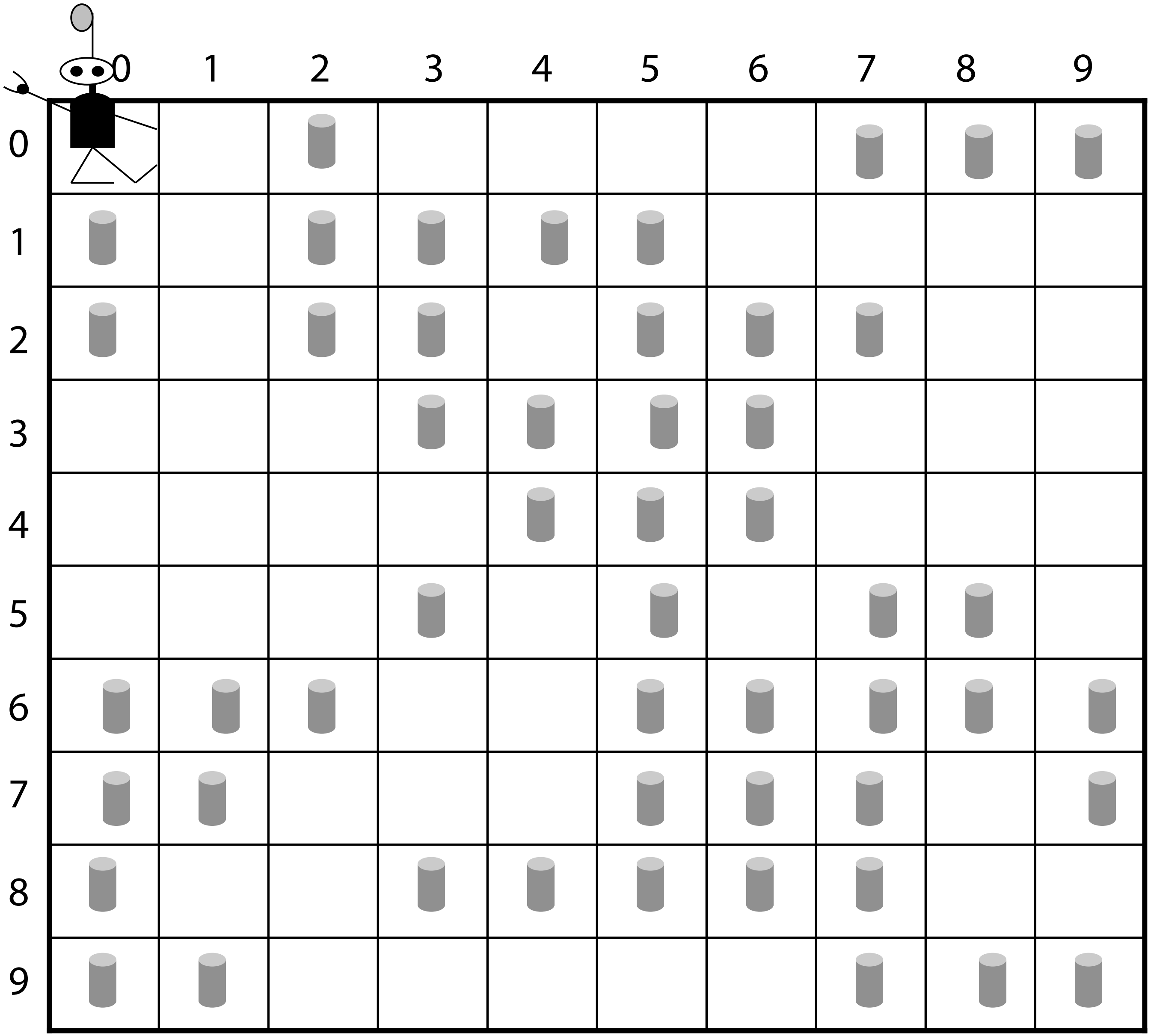
	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 2 Score: 0



