

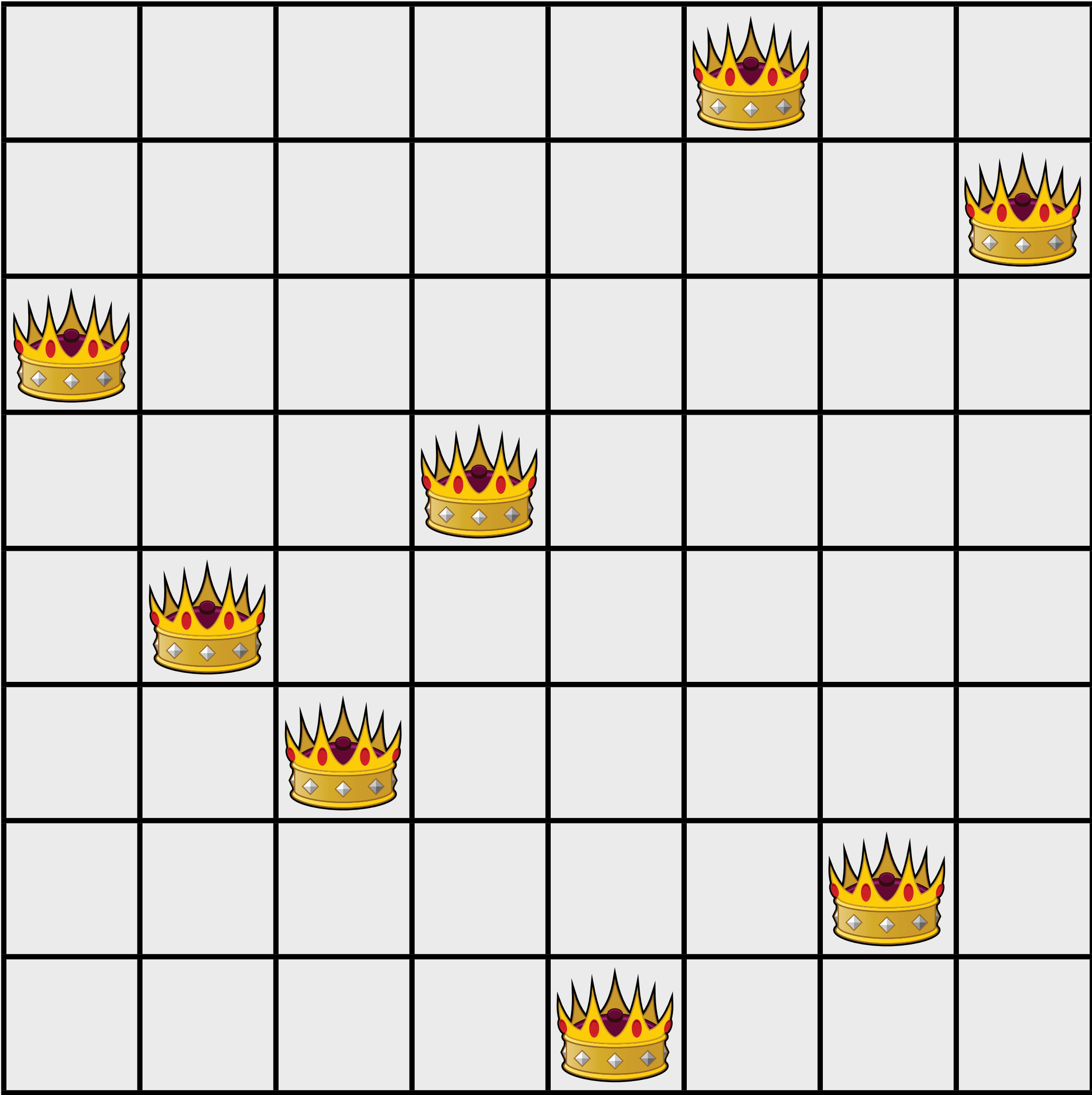
# Discrete Optimization

## Local Search: Part I