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

Time: 3 Score: 0

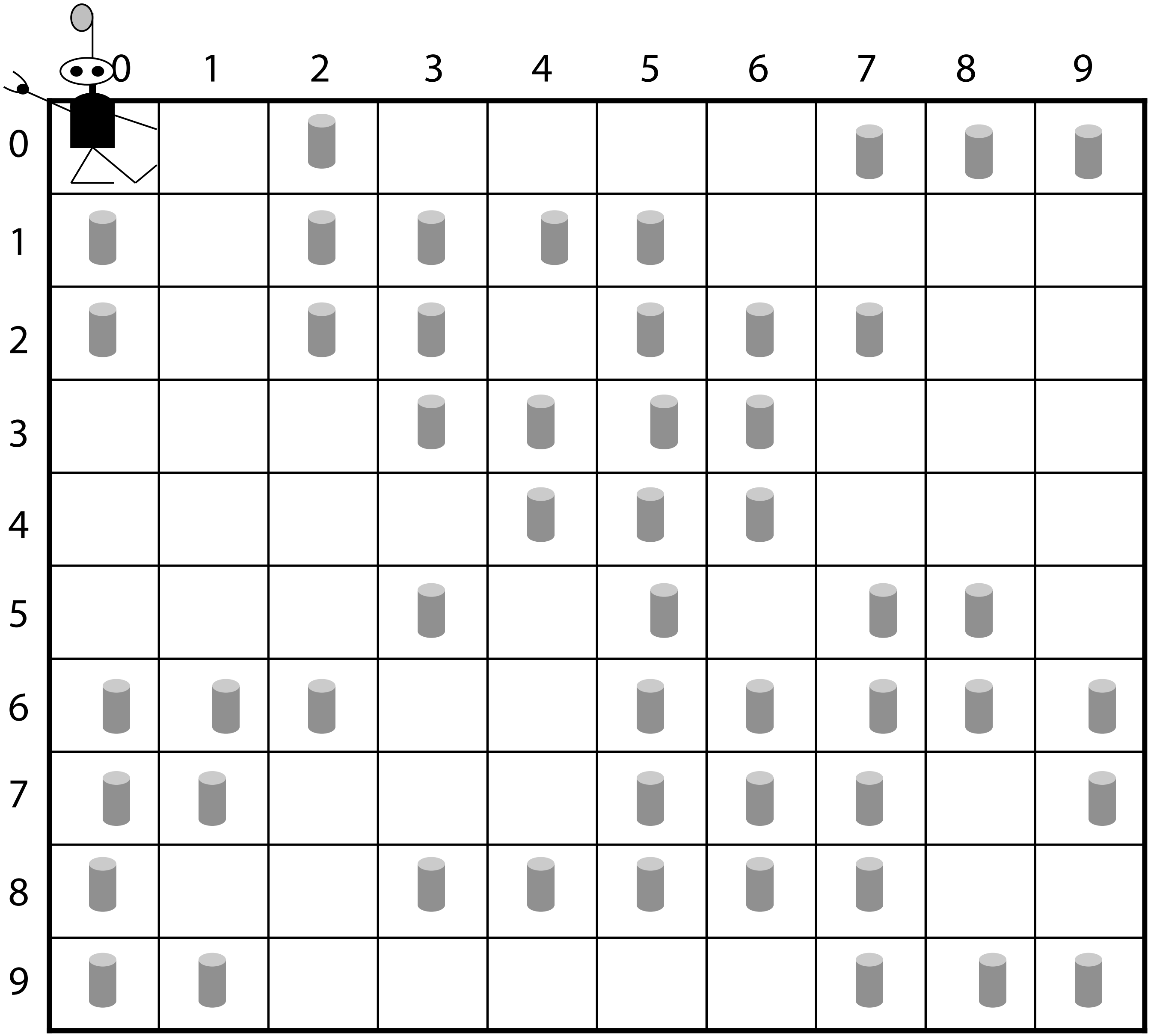


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Generation 200

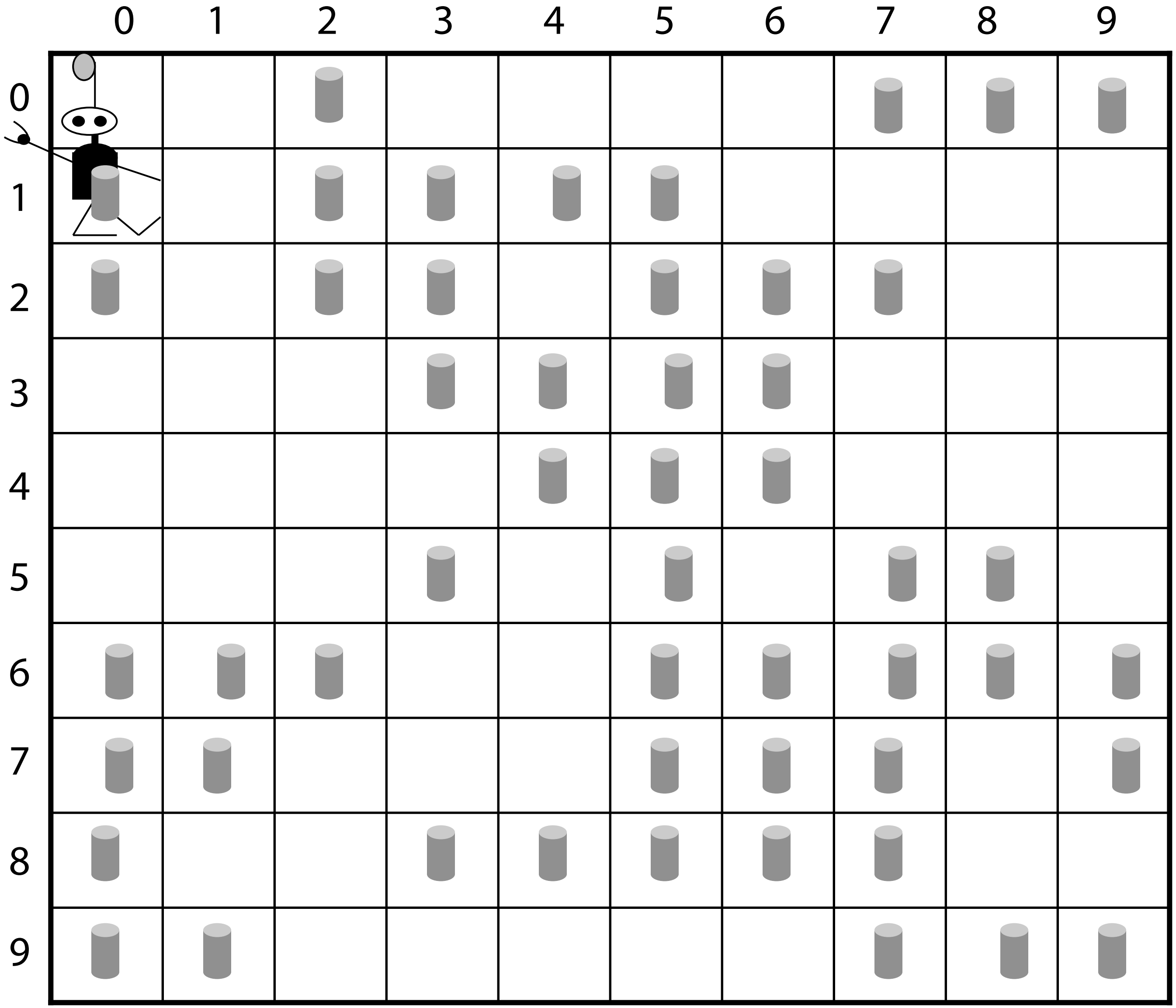
Fitness = 240

Time: 0   Score: 0

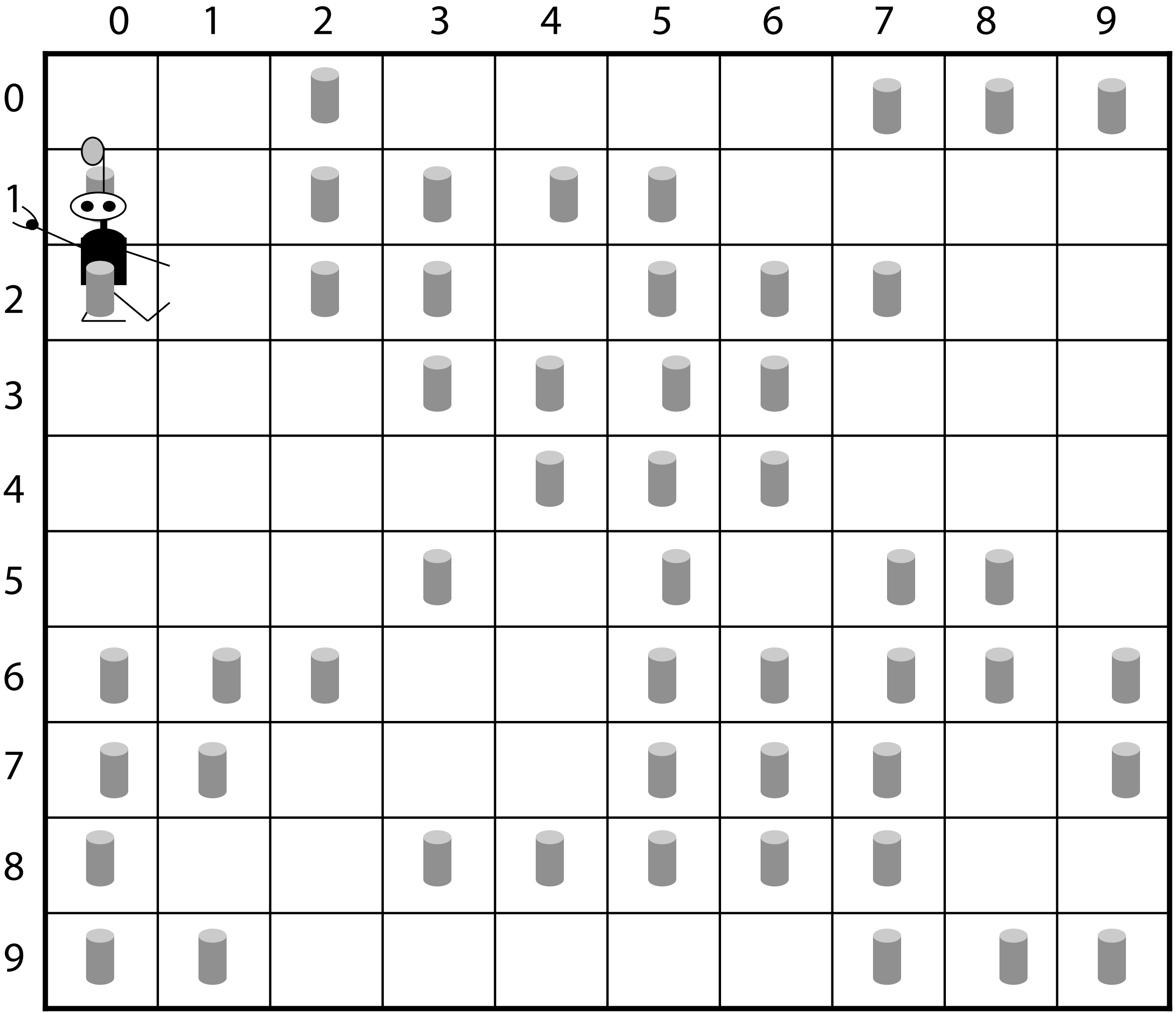


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

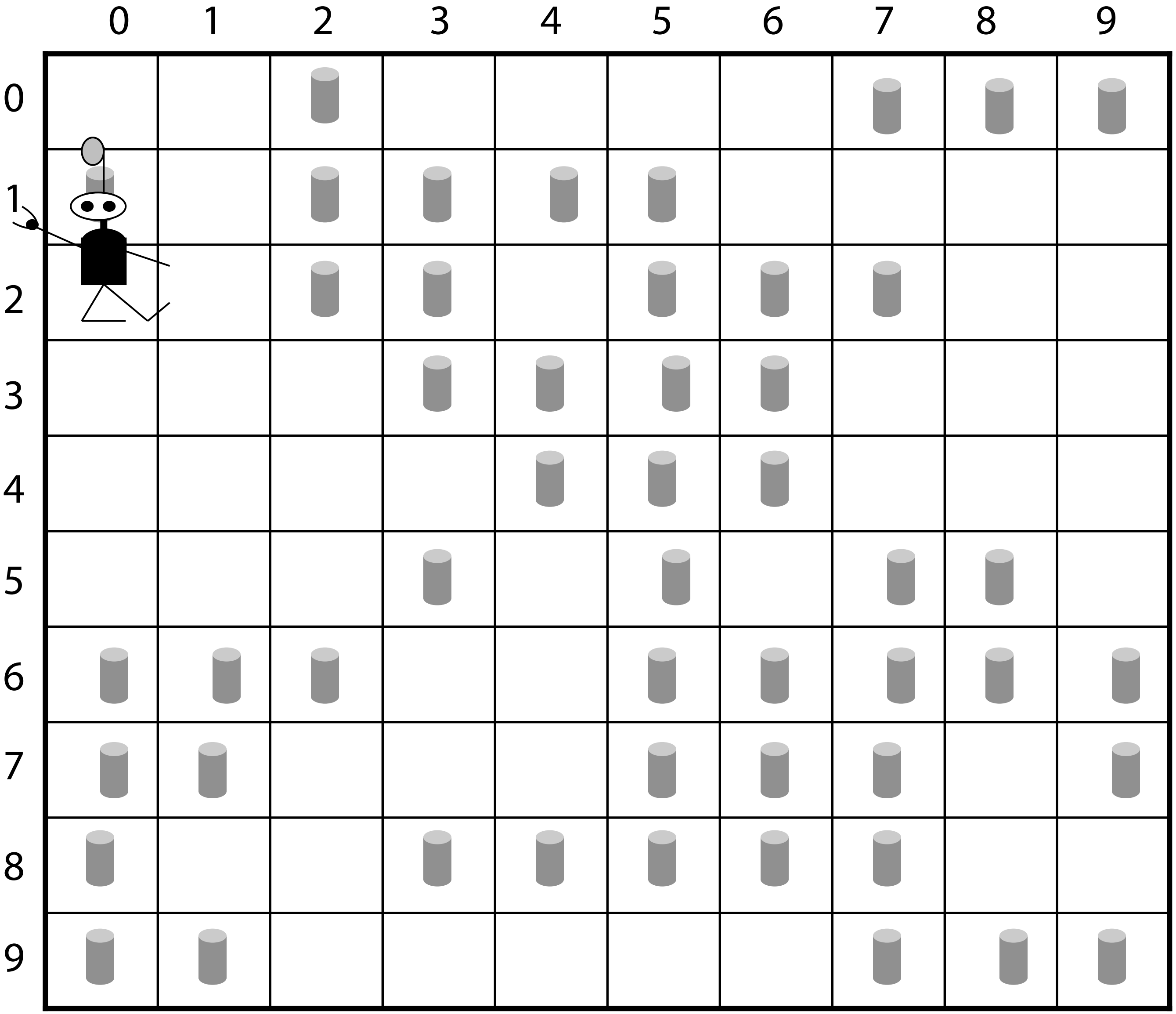
Time: 1   Score: 0



Time: 2 **Score: 0**

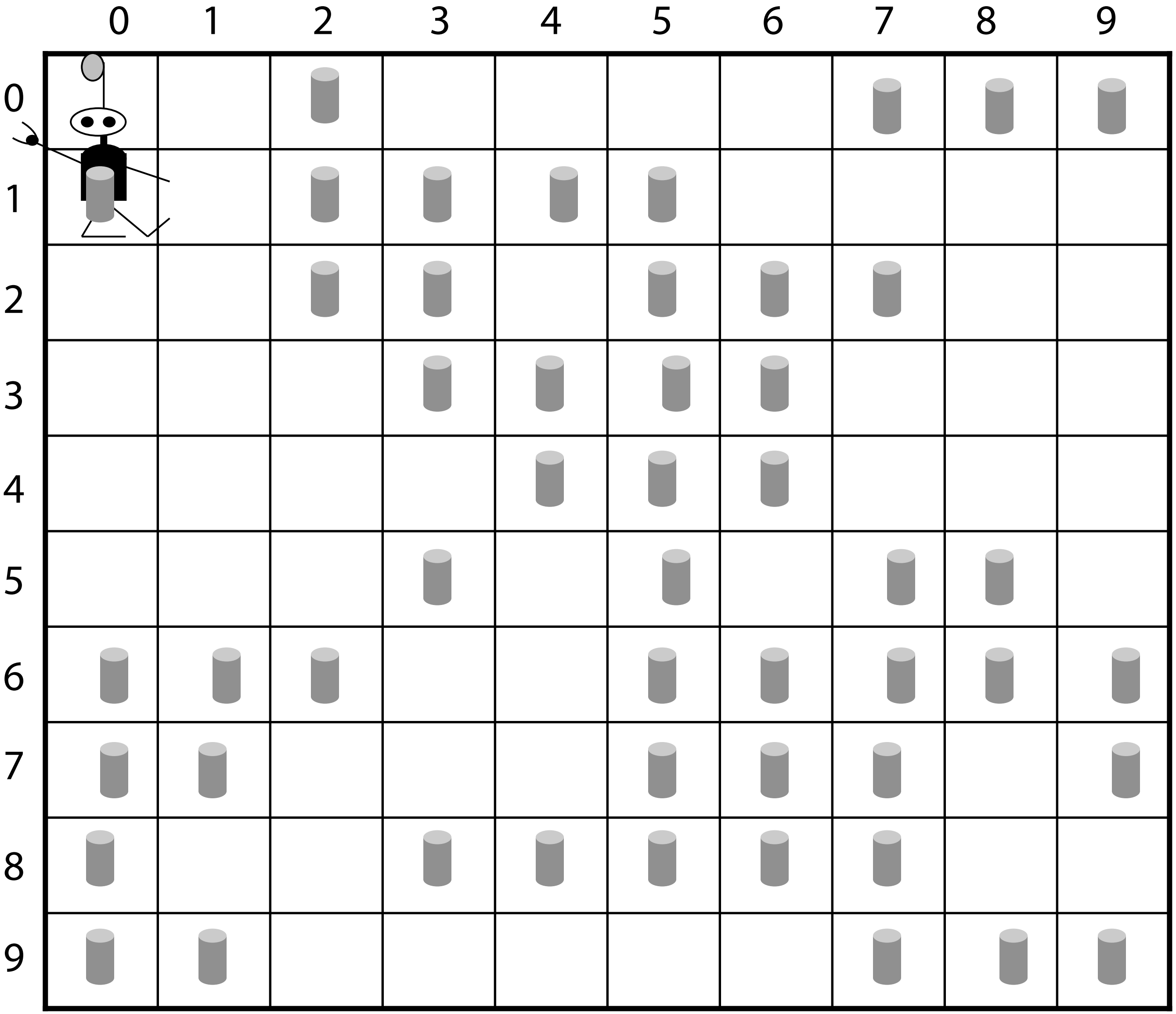


Time: 3   Score: 10

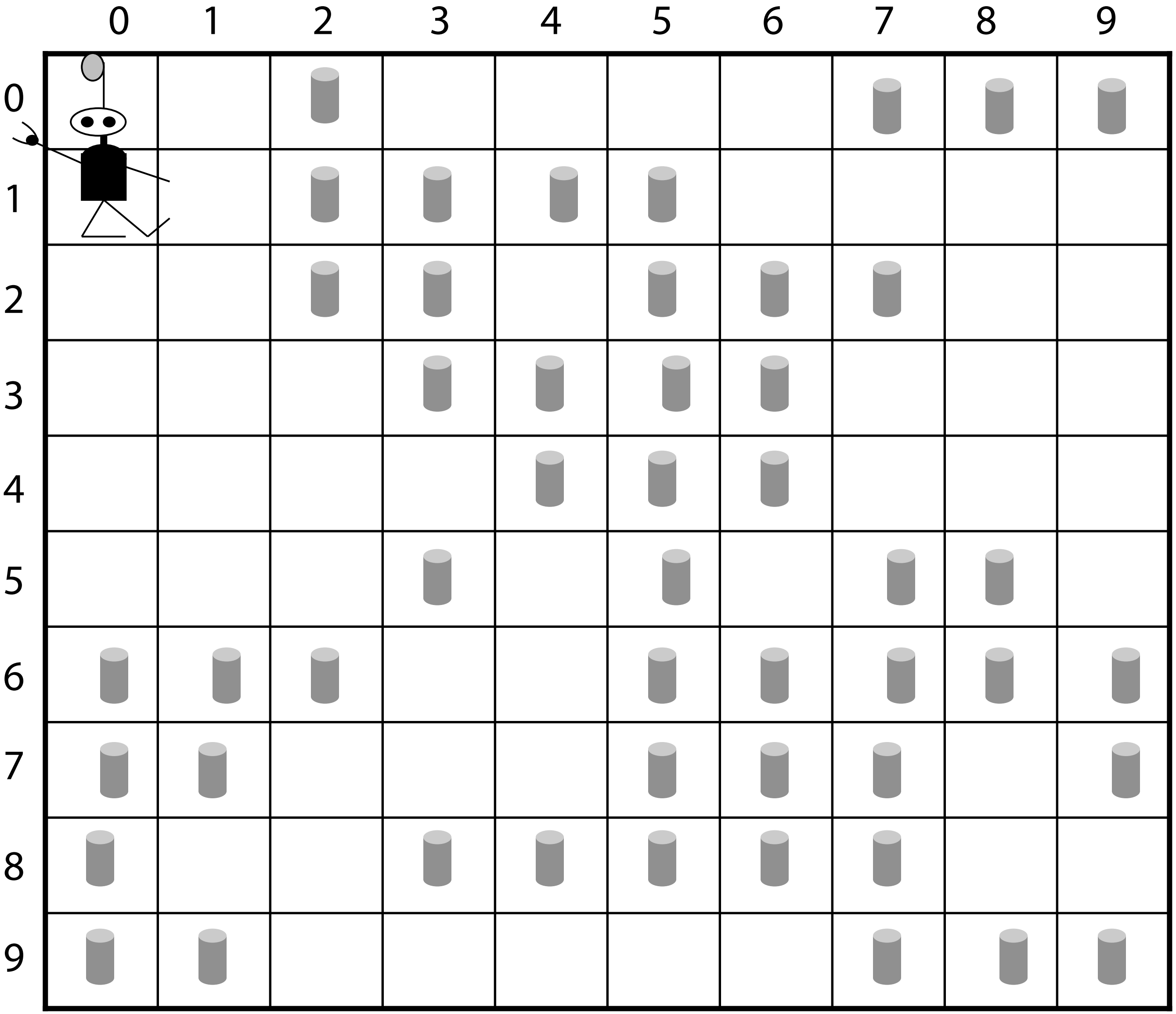


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

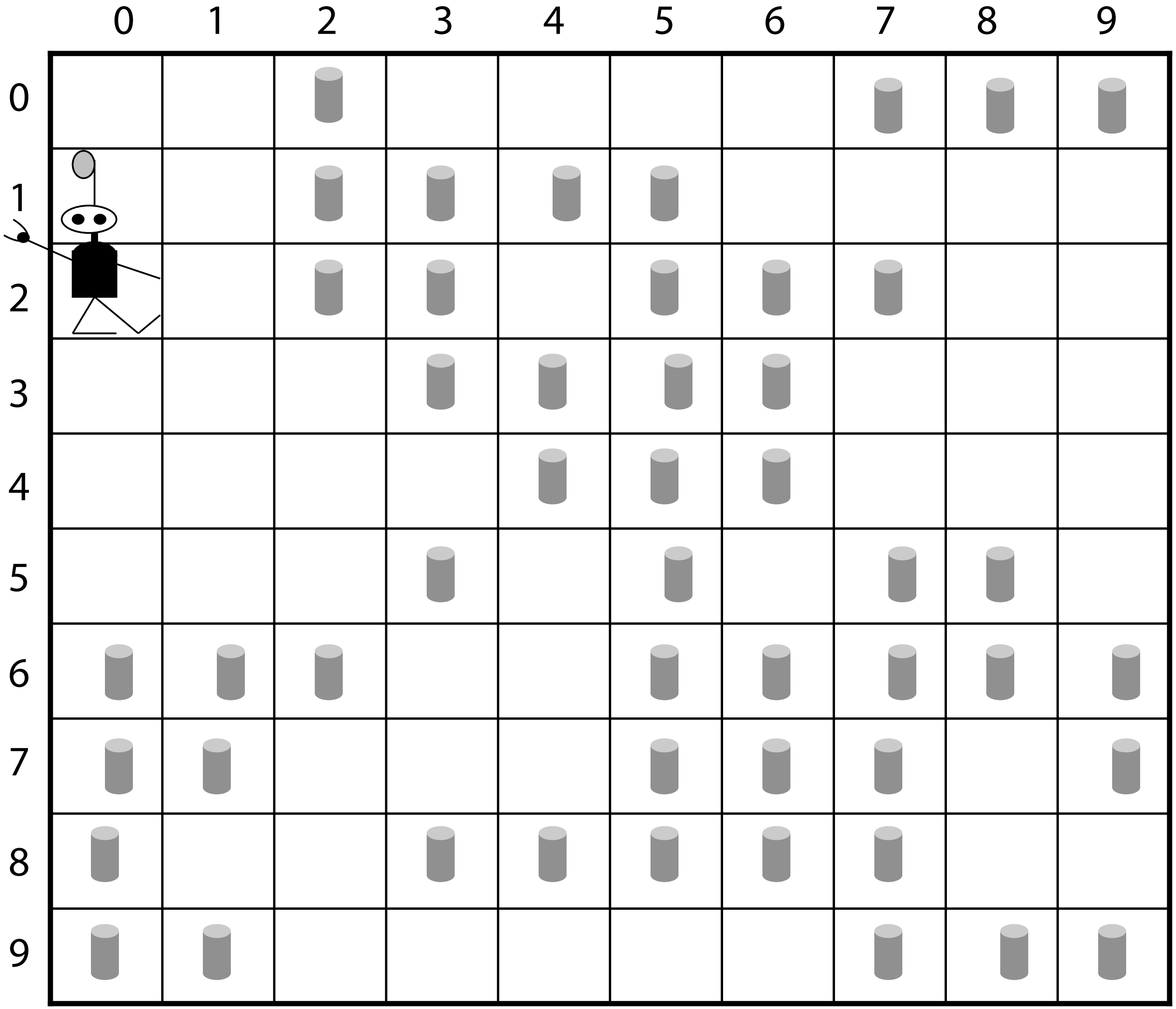
Time: 4 Score: 10



Time: 5 Score: 20

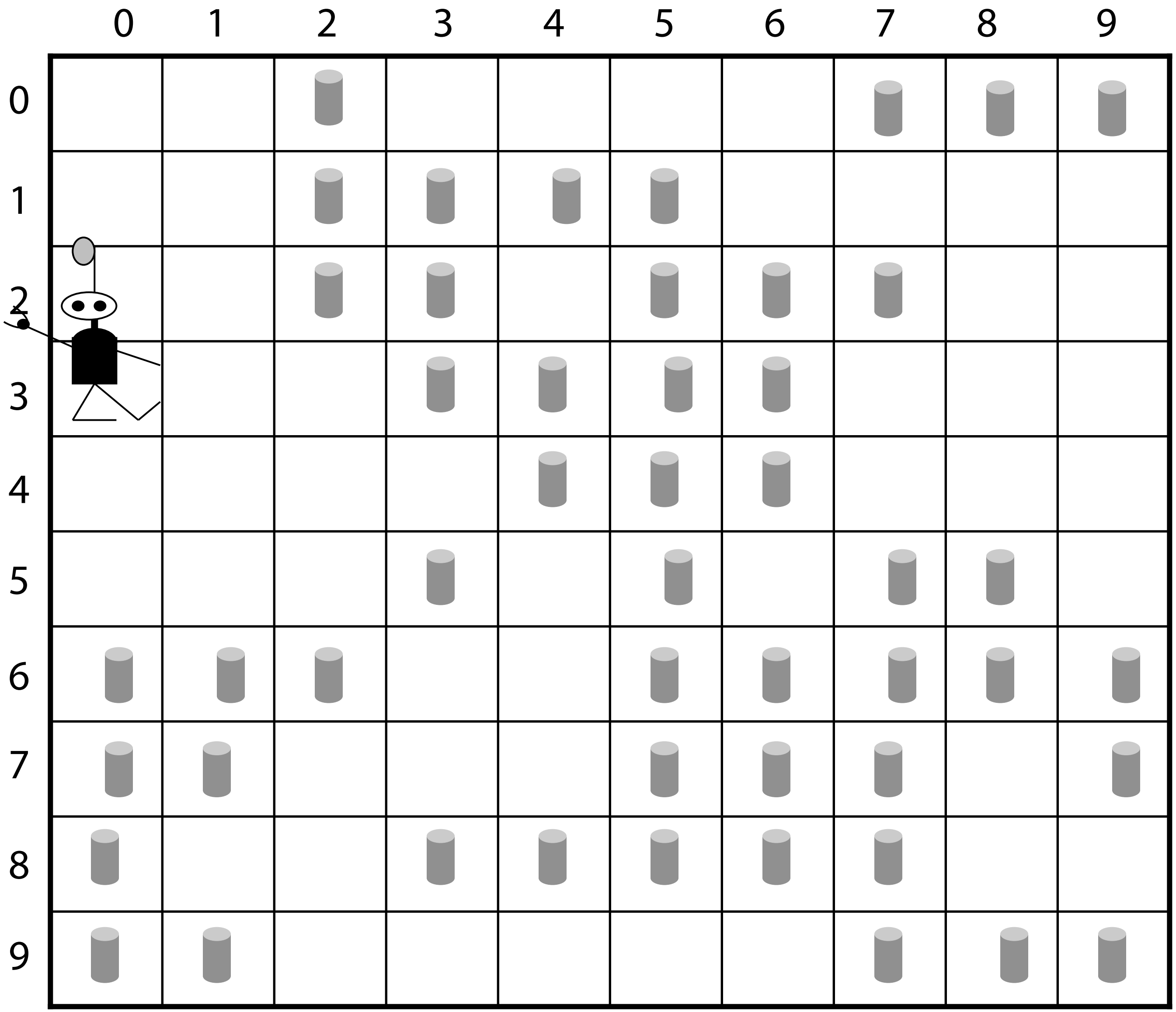


Time: 6   Score: 20

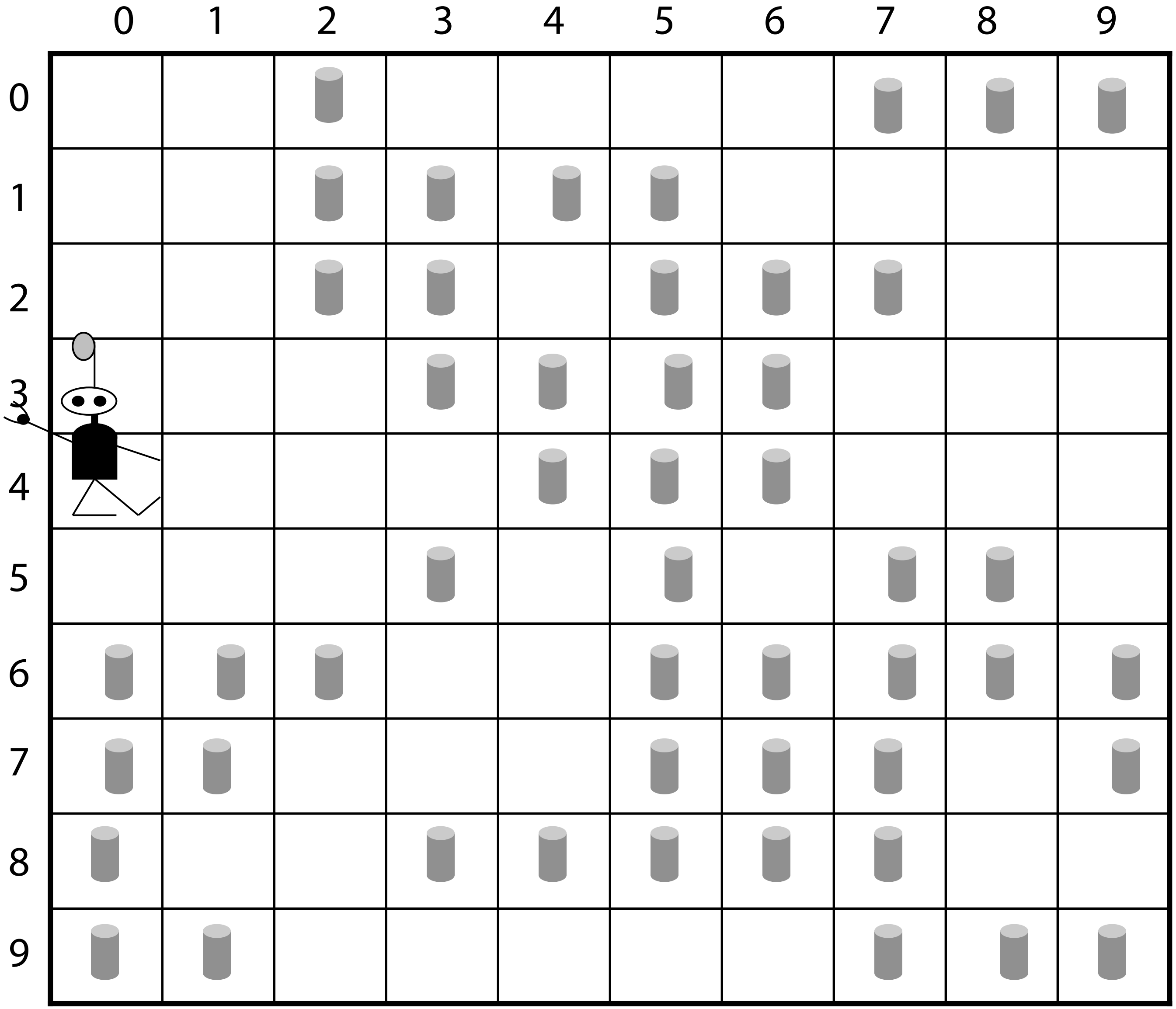


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 7 **Score: 20**

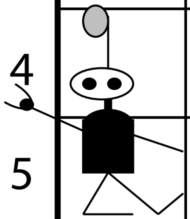


Time: 8 Score: 20

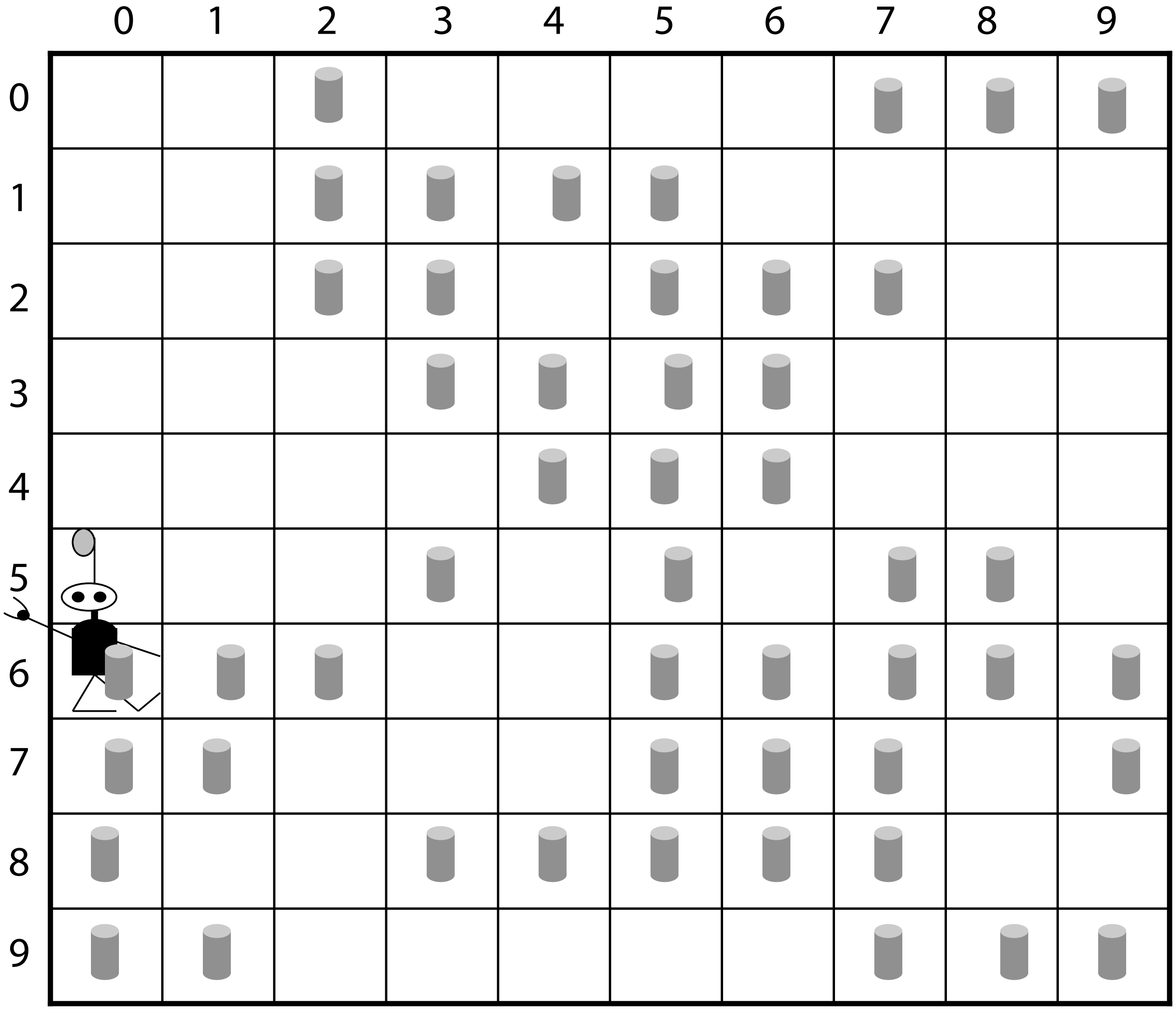


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

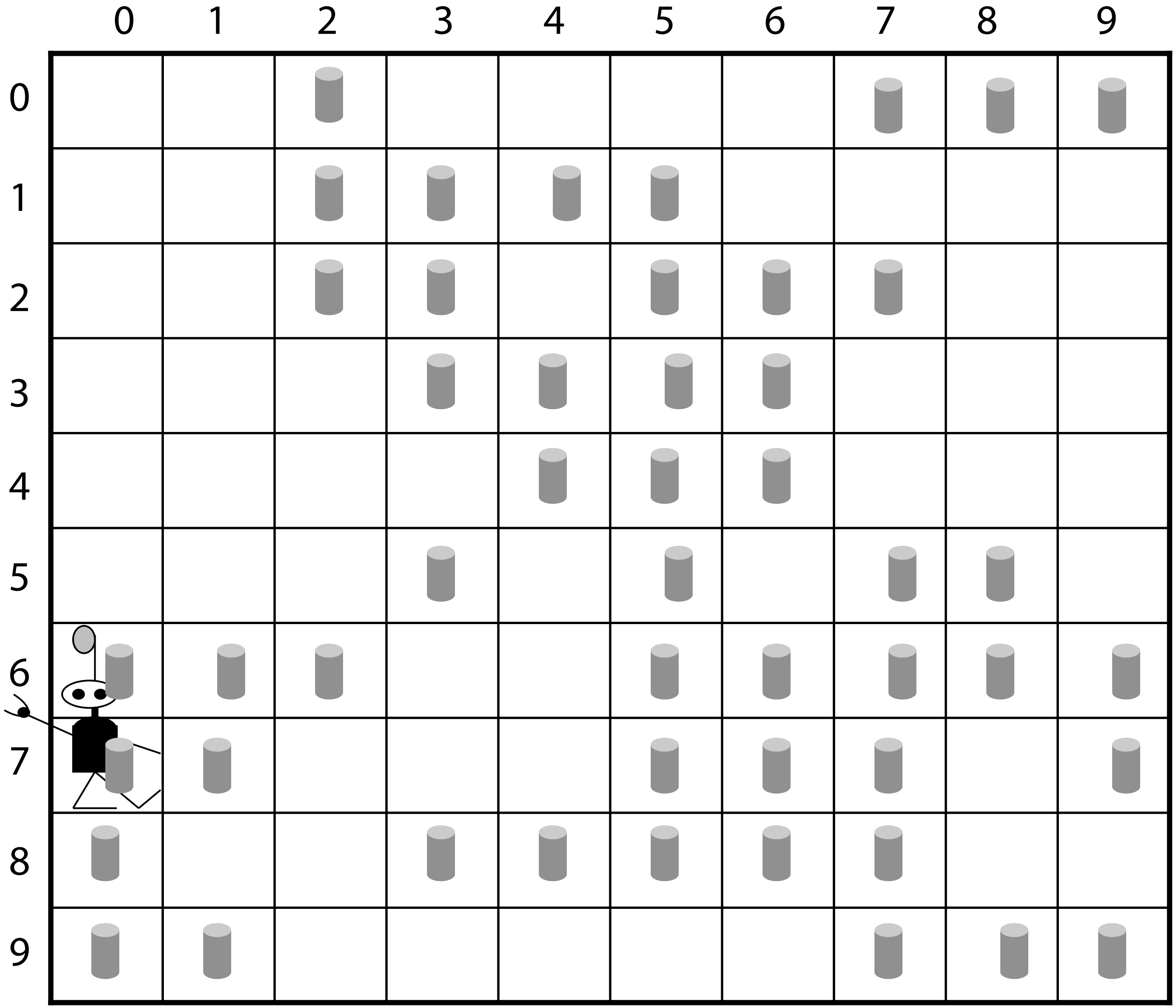
Time: 9 Score: 20

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

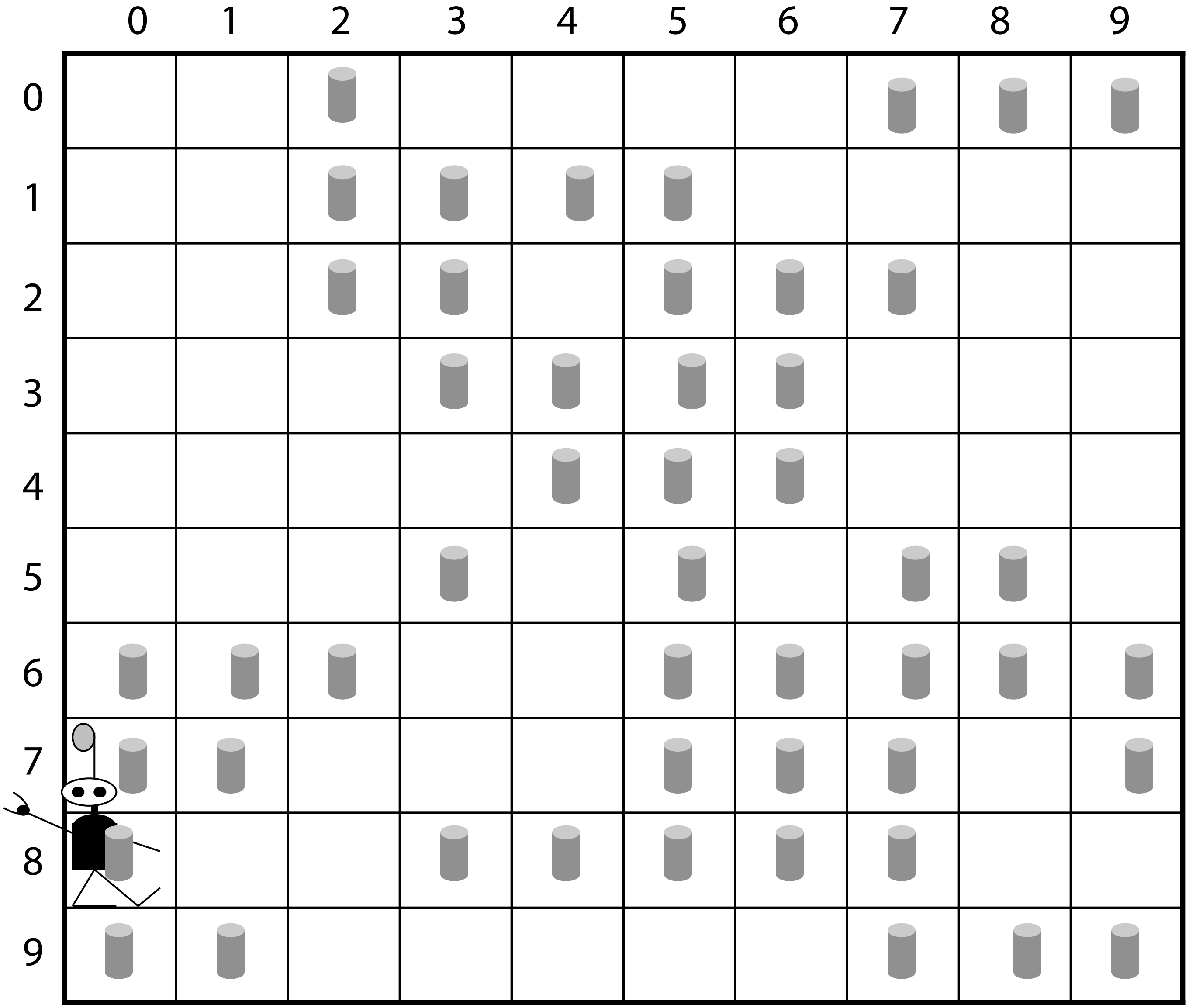
Time: 10 Score: 20



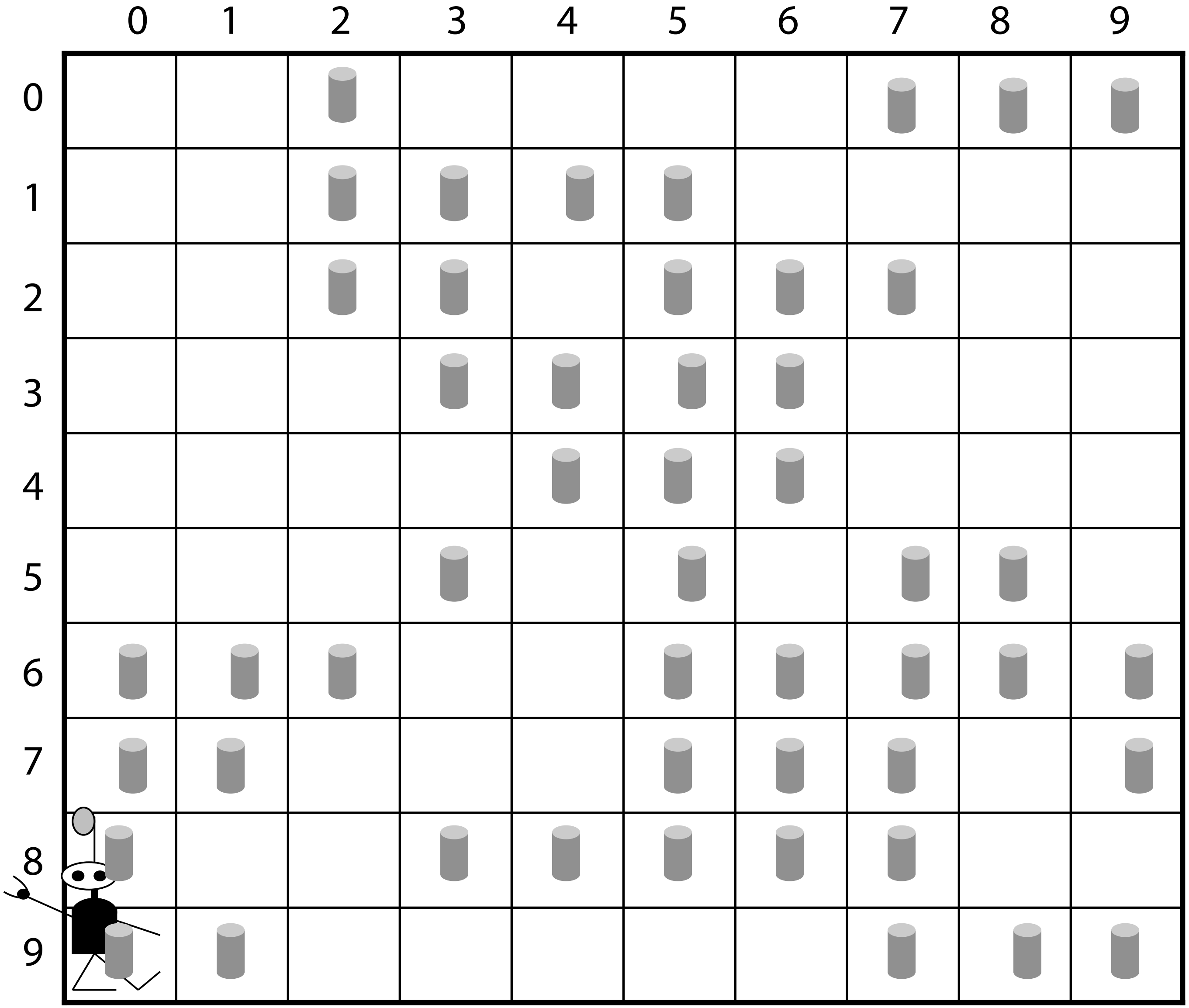
Time: 11 Score: 20



Time: 12 Score: 20

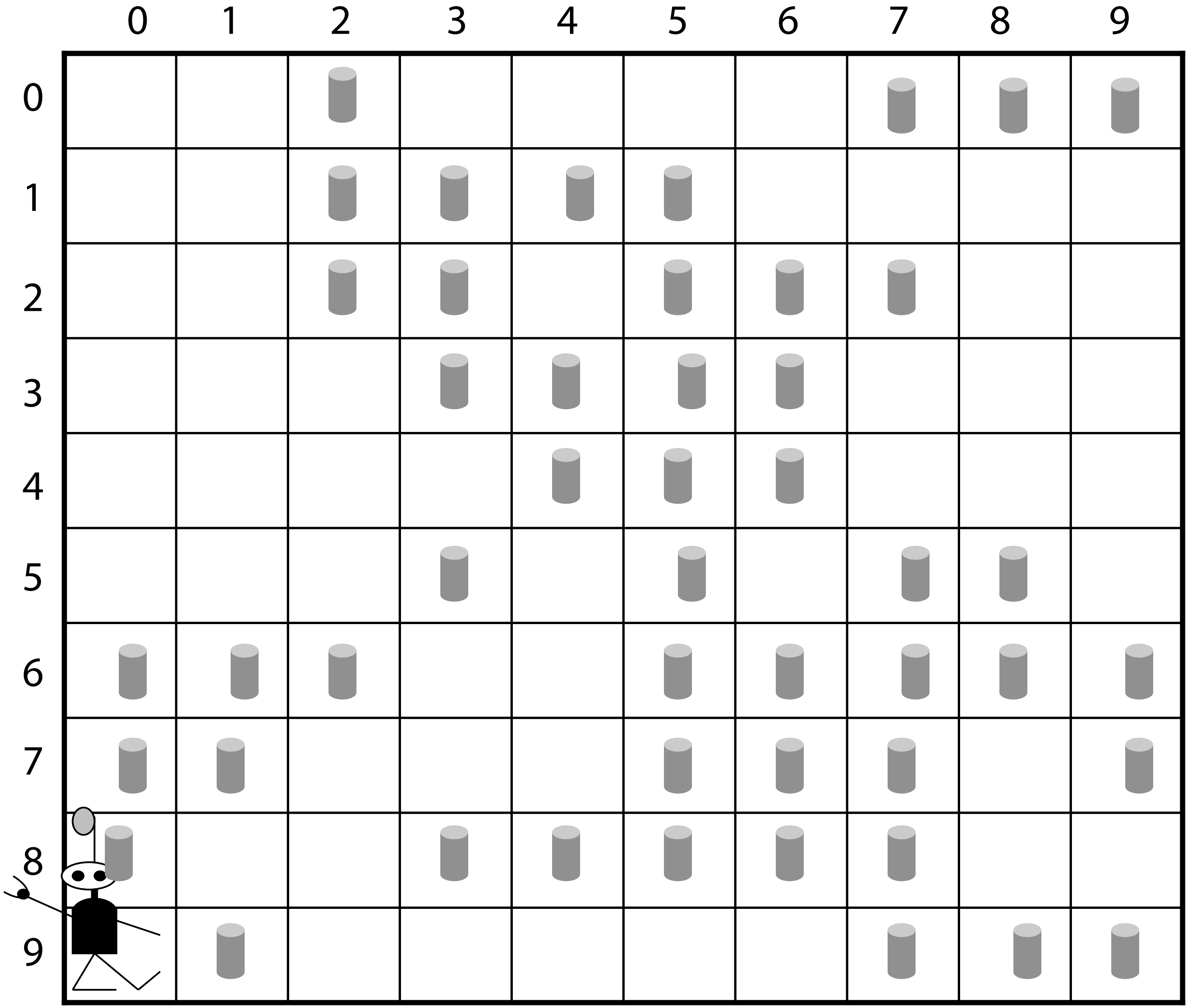


Time: 13 Score: 20



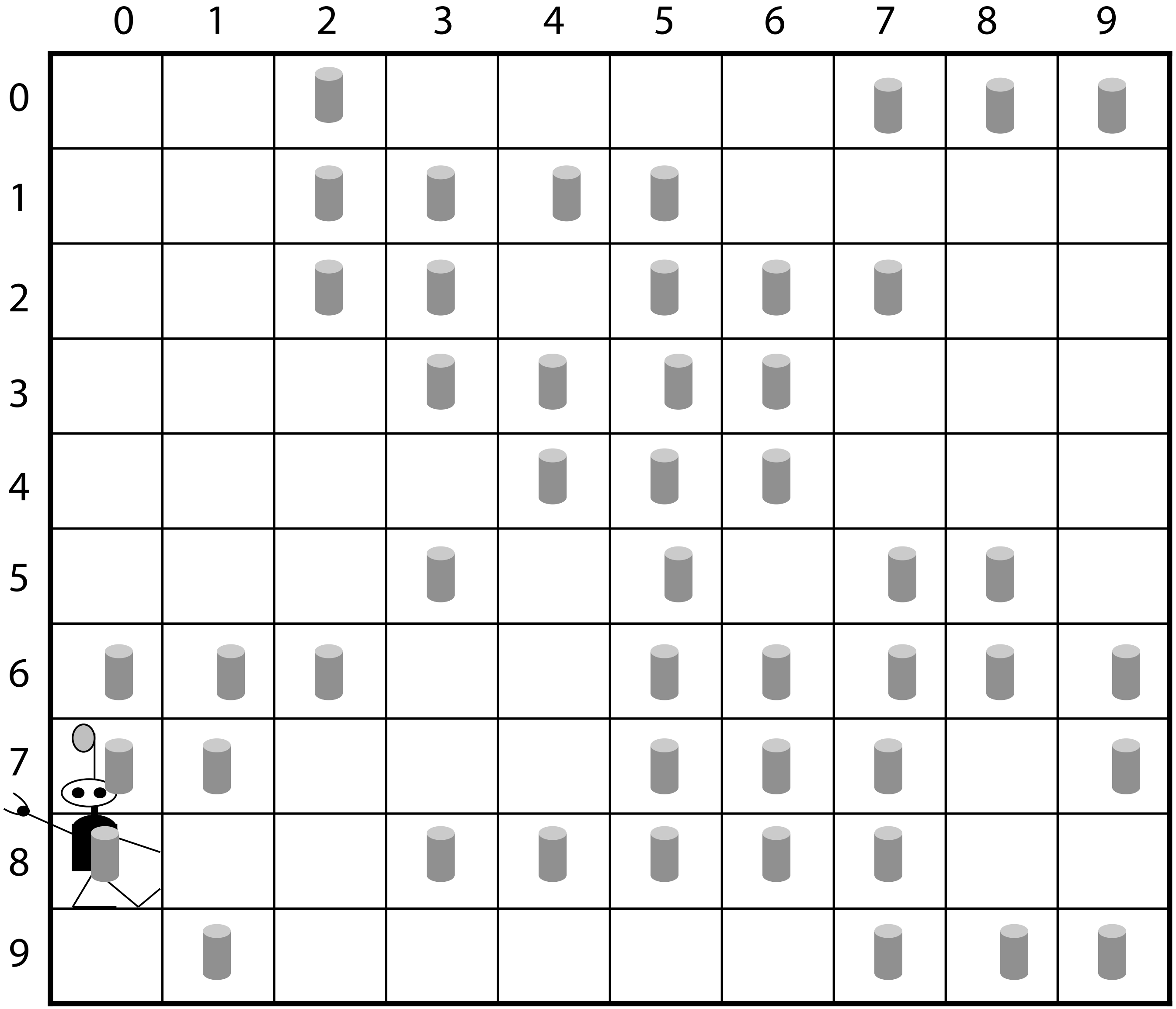
	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 14 Score: 30

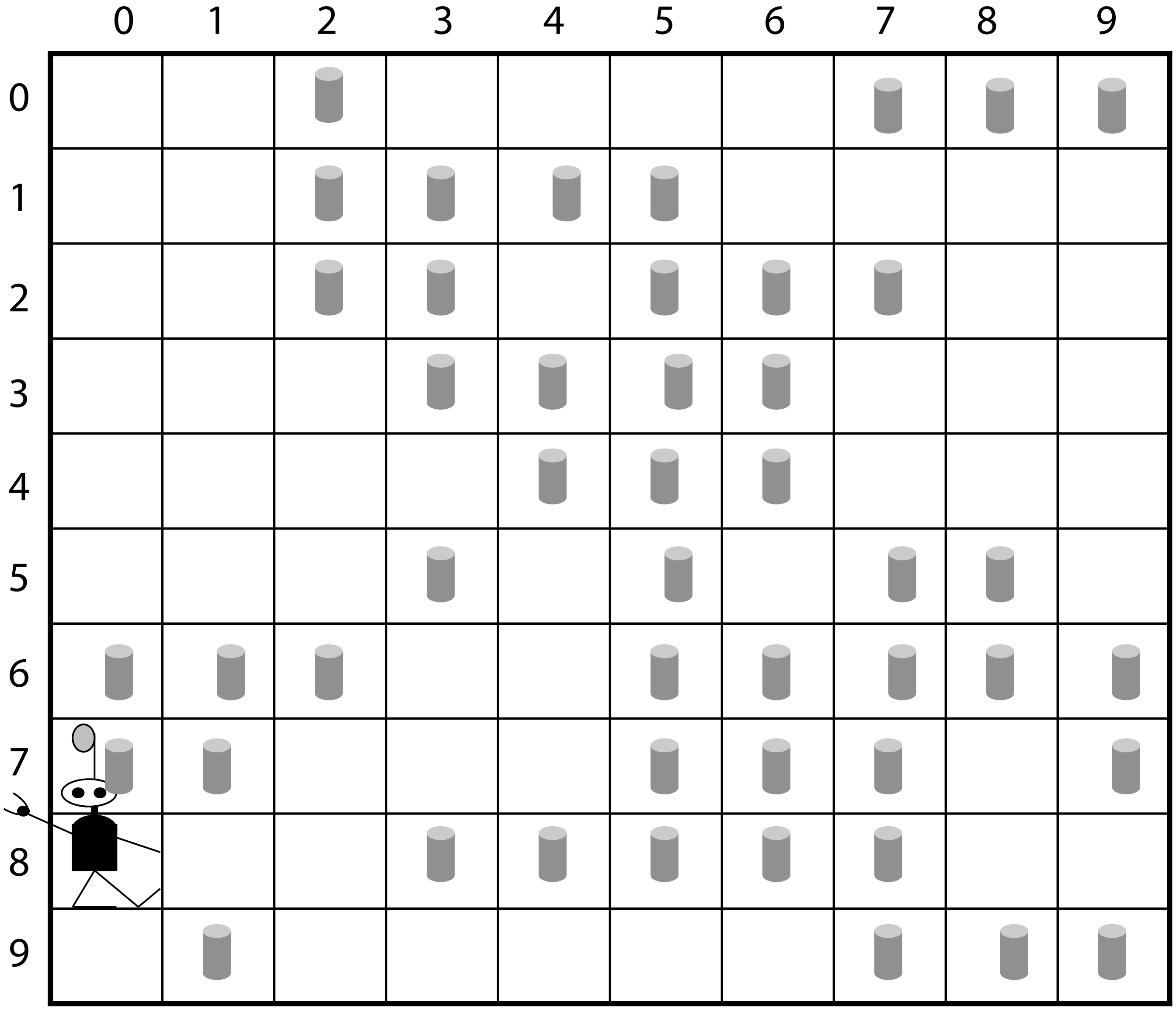


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9		●						●	●	●

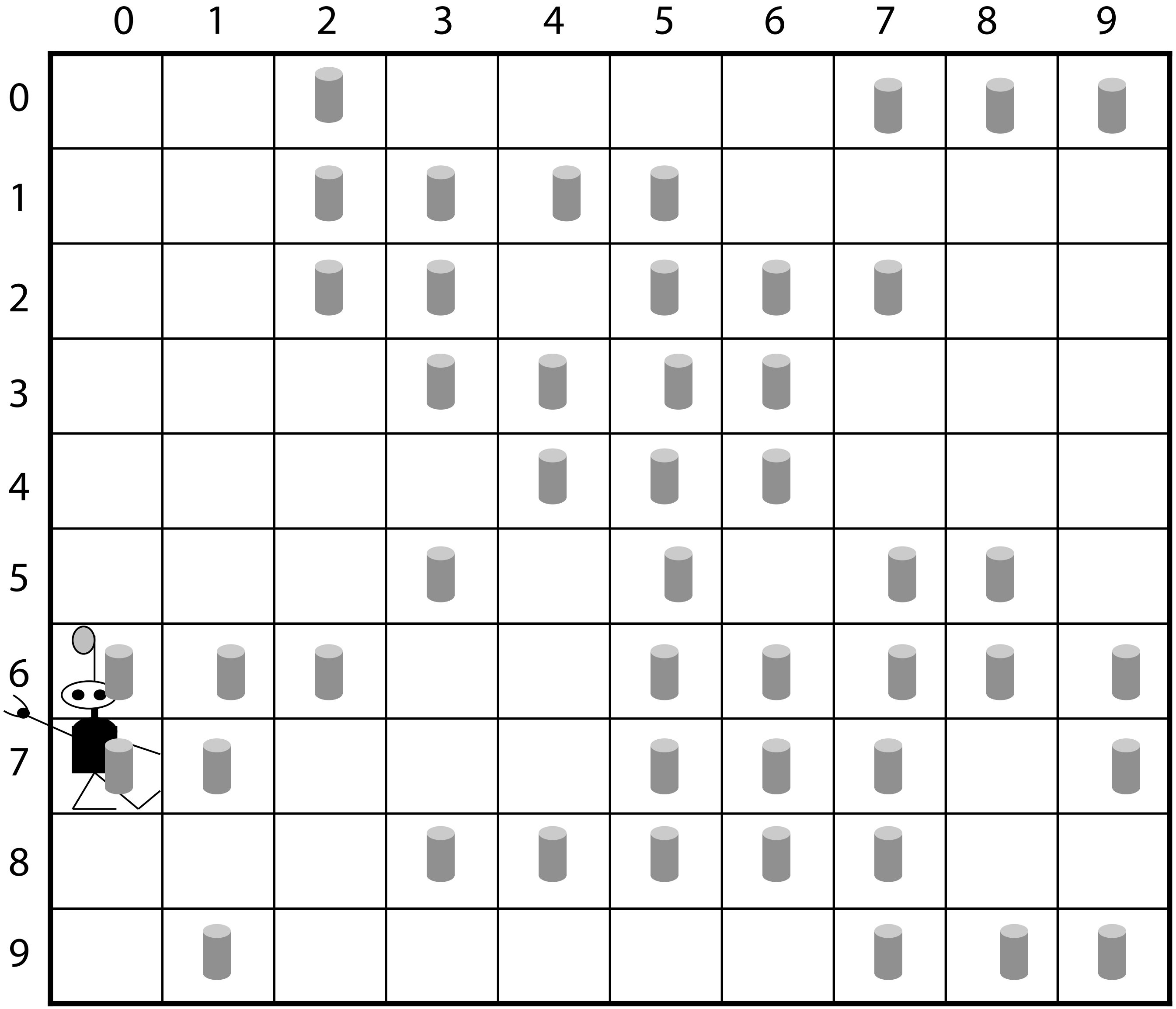
Time: 15 Score: 30



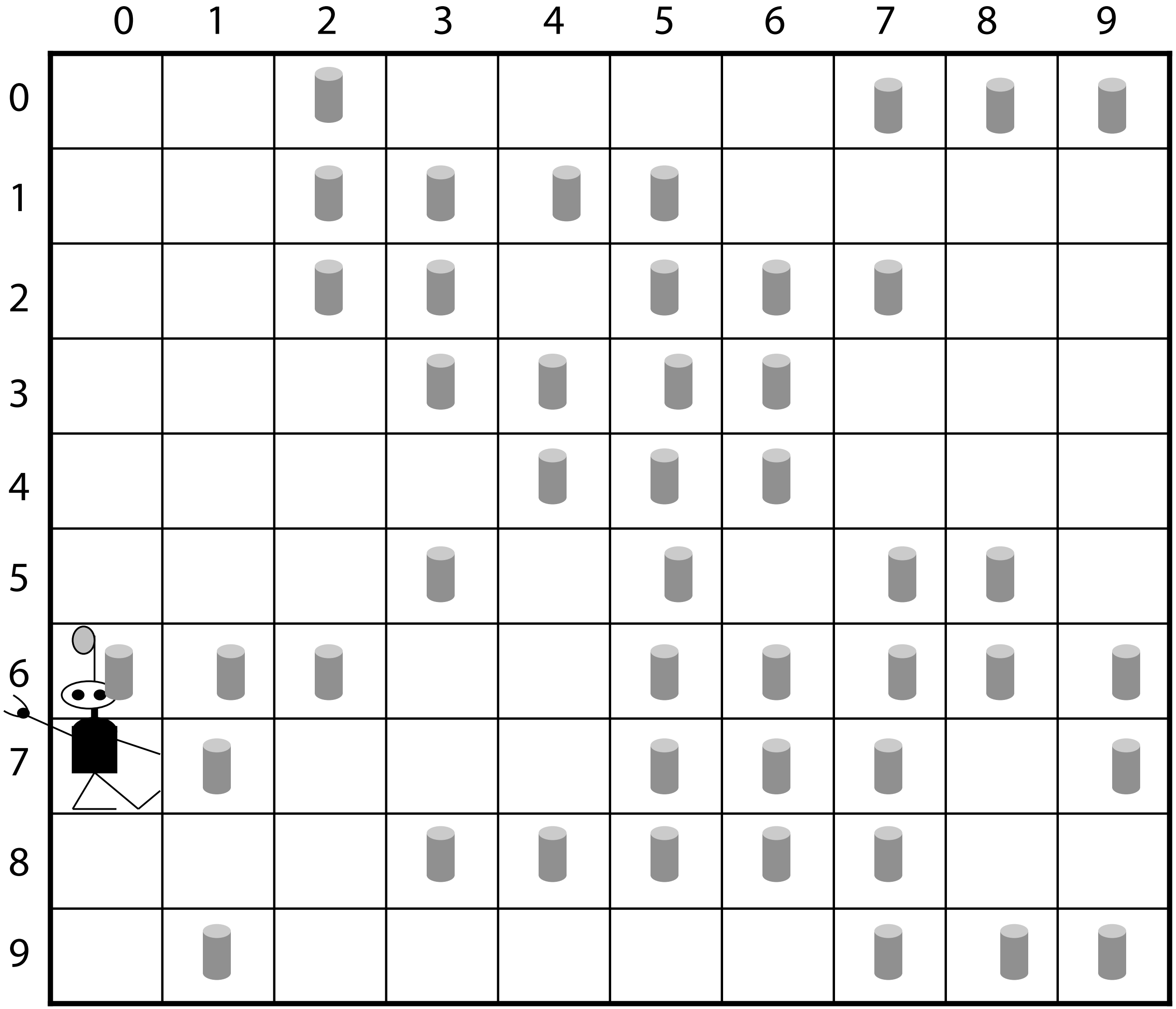
Time: 16 Score: 40



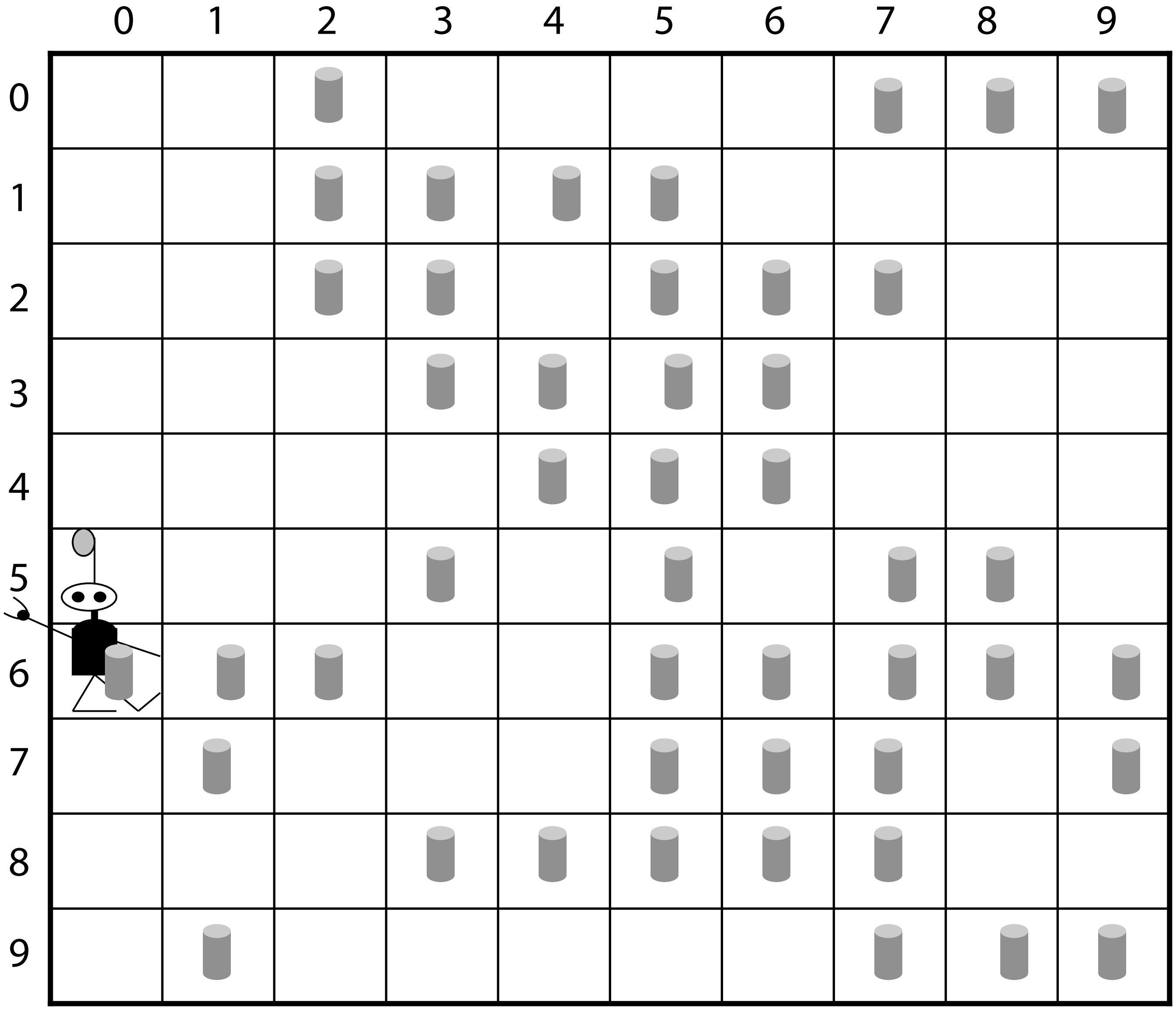
Time: 17 Score: 40



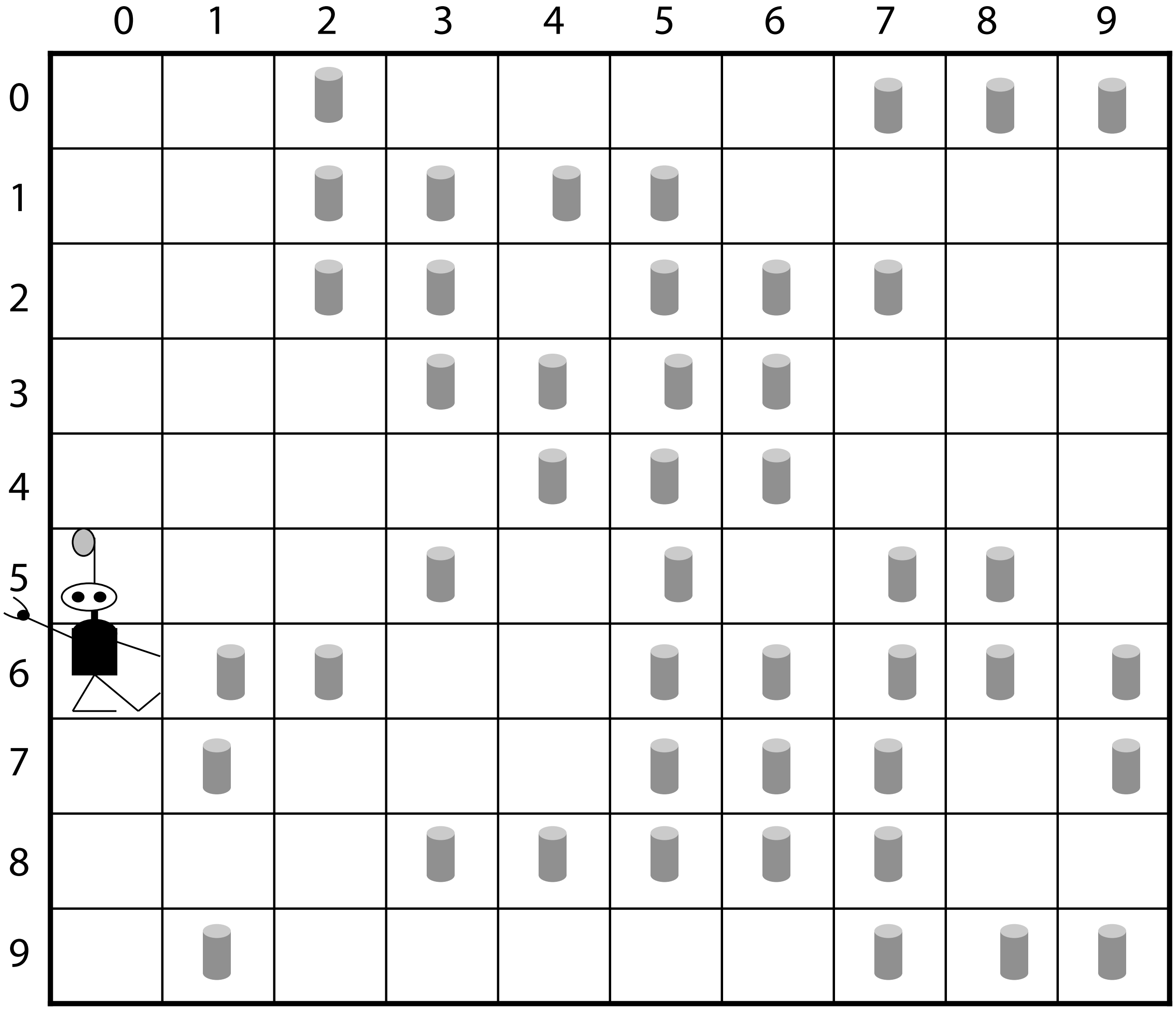
Time: 18 Score: 50



Time: 19 Score: 50



Time: 20 Score: 60

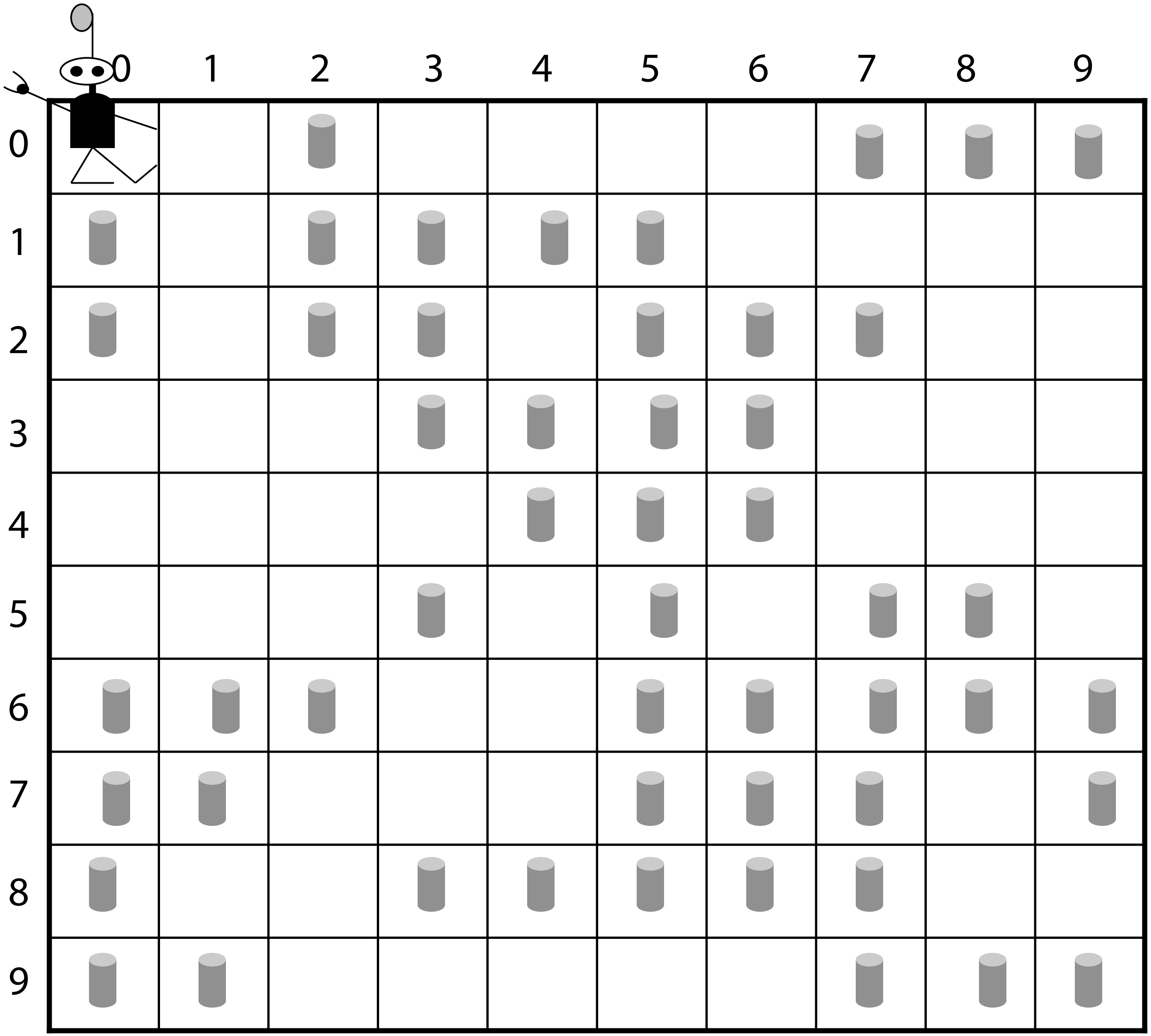


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6		●	●			●	●	●	●	●
7		●				●	●	●		●
8				●	●	●	●	●		
9		●						●	●	●

Generation 1000

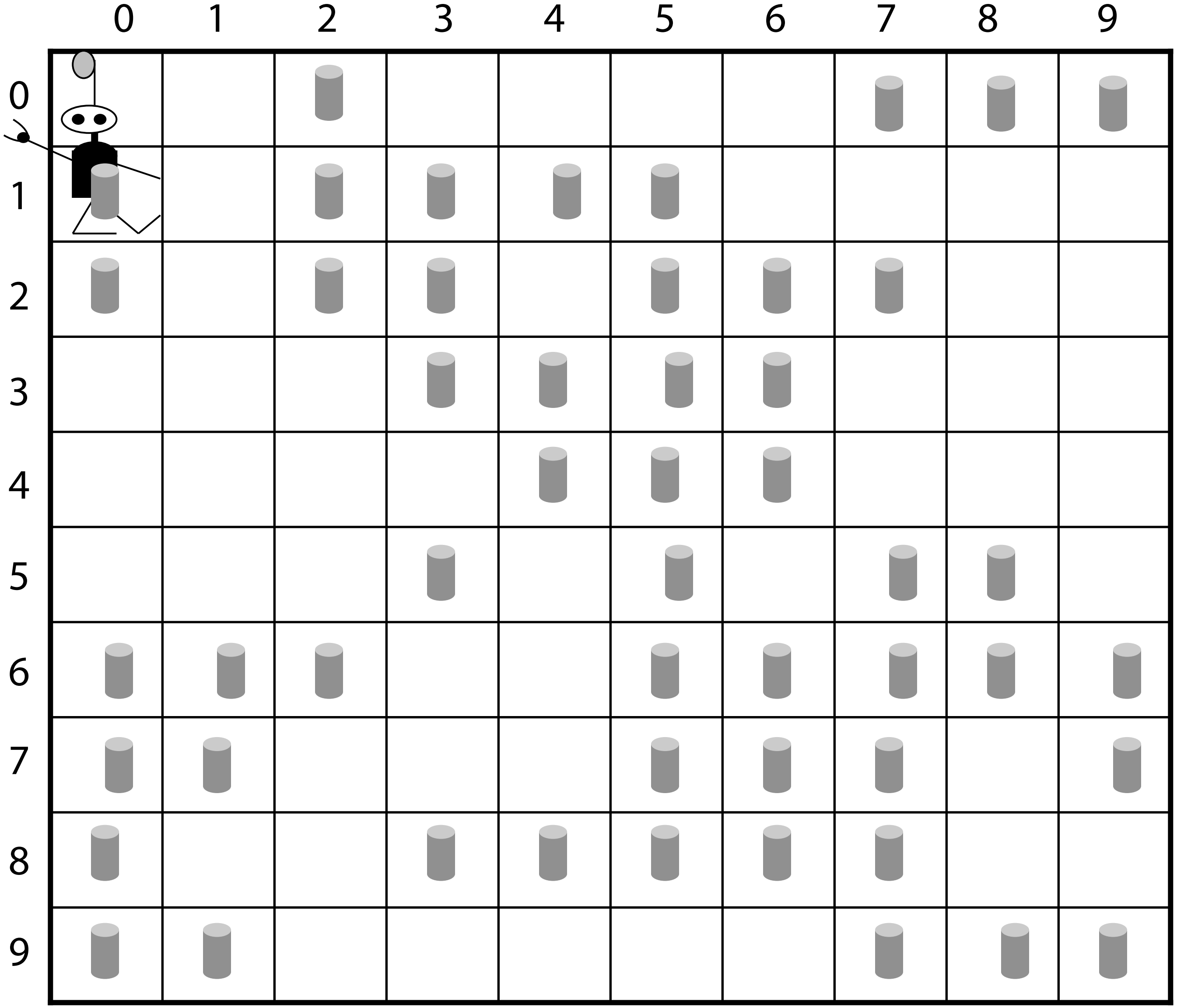
Fitness = 492

Time: 0 Score: 0

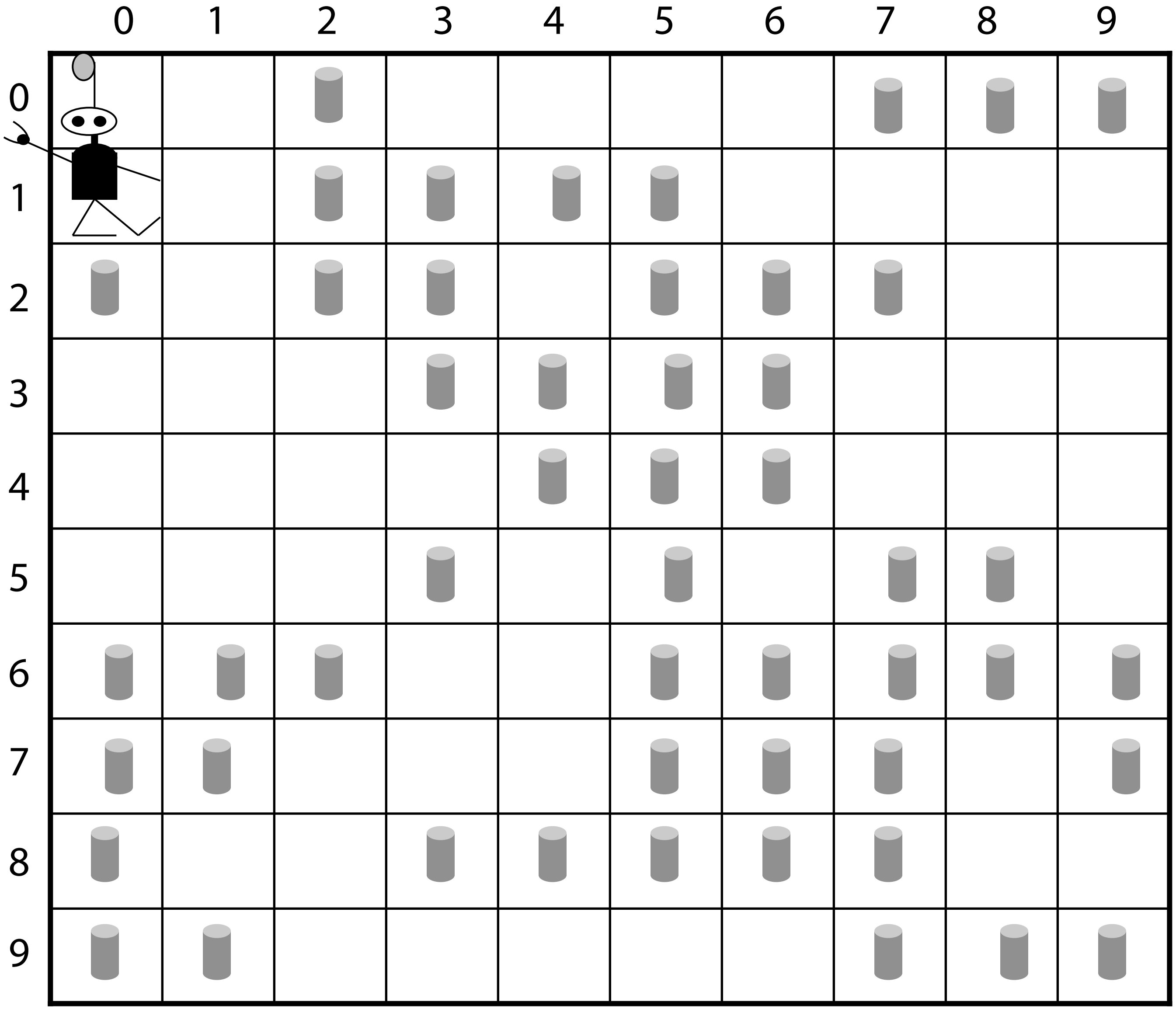


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1	●		●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

Time: 1 Score: 0

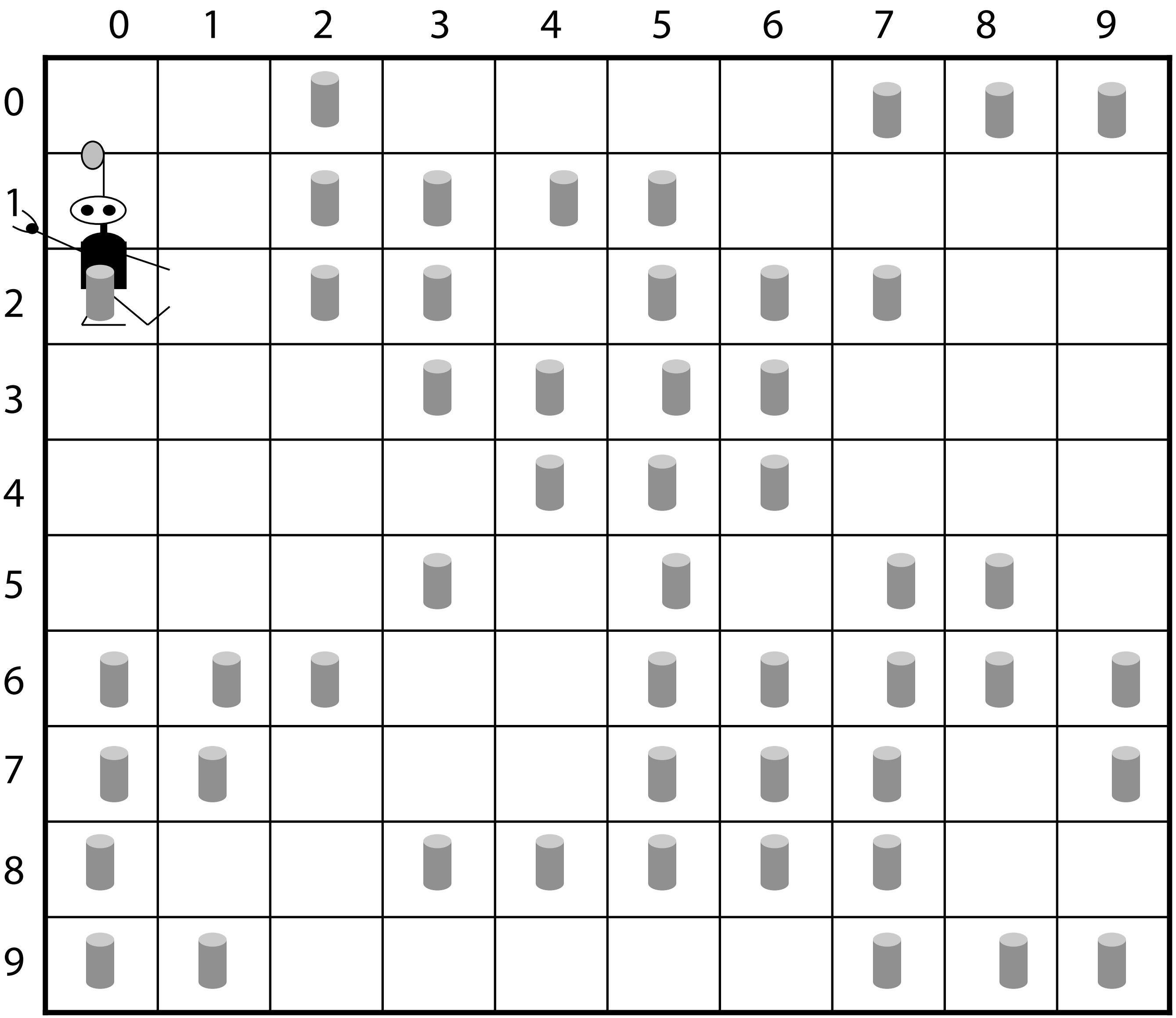


Time: 2   Score: 10

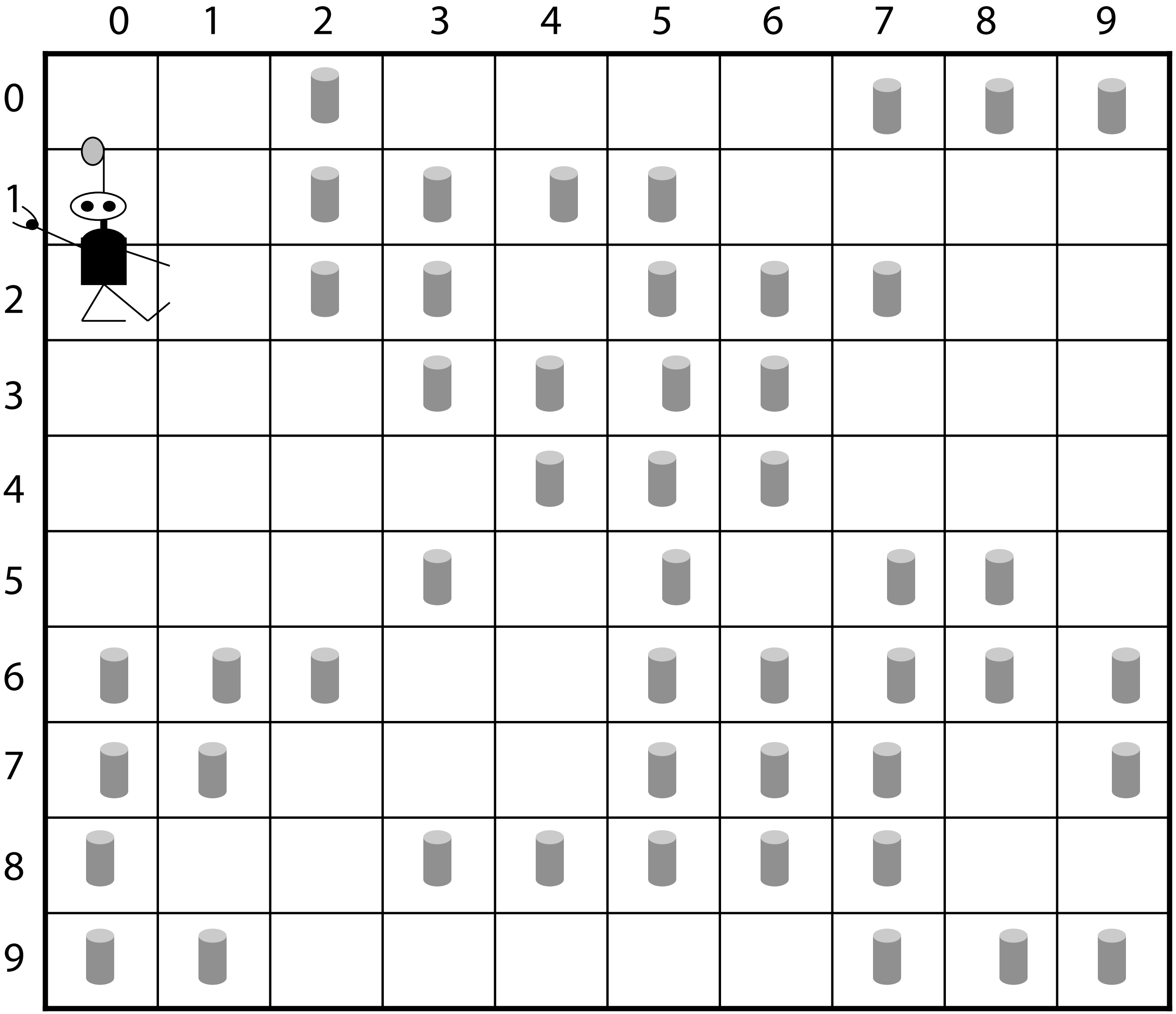


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2	●		●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

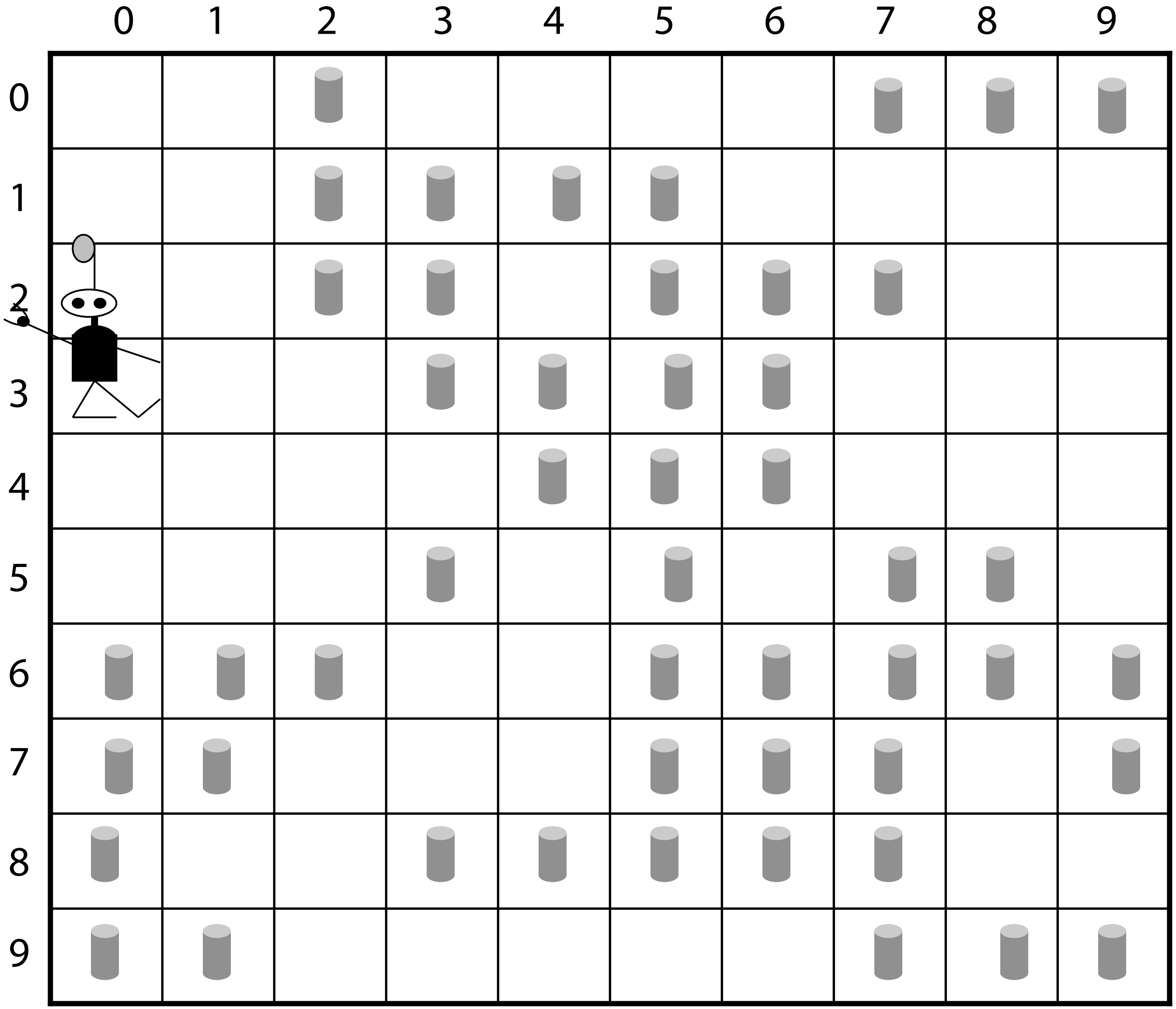
Time: 3 Score: 10



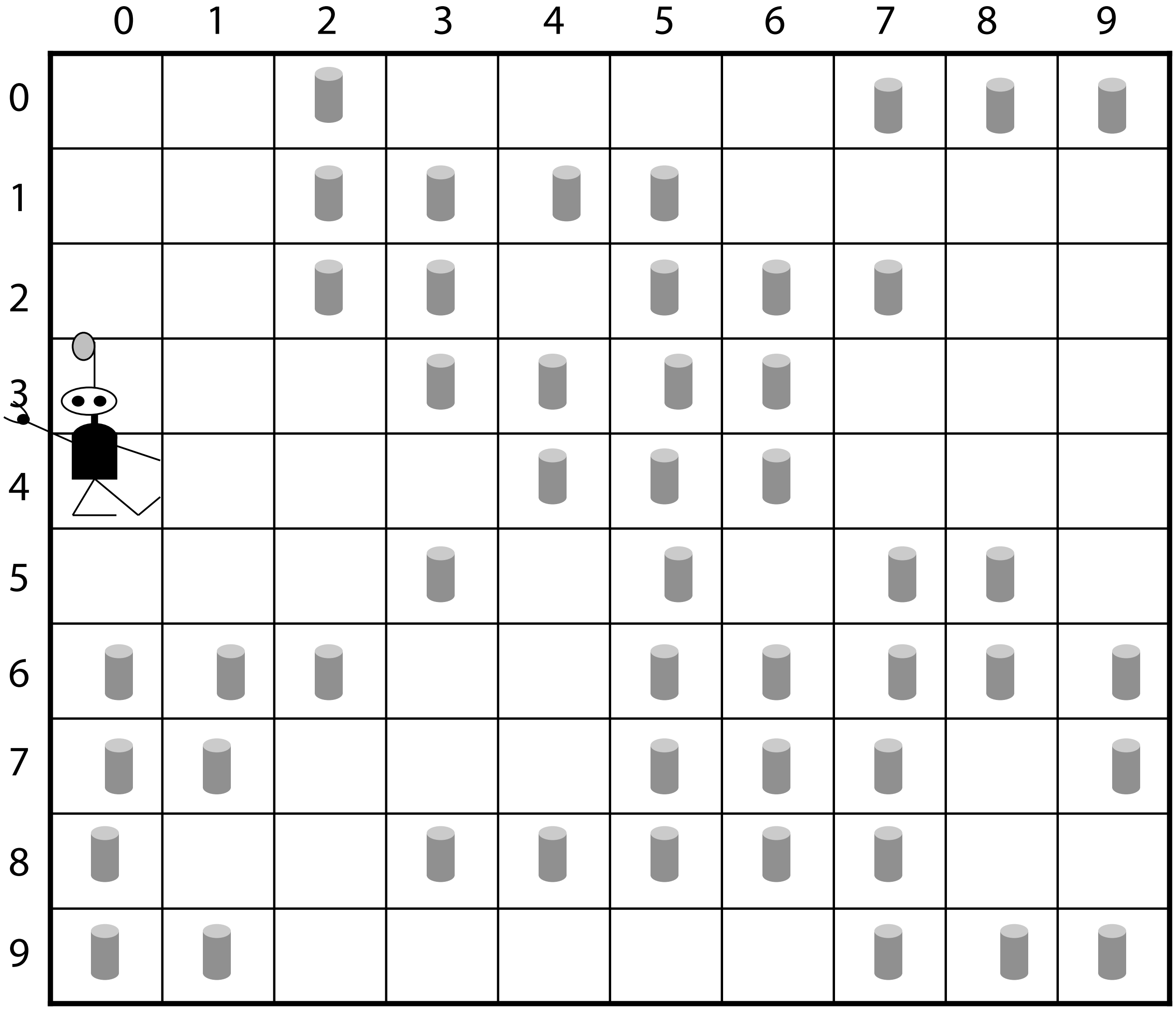
Time: 4 Score: 20



Time: 5 Score: 20

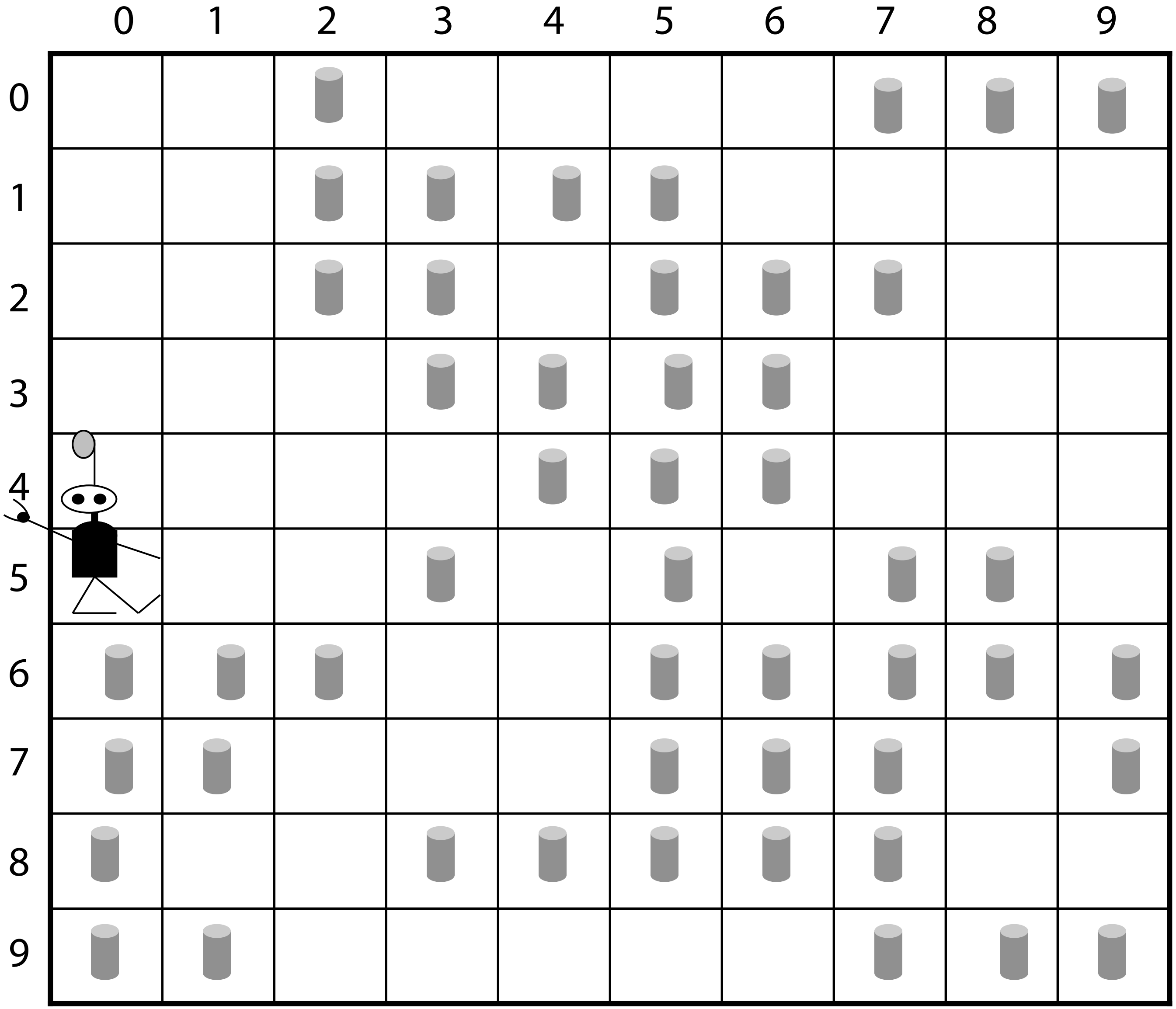


Time: 6 Score: 20

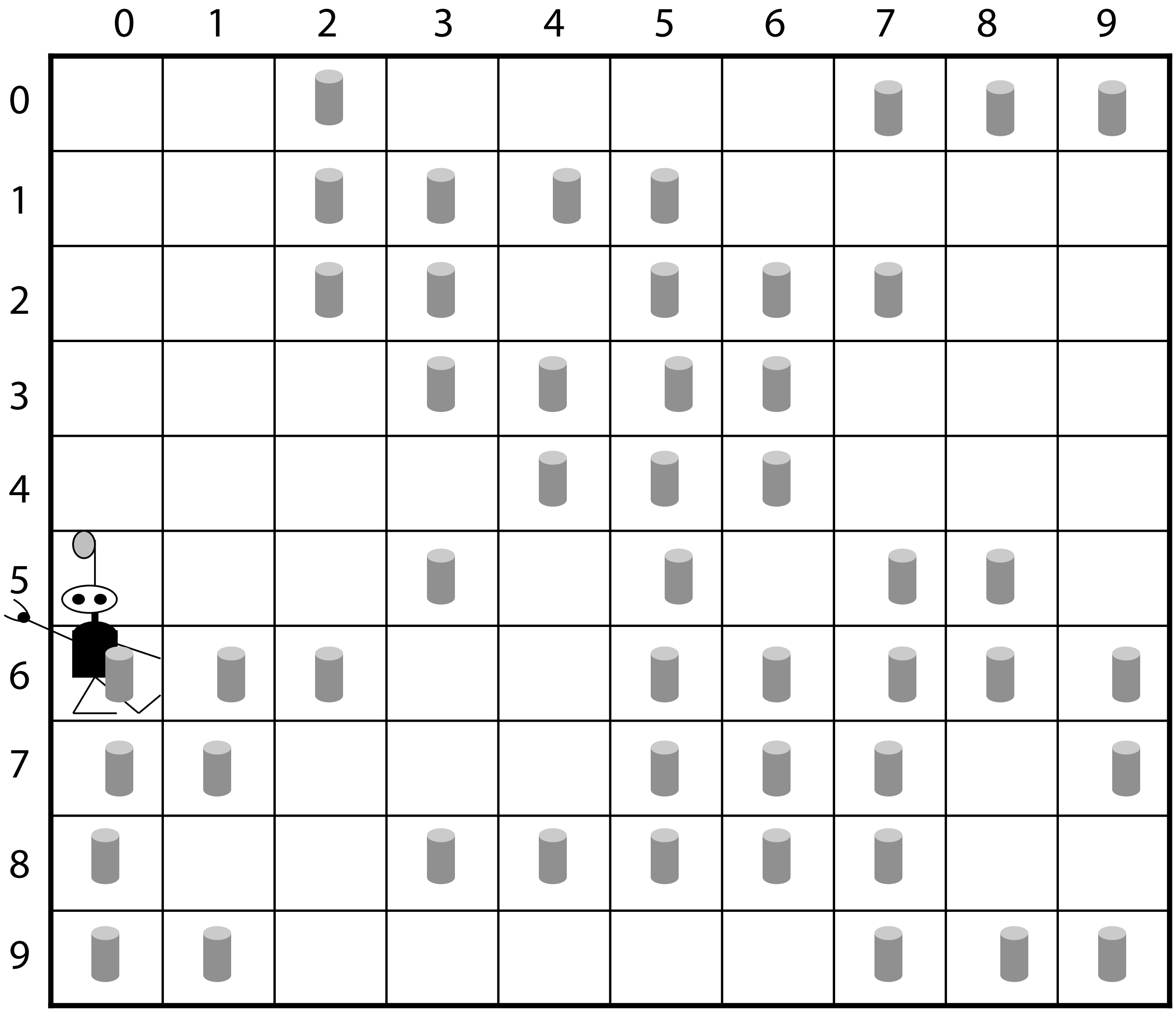


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6	●	●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

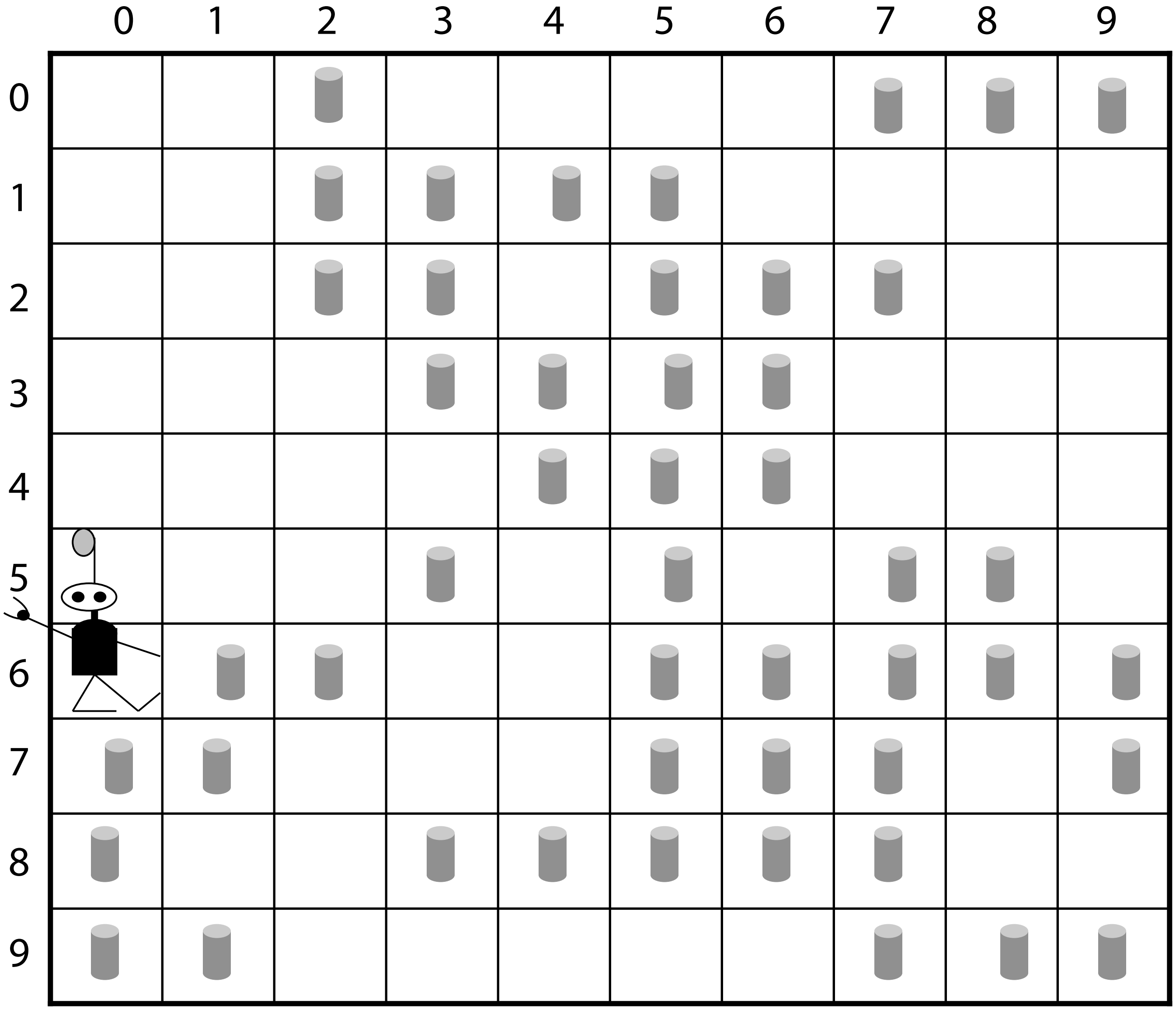
Time: 7 Score: 20



Time: 8 Score: 20

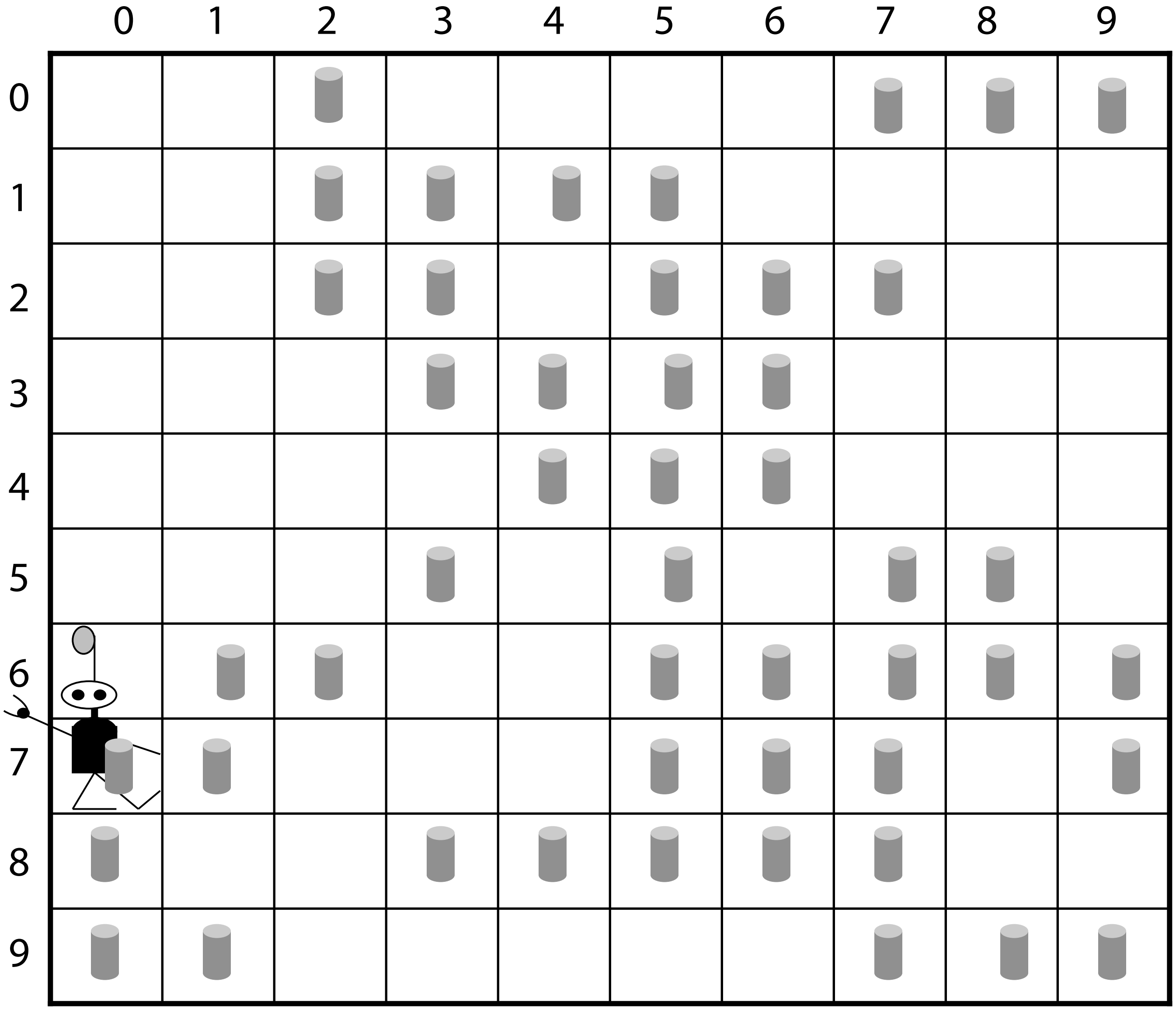


Time: 9 Score: 30

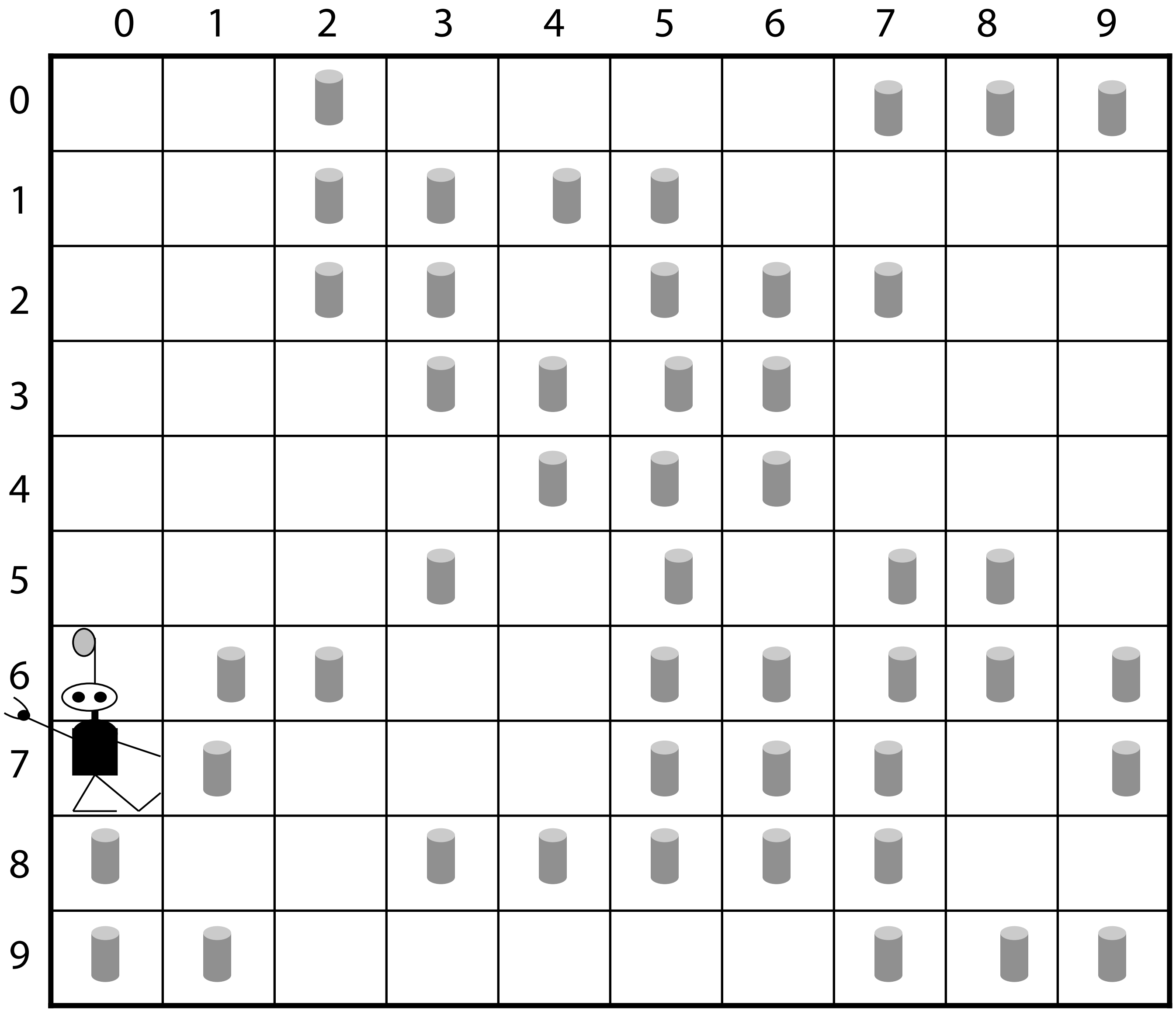


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6		●	●			●	●	●	●	●
7	●	●				●	●	●		●
8	●			●	●	●	●	●		
9	●	●						●	●	●

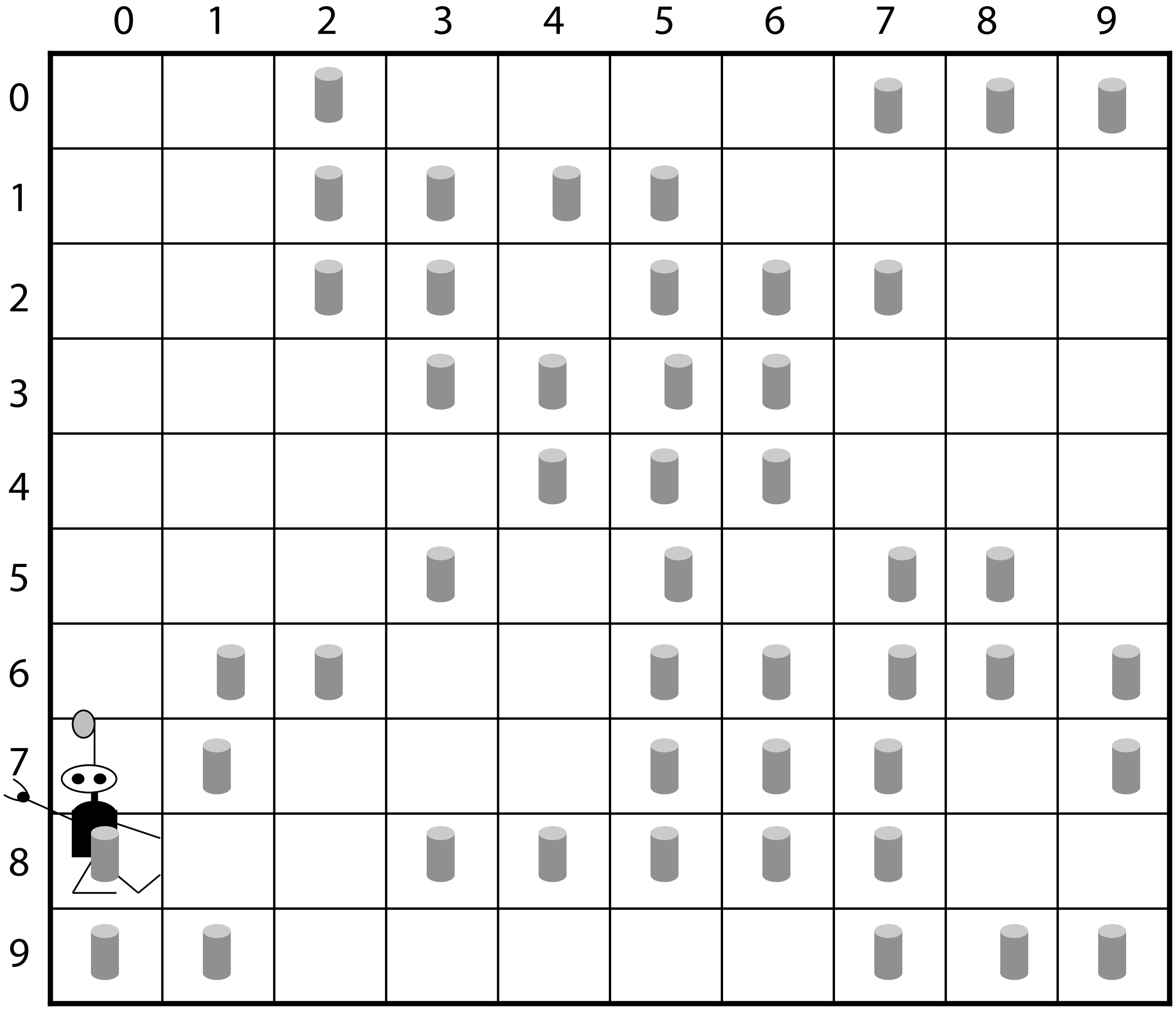
Time: 10 Score: 30



Time: 11 Score: 40

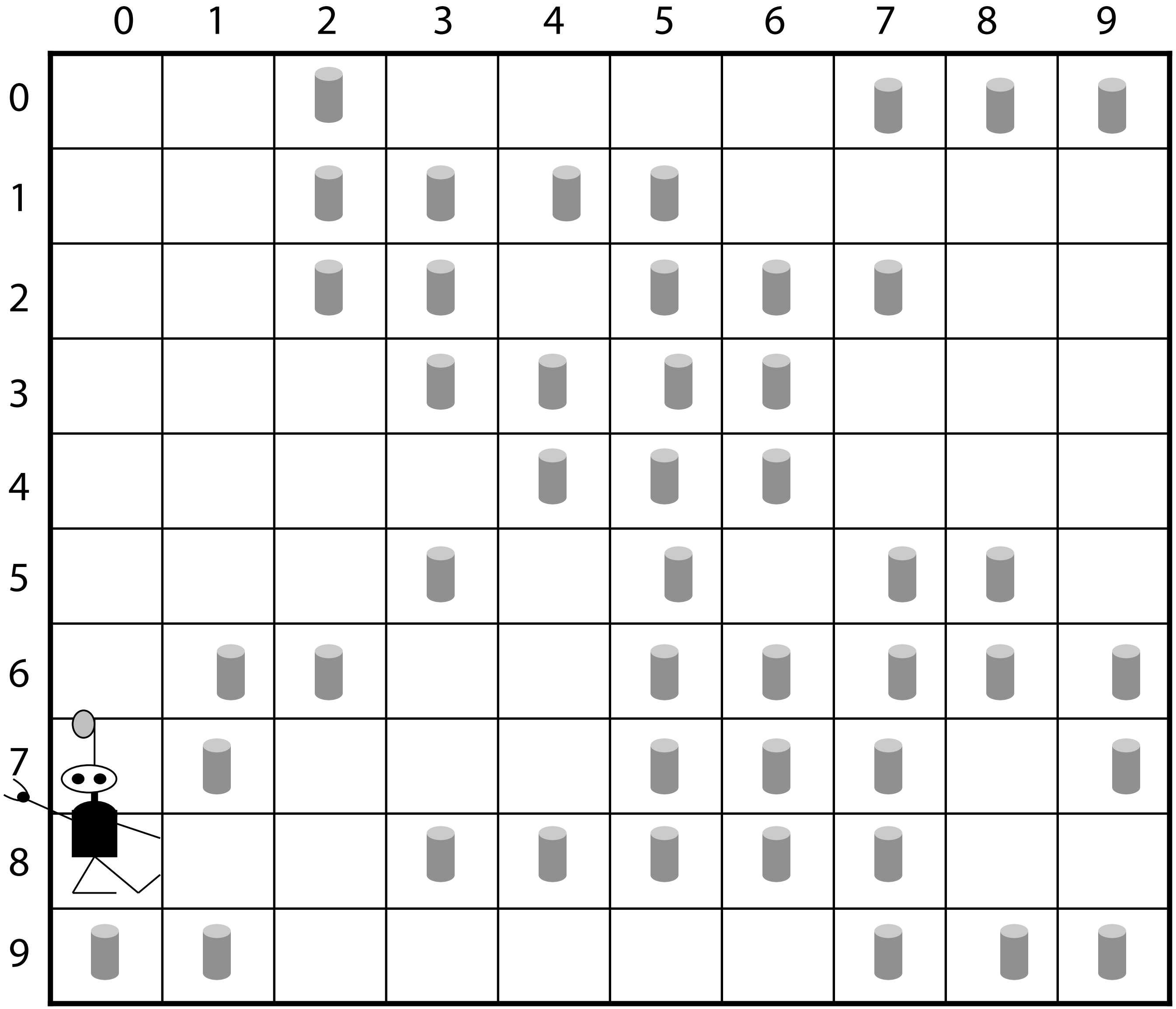


Time: 12   Score: 40

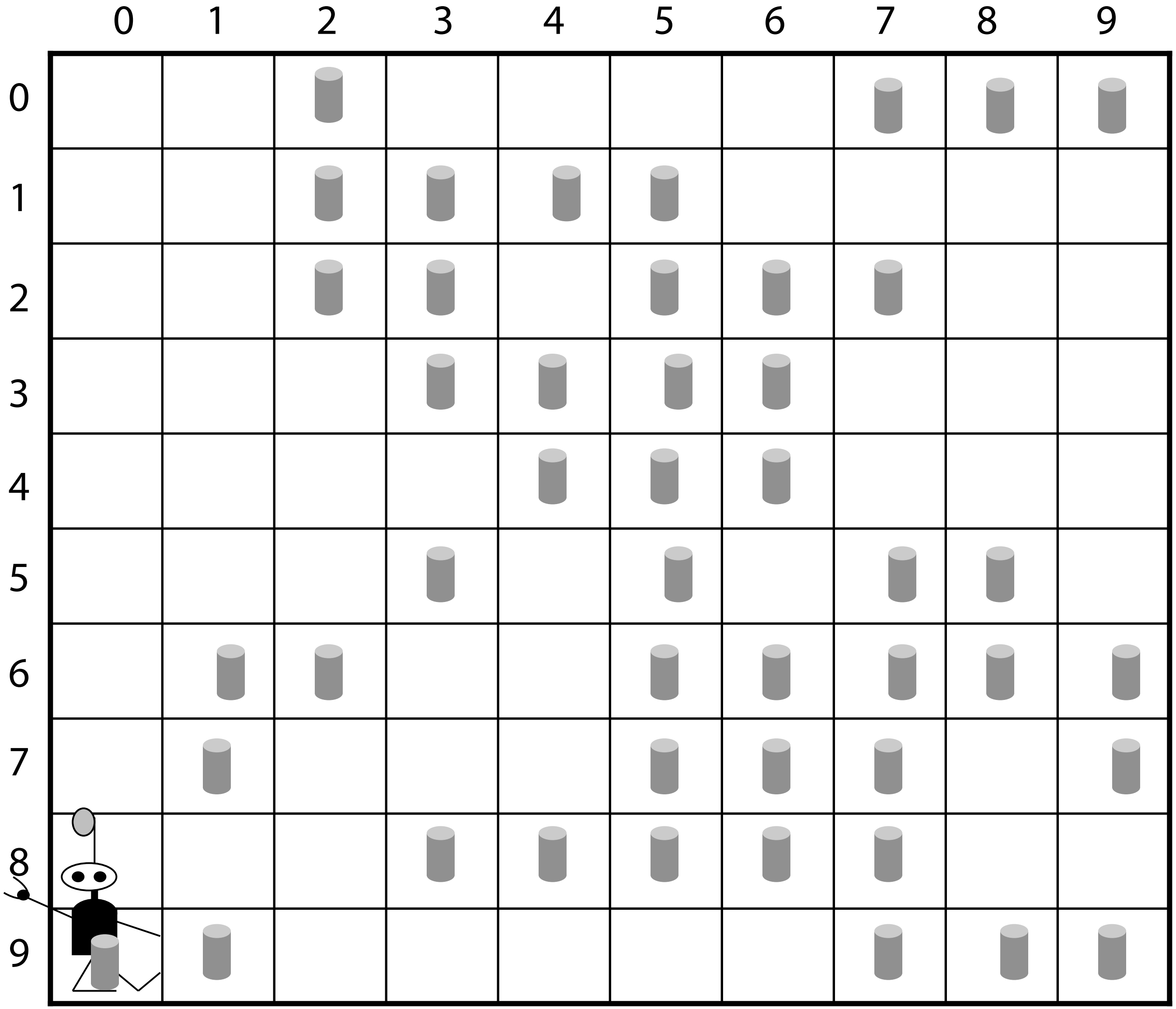


	0	1	2	3	4	5	6	7	8	9
0			▮					▮	▮	▮
1			▮	▮	▮	▮				
2			▮	▮		▮	▮	▮		
3				▮	▮	▮	▮			
4					▮	▮	▮			
5				▮		▮		▮	▮	
6		▮	▮			▮	▮	▮	▮	▮
7		▮				▮	▮	▮		▮
8	▮			▮	▮	▮	▮	▮		
9	▮	▮						▮	▮	▮

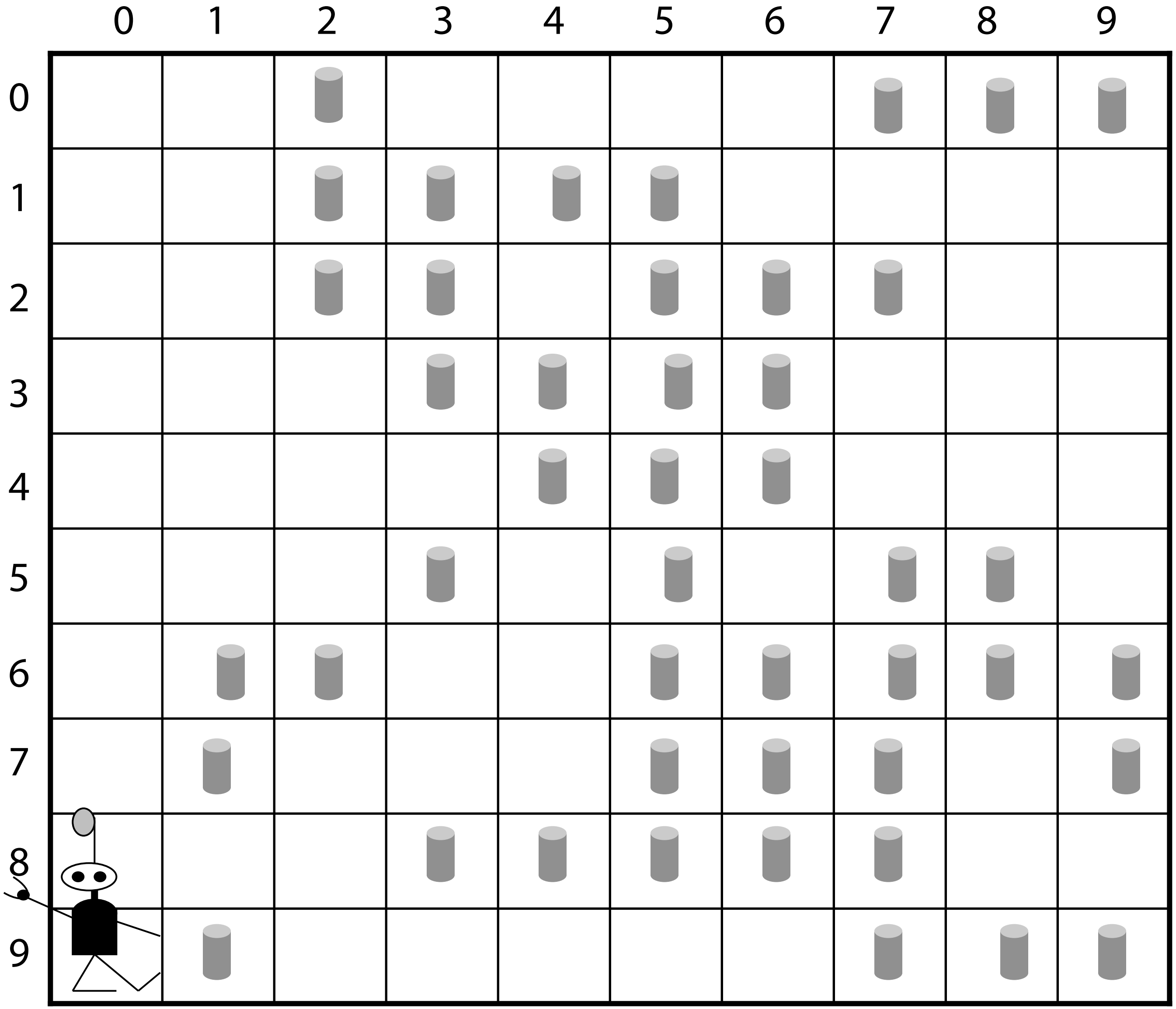
Time: 13   Score: 50



Time: 14 Score: 50

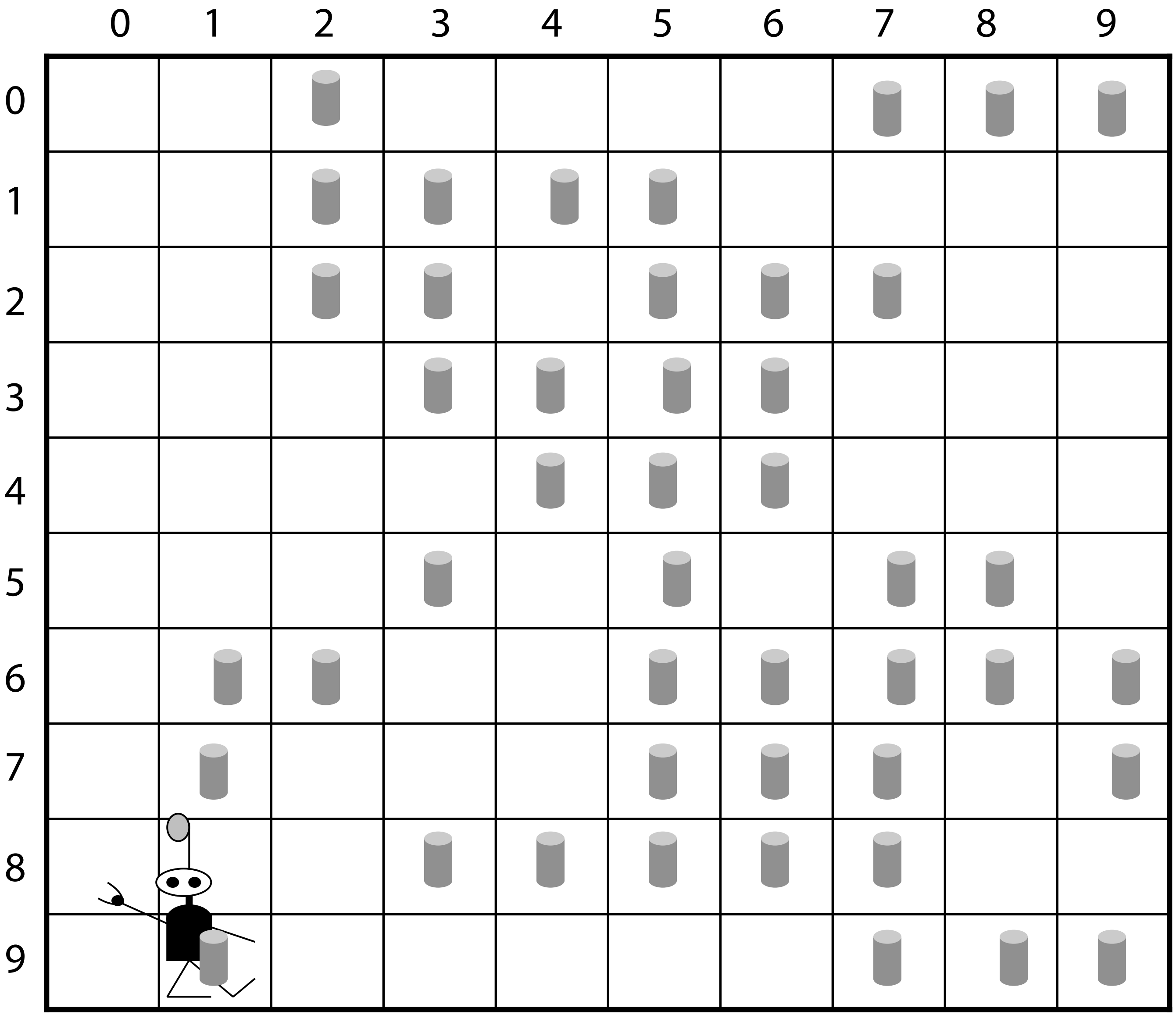


Time: 15 Score: 60

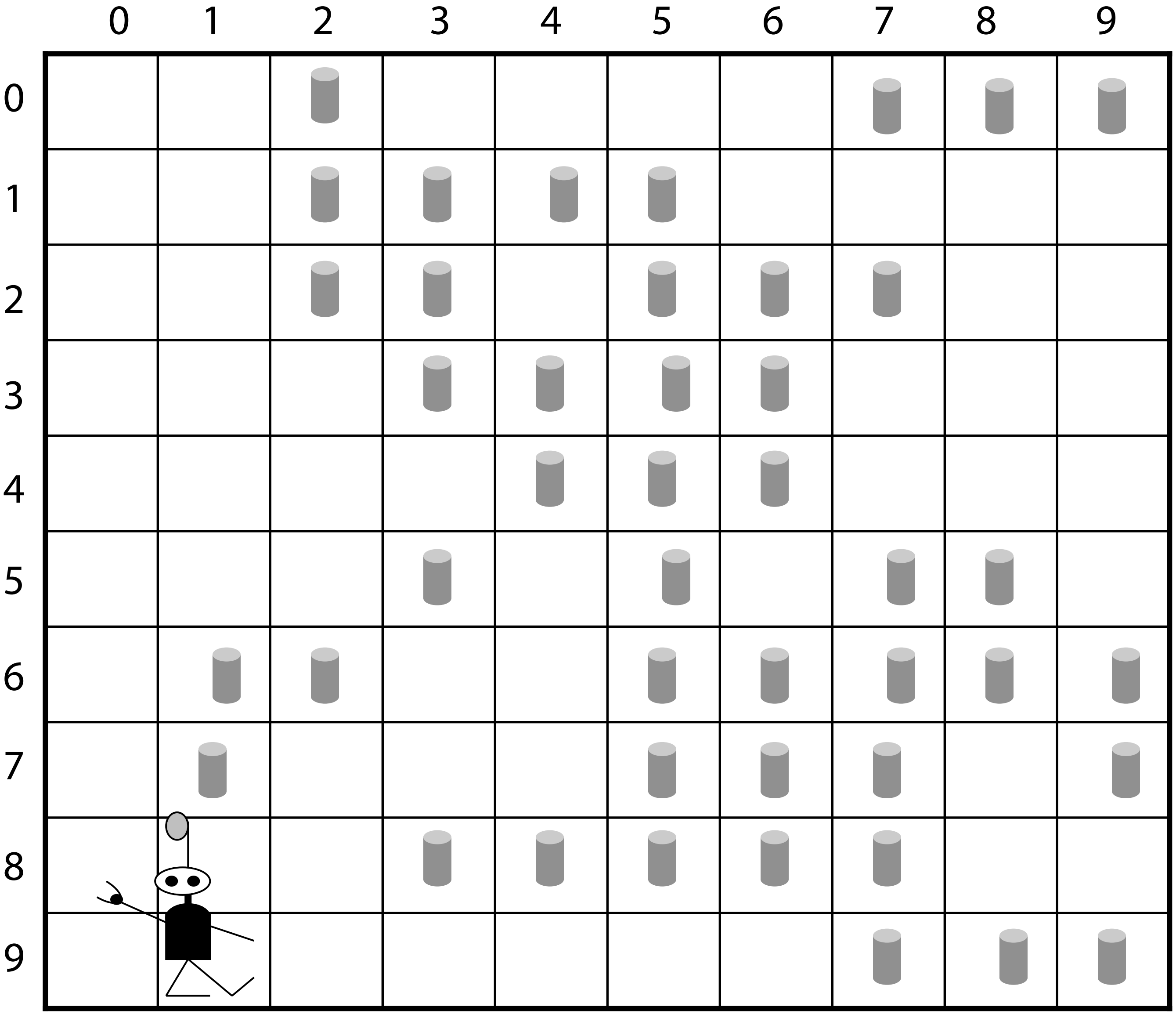


	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6		●	●			●	●	●	●	●
7		●				●	●	●		●
8				●	●	●	●	●		
9		●						●	●	●

Time: 16   Score: 60

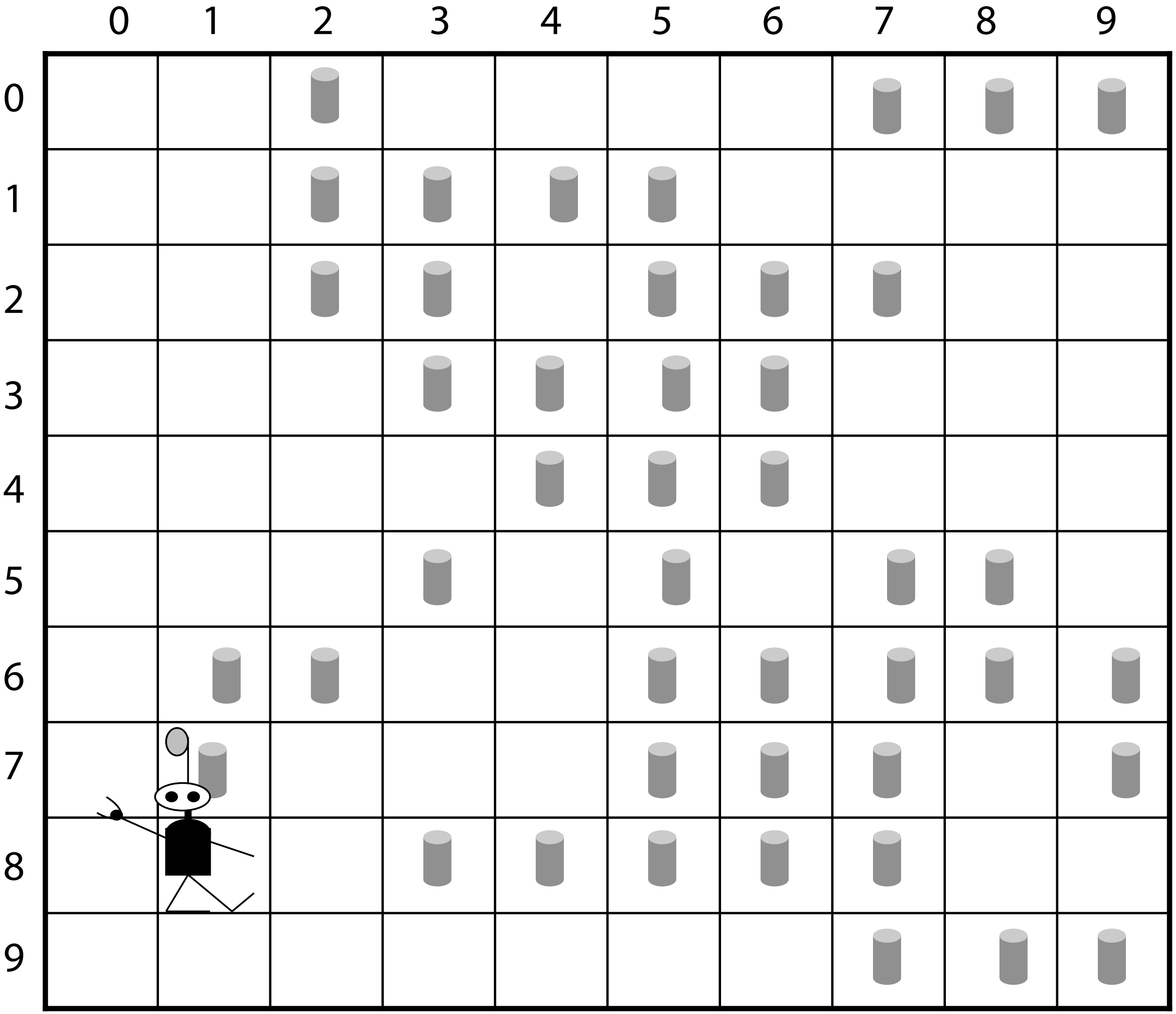


Time: 17 Score: 70



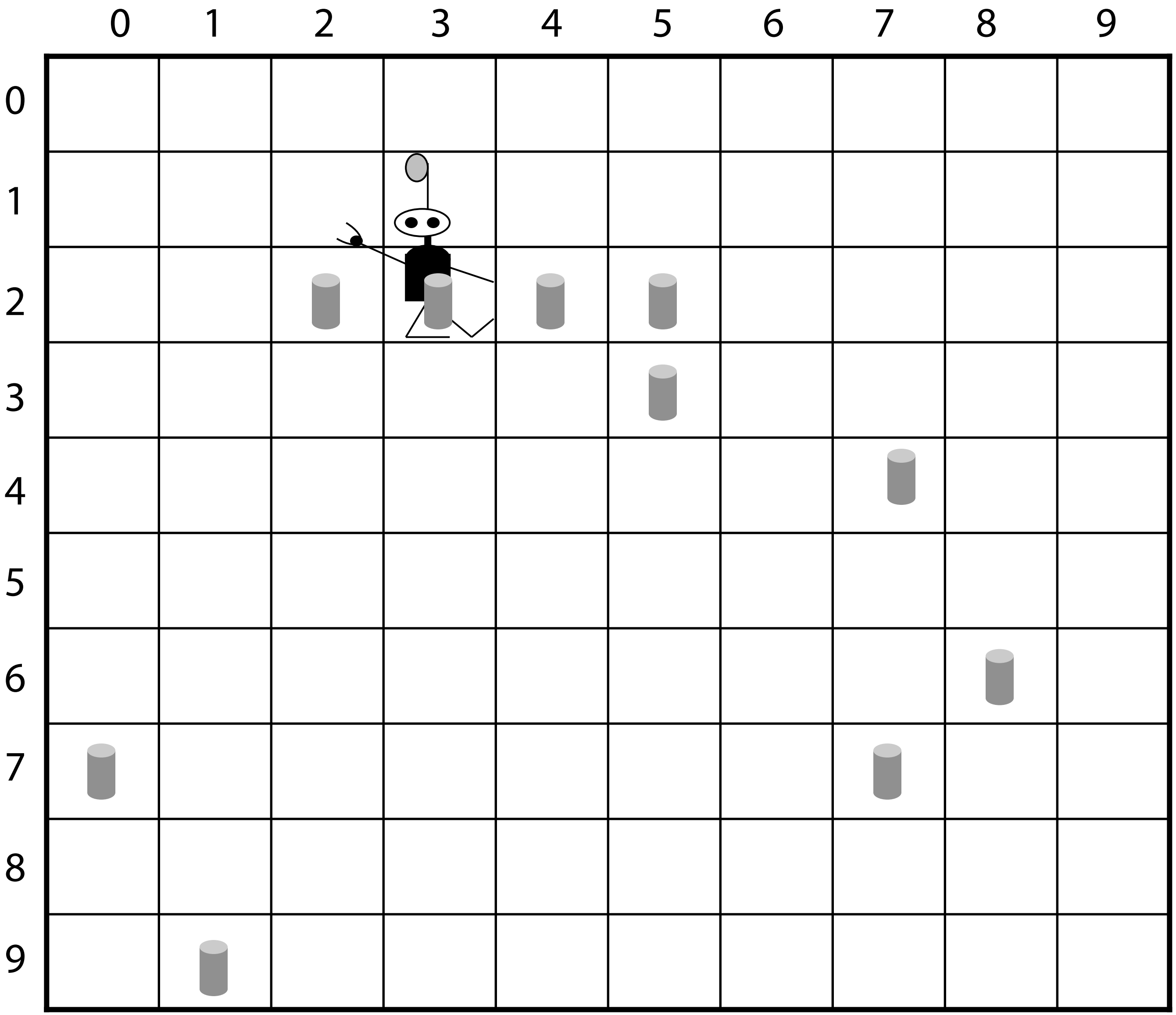
	0	1	2	3	4	5	6	7	8	9
0			●					●	●	●
1			●	●	●	●				
2			●	●		●	●	●		
3				●	●	●	●			
4					●	●	●			
5				●		●		●	●	
6		●	●			●	●	●	●	●
7		●				●	●	●		●
8				●	●	●	●	●		
9								●	●	●

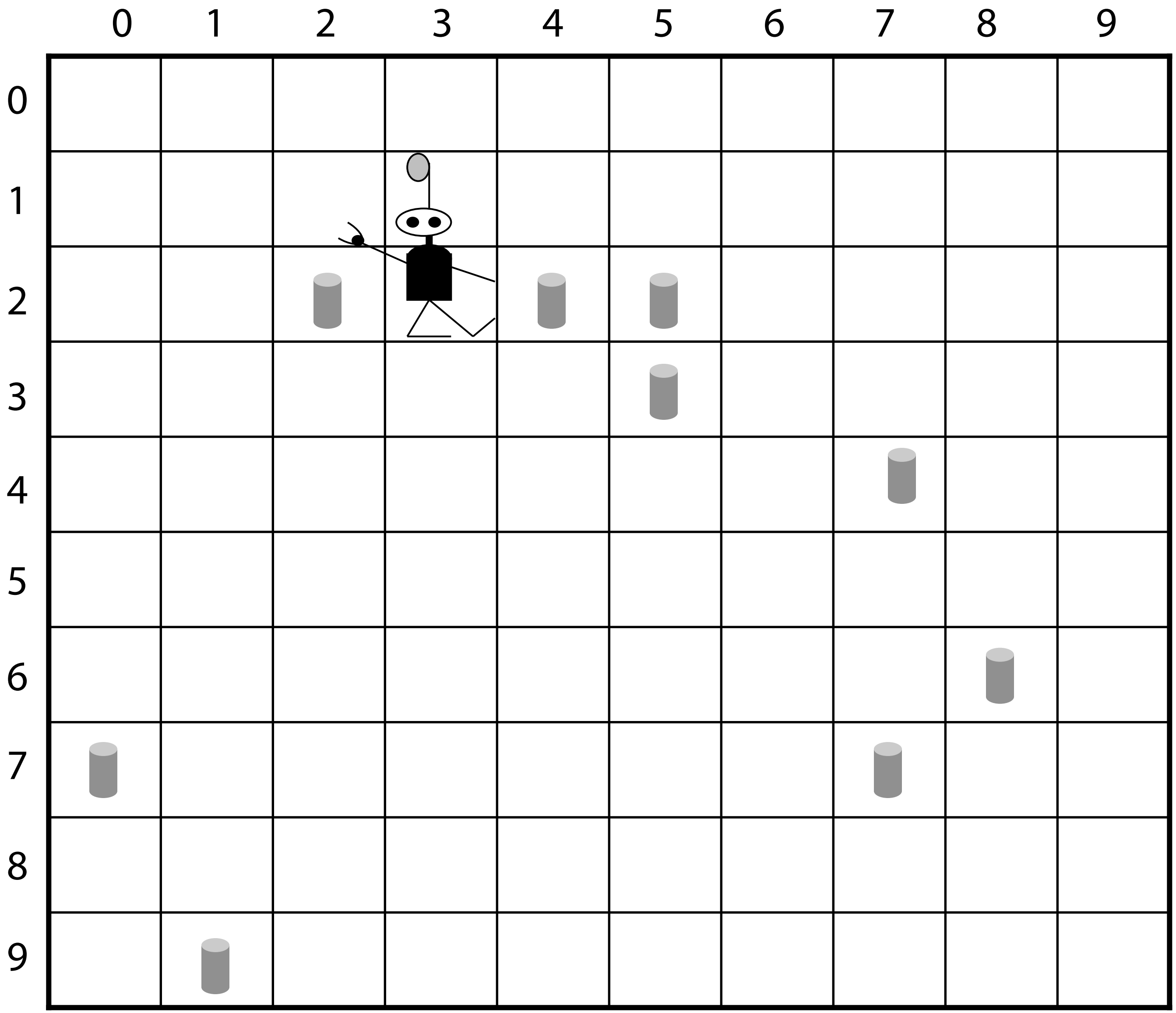
Time: 18 Score: 70

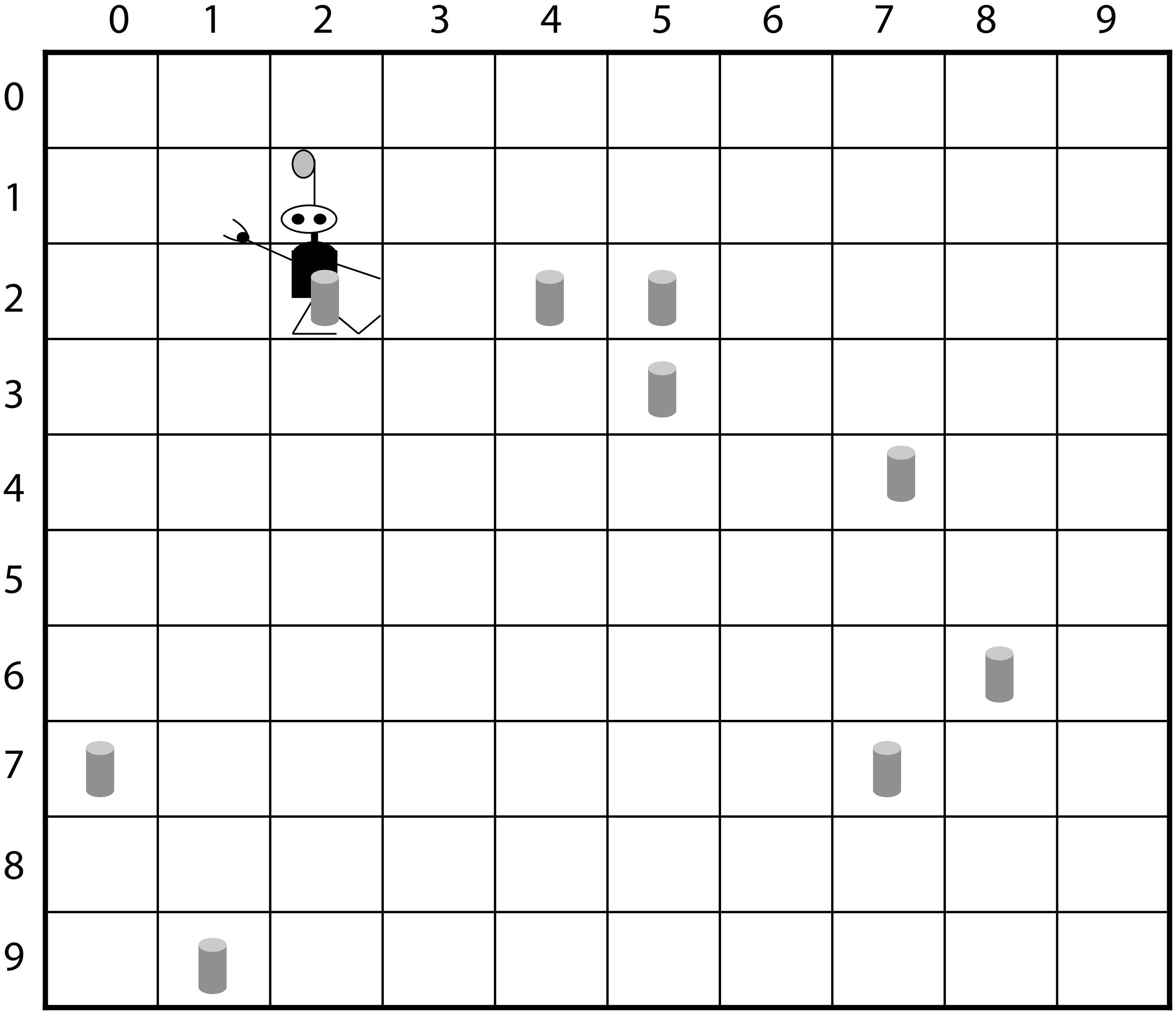


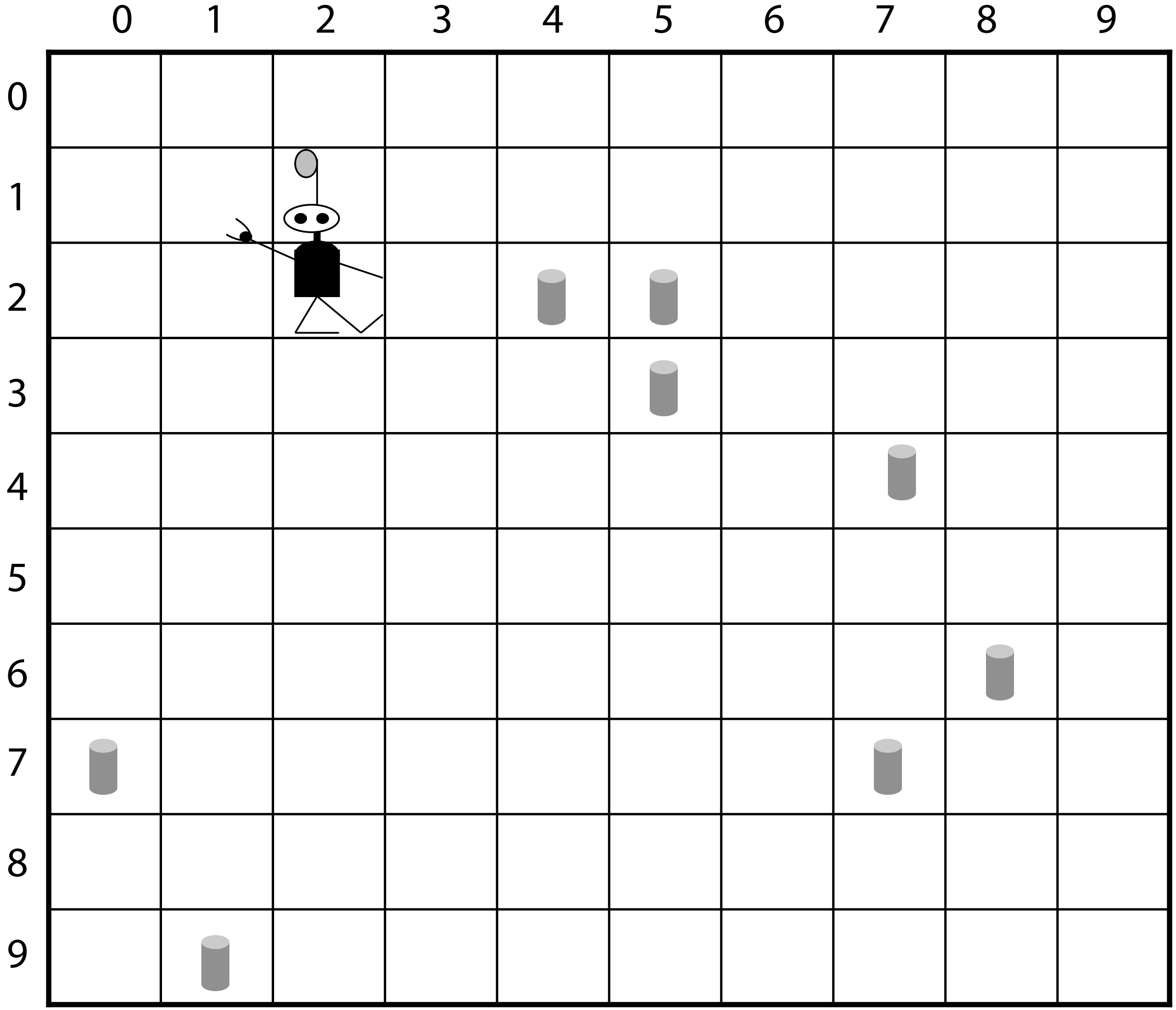
Why Did The GA’s Strategy Outperform Mine?

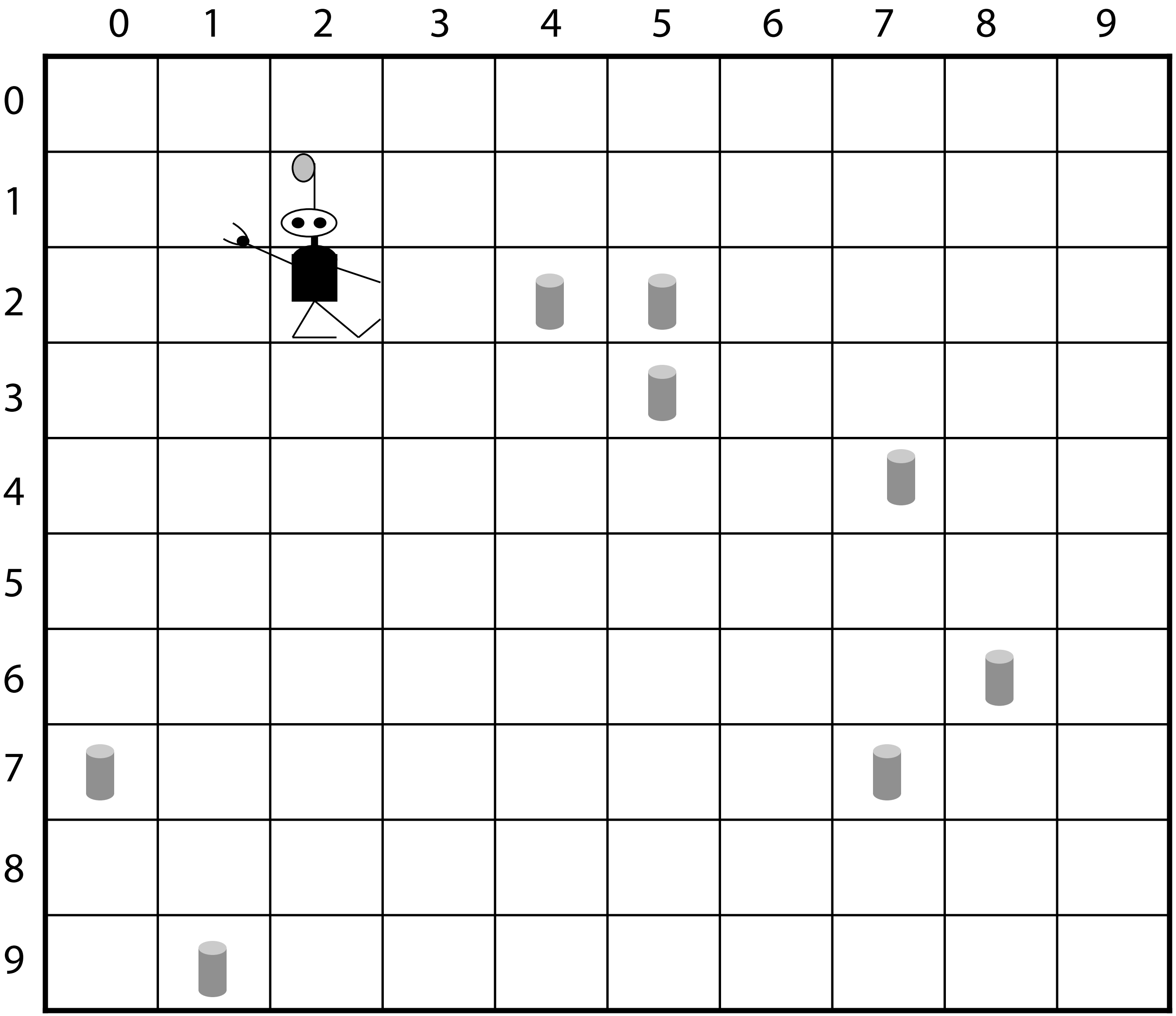
My Strategy


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

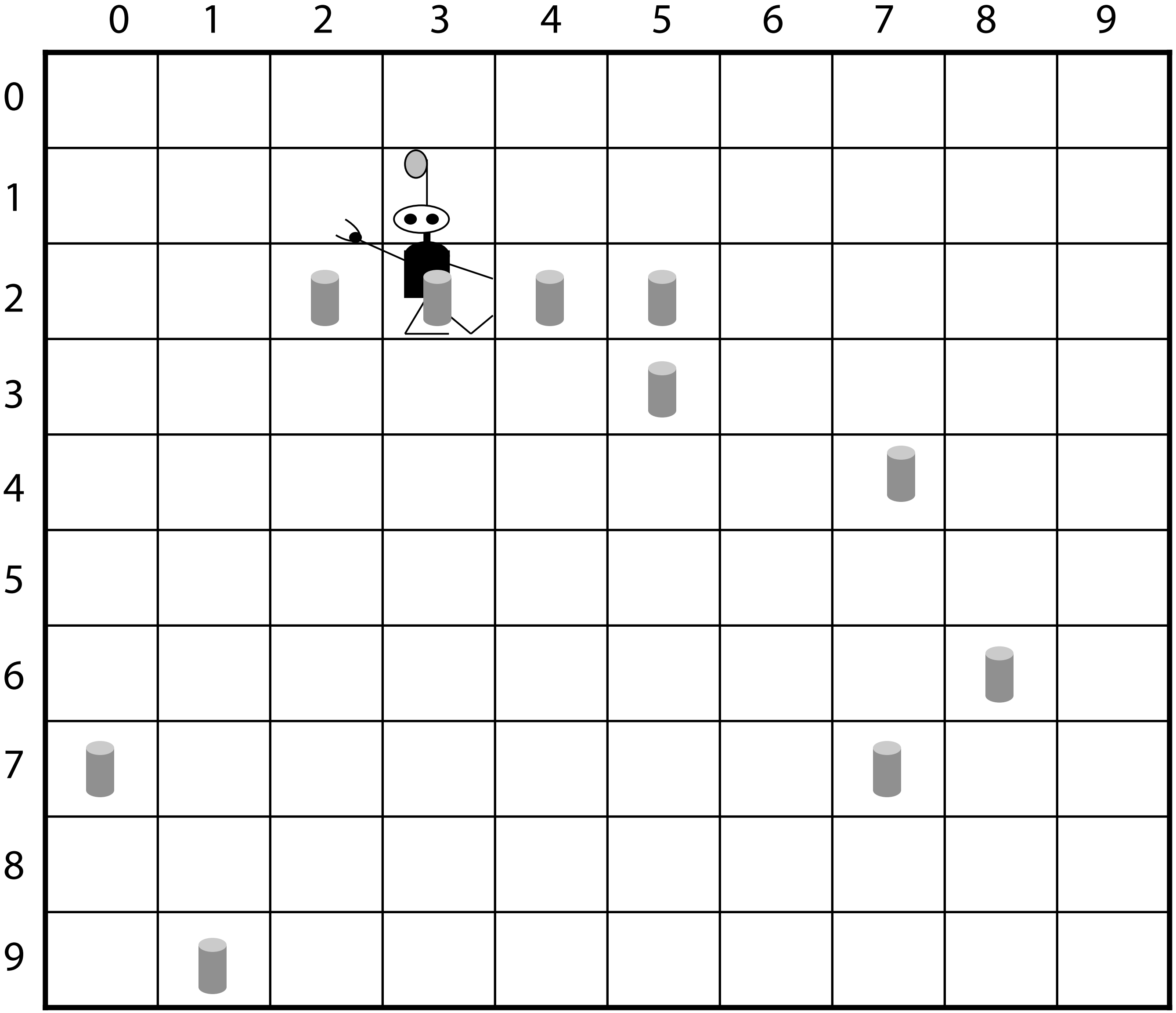

0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9

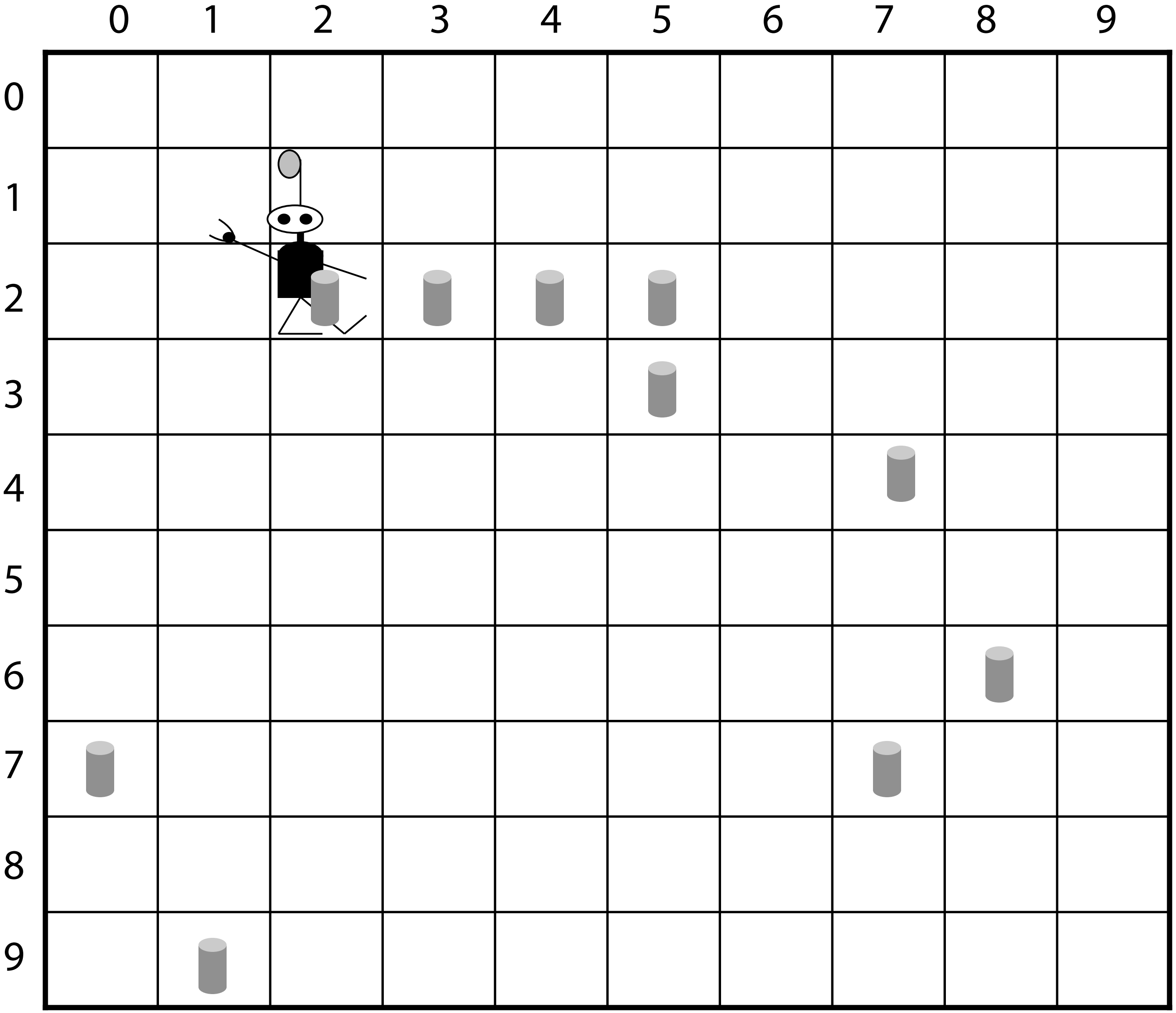

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

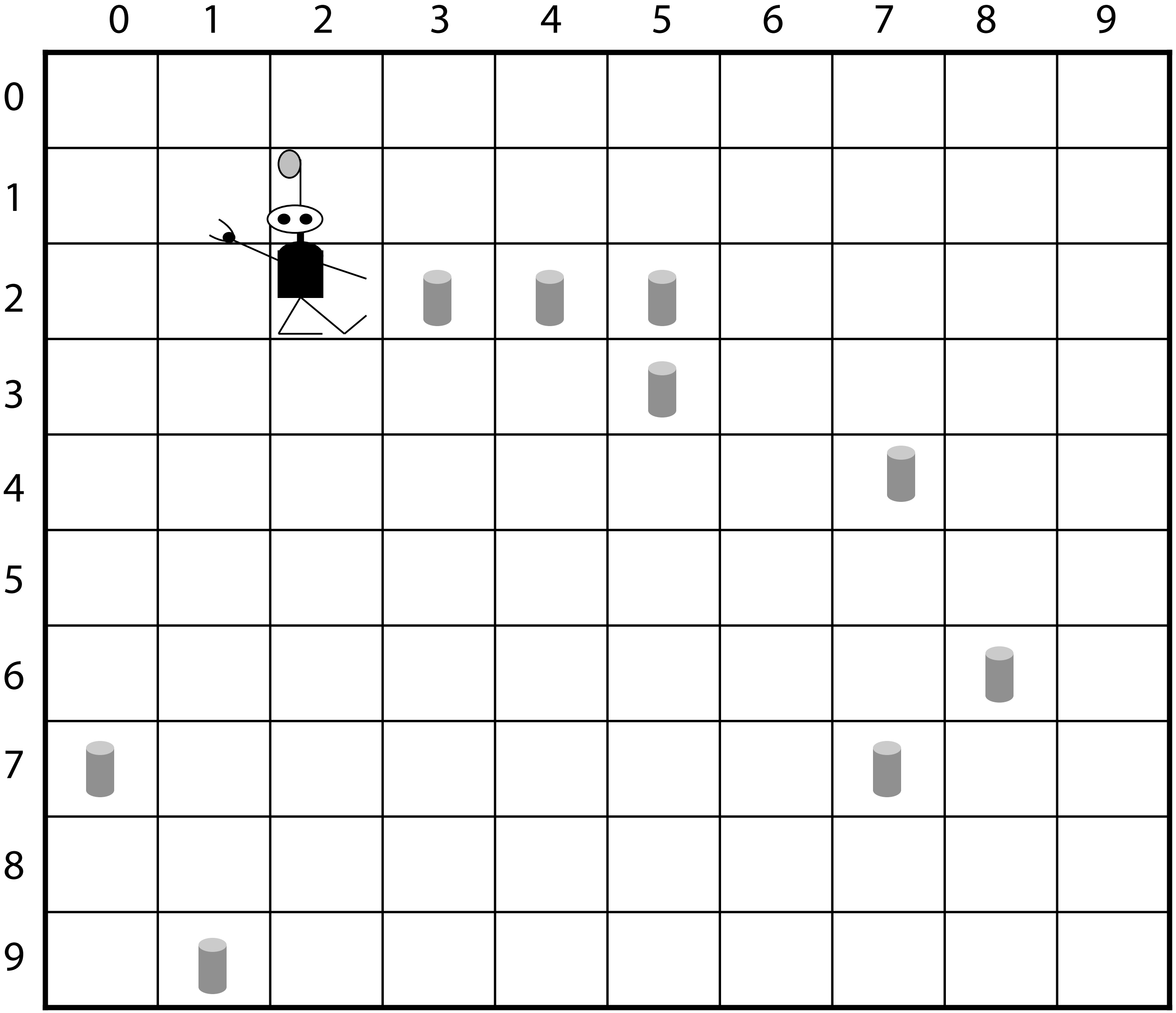

0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9

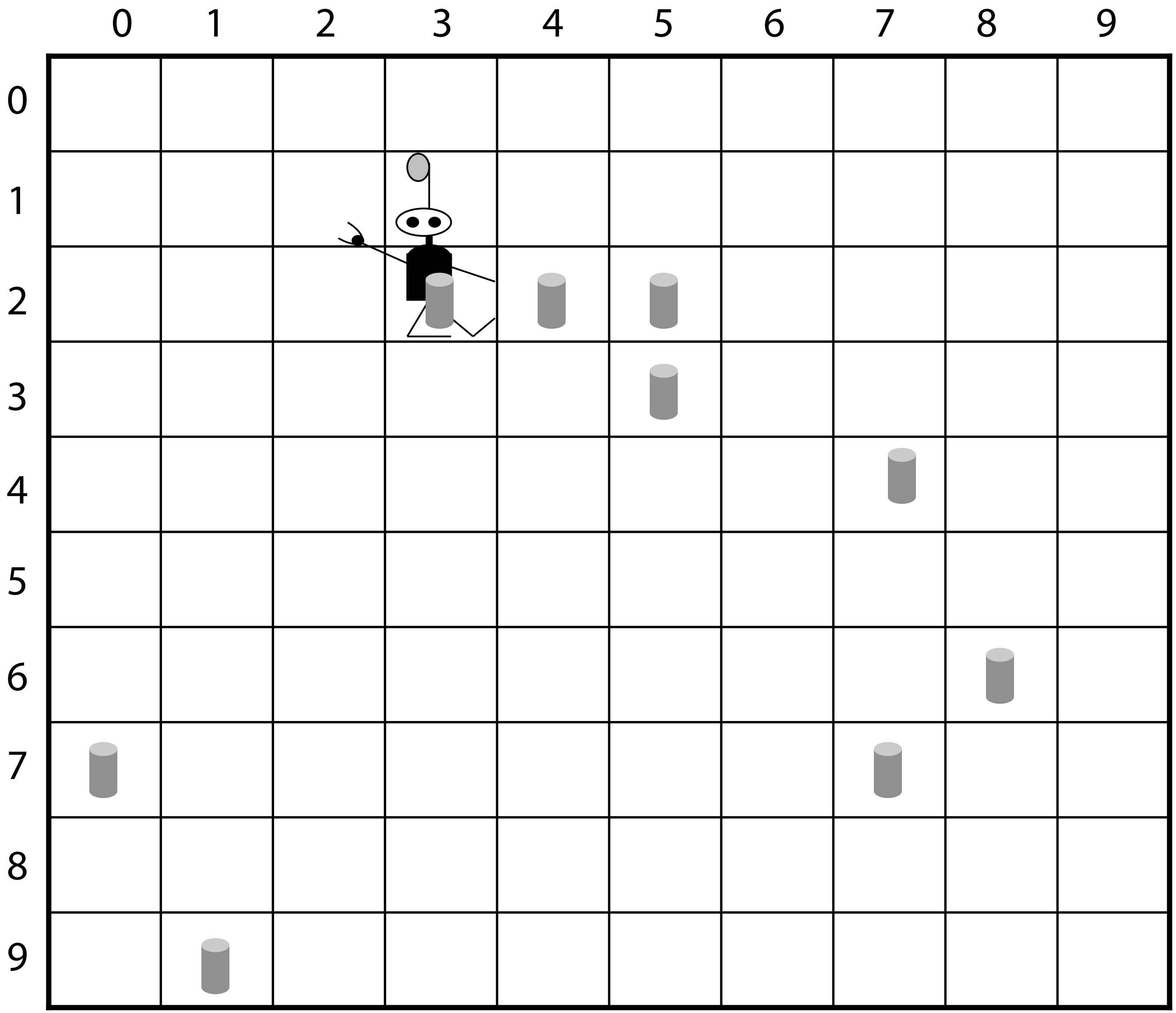

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

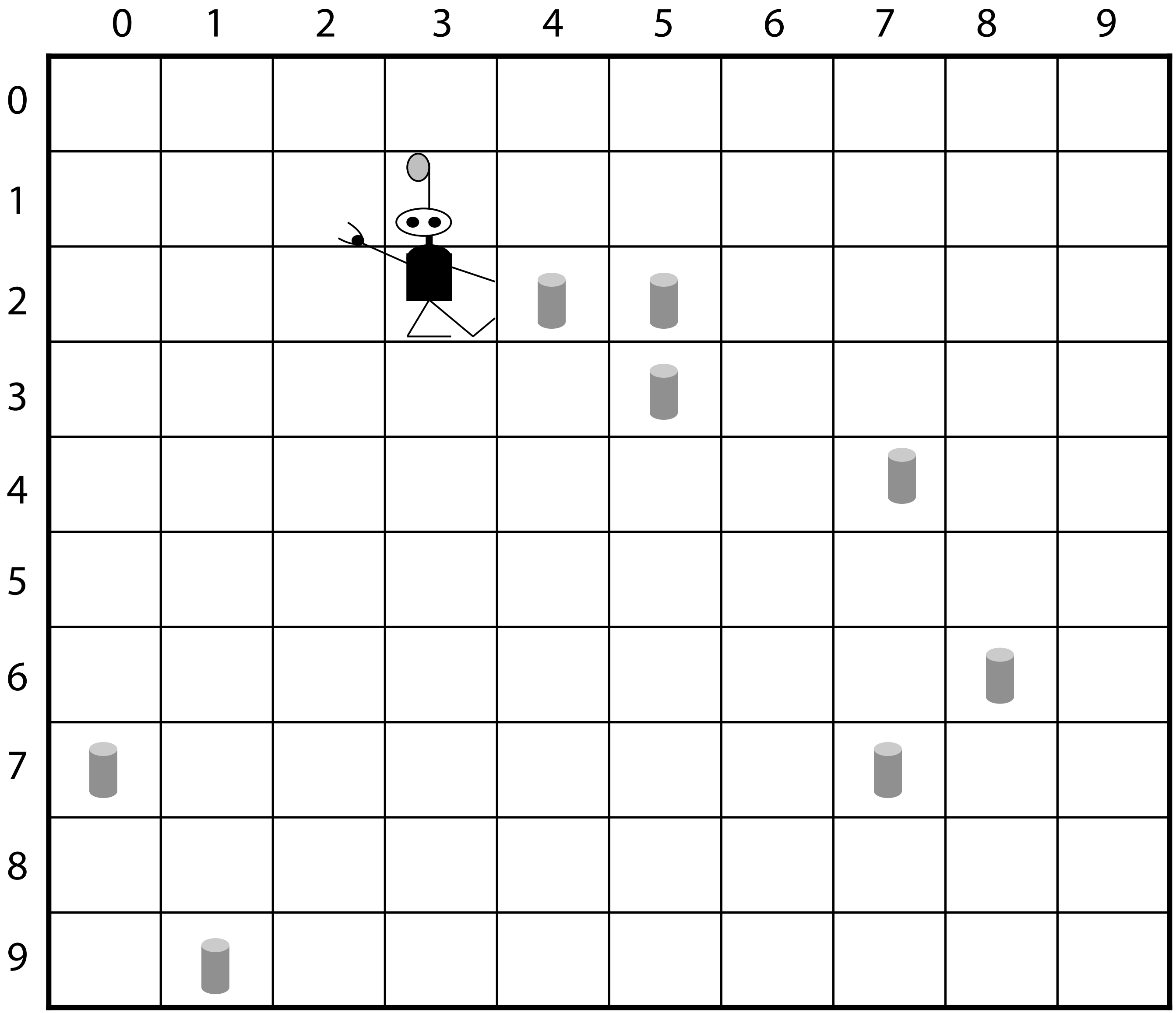
The GA’s Evolved Strategy

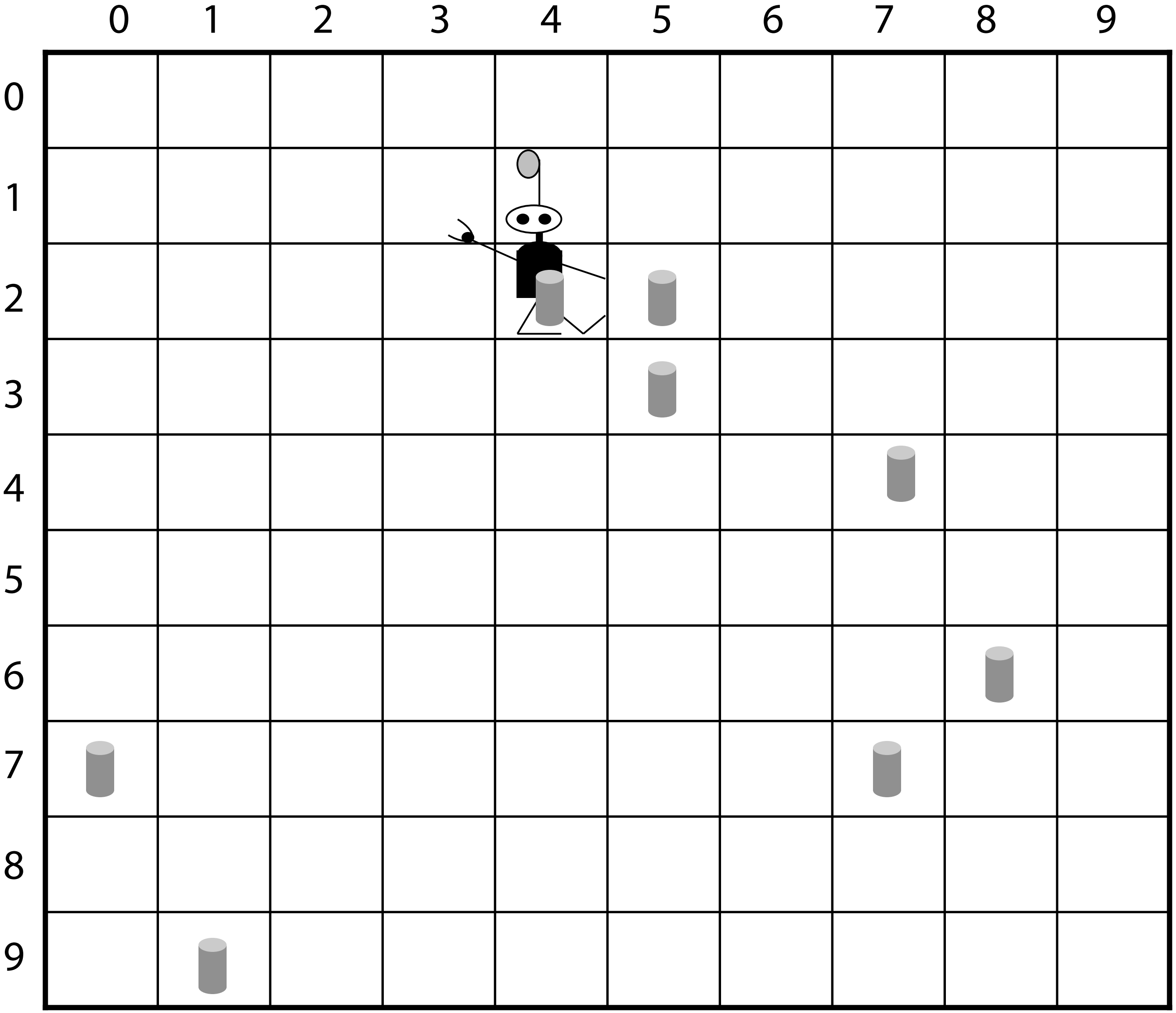

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

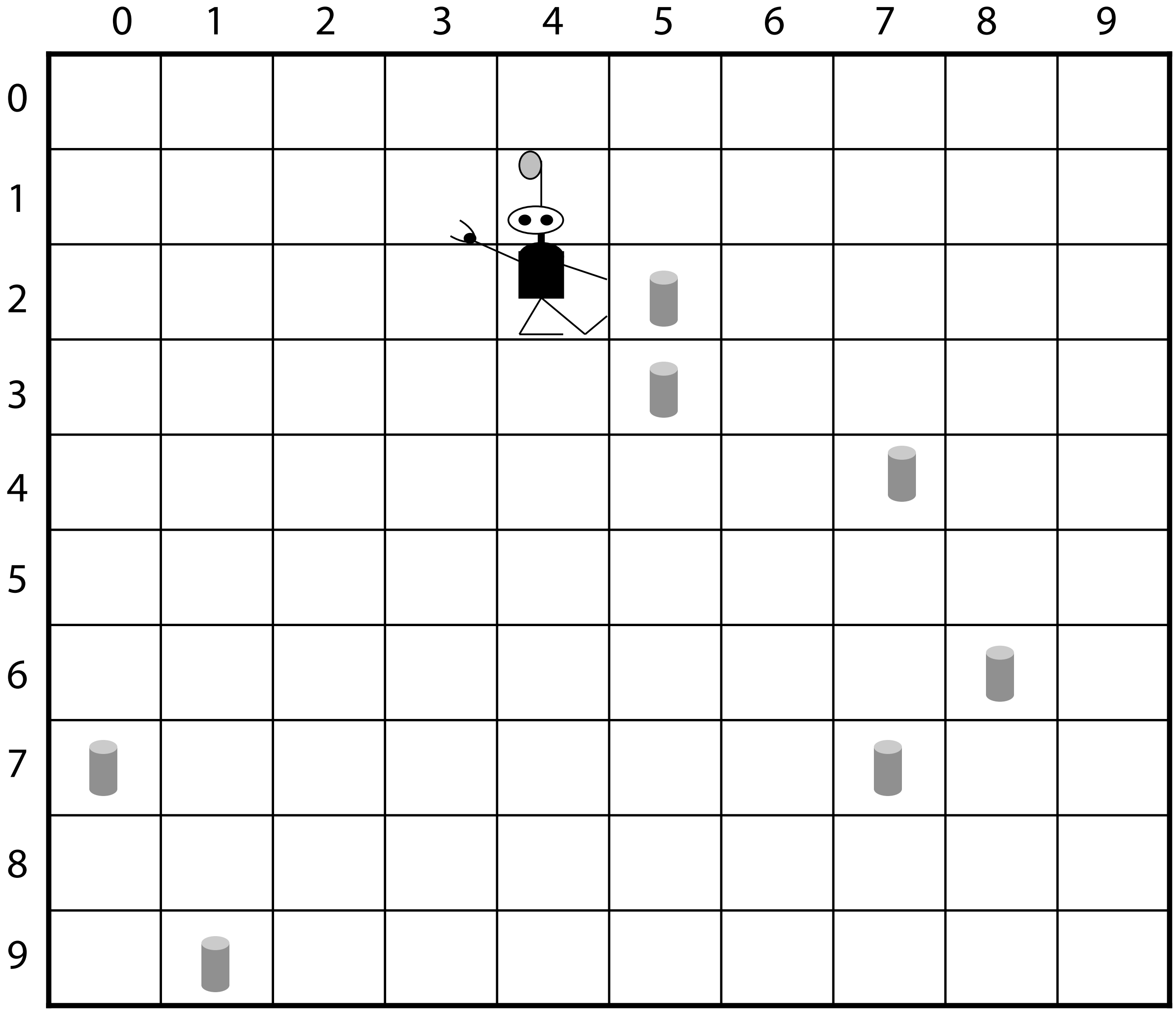

0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9

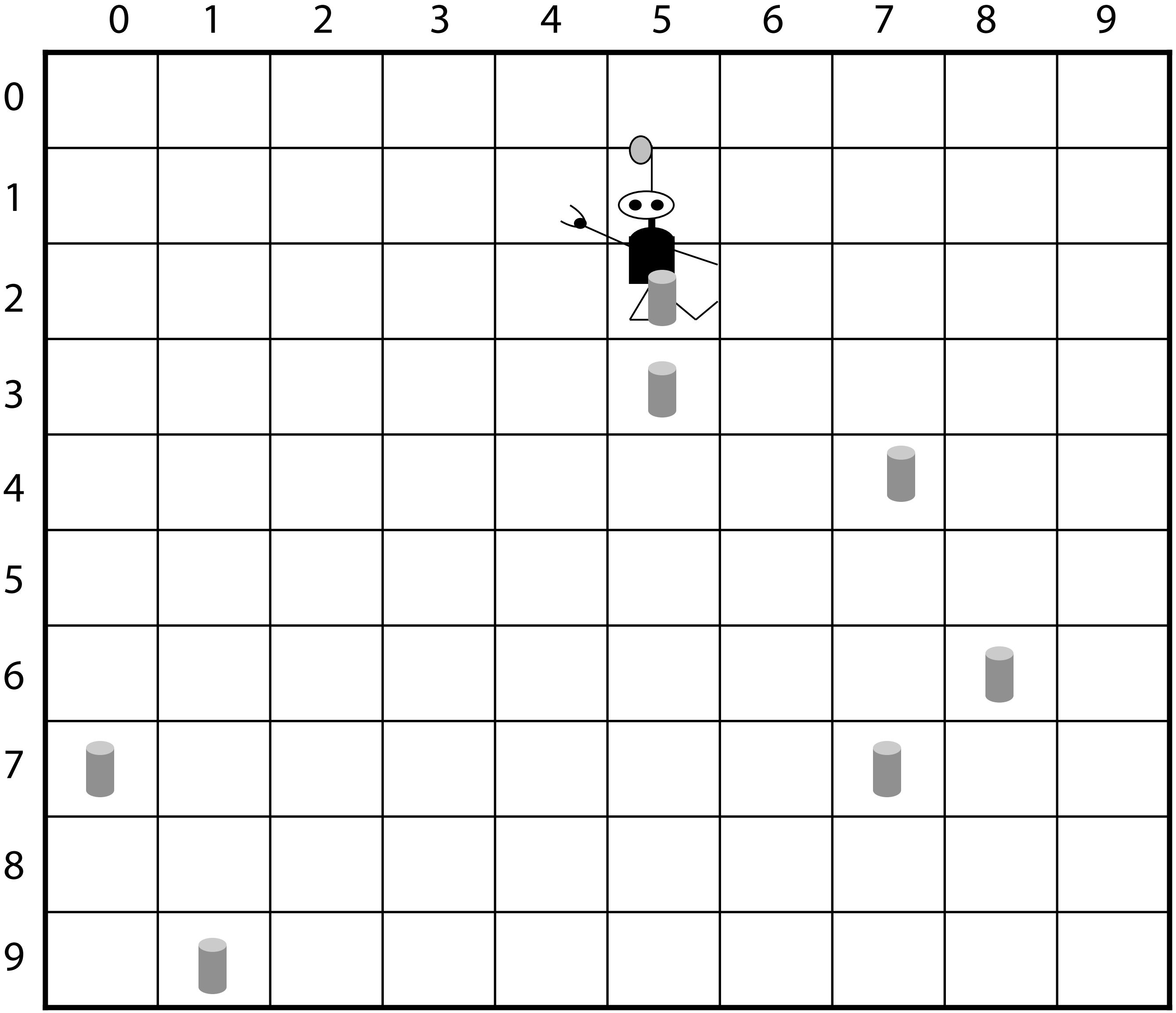

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

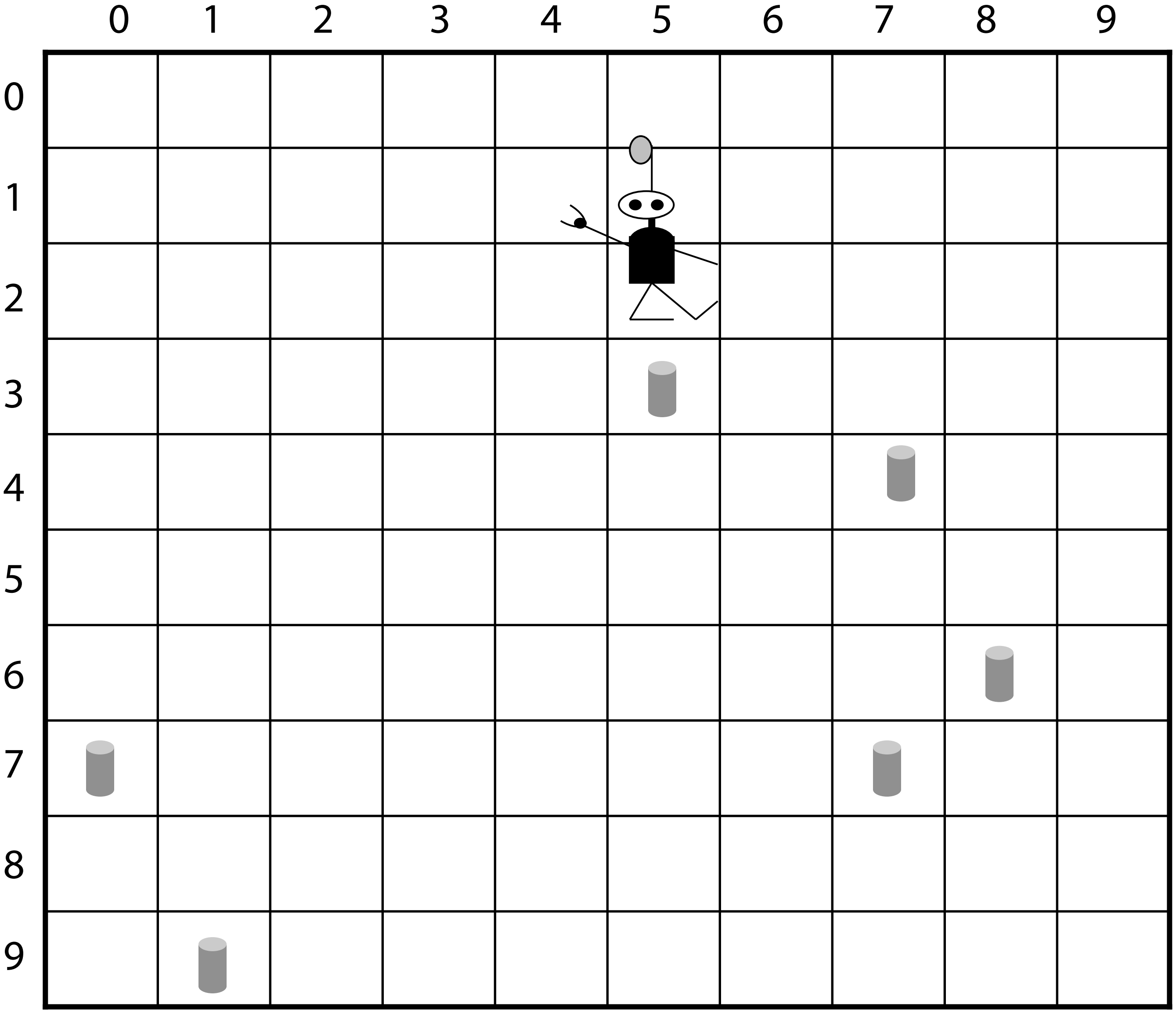

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

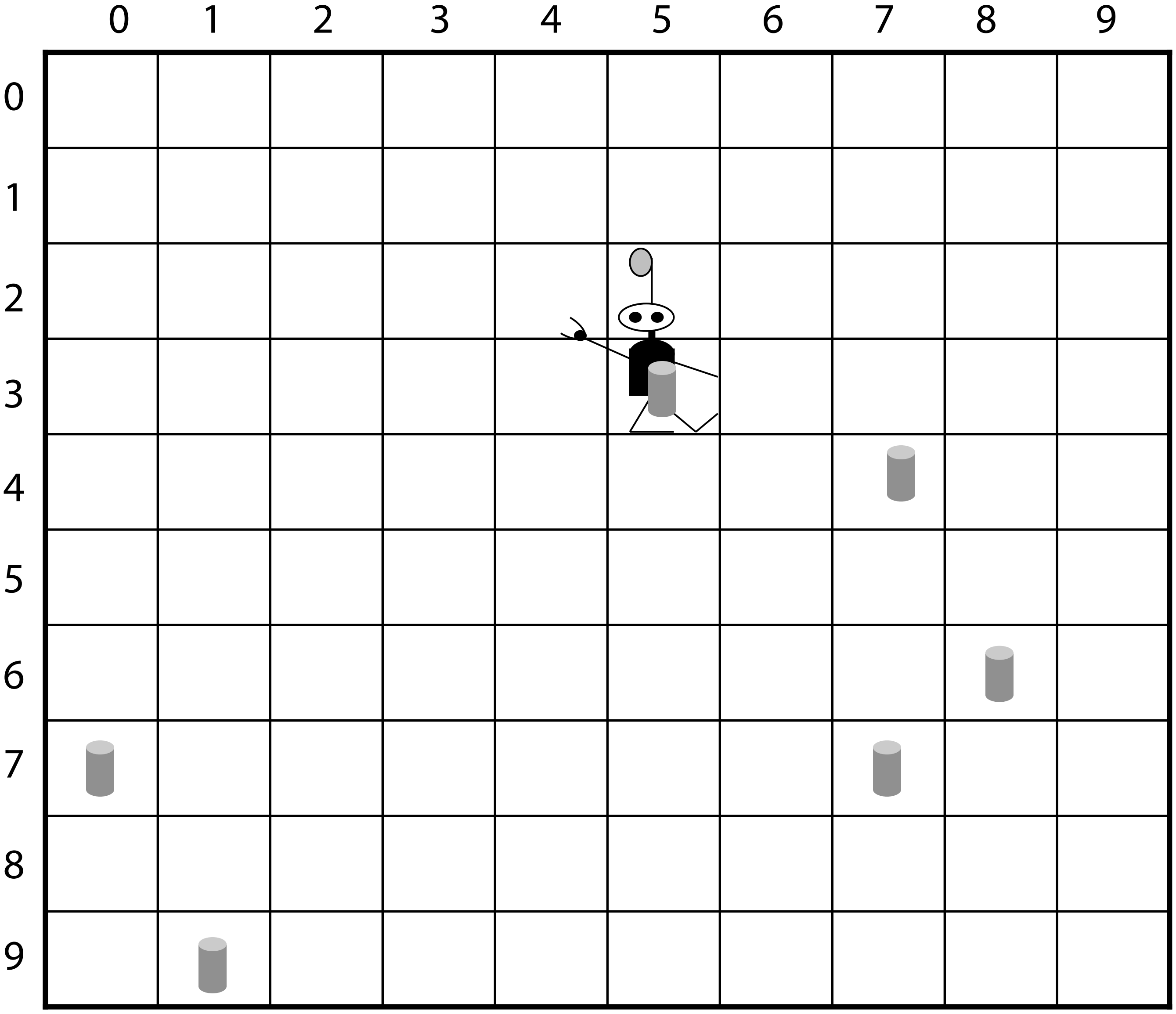

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9

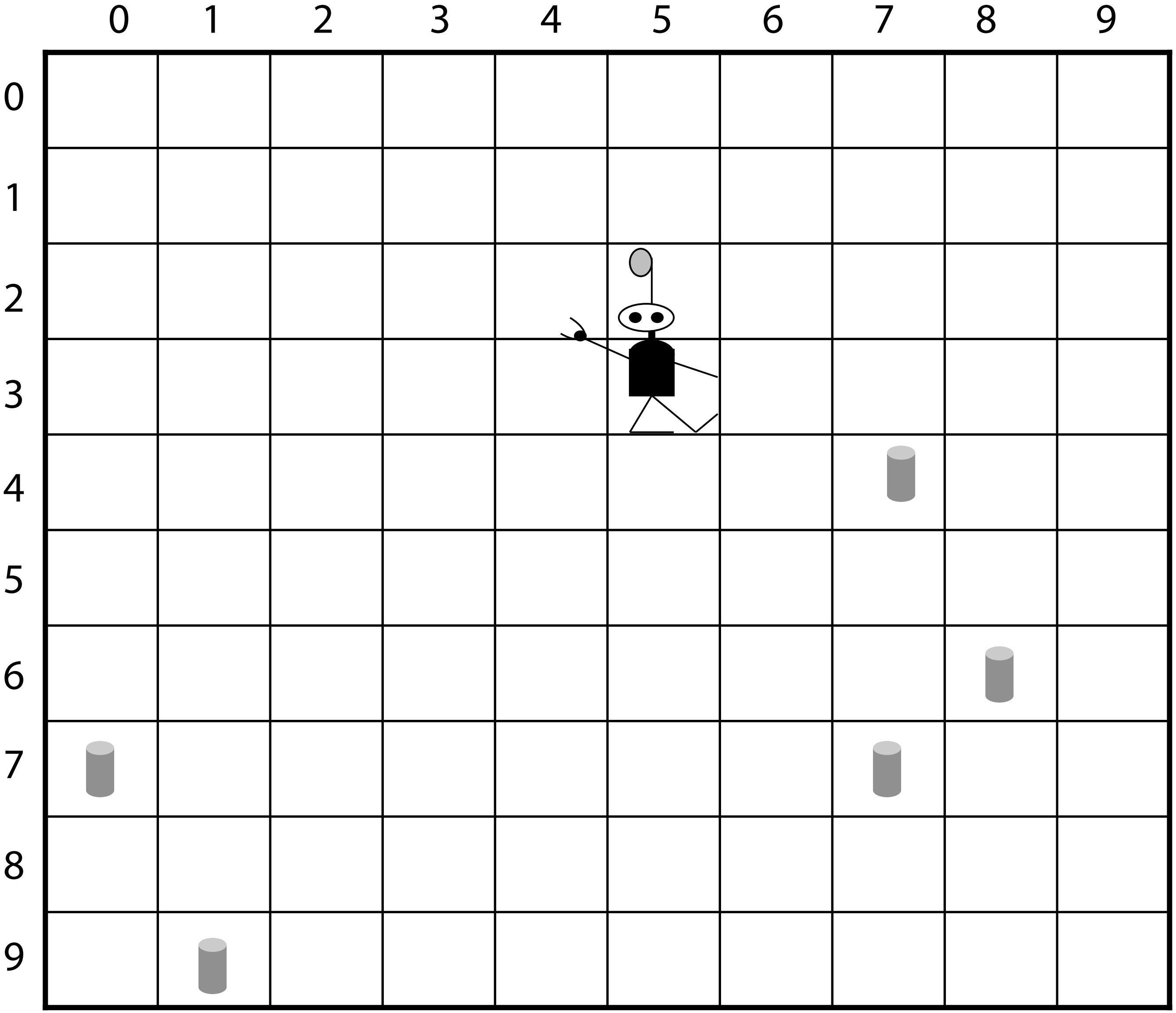

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9


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


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


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


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


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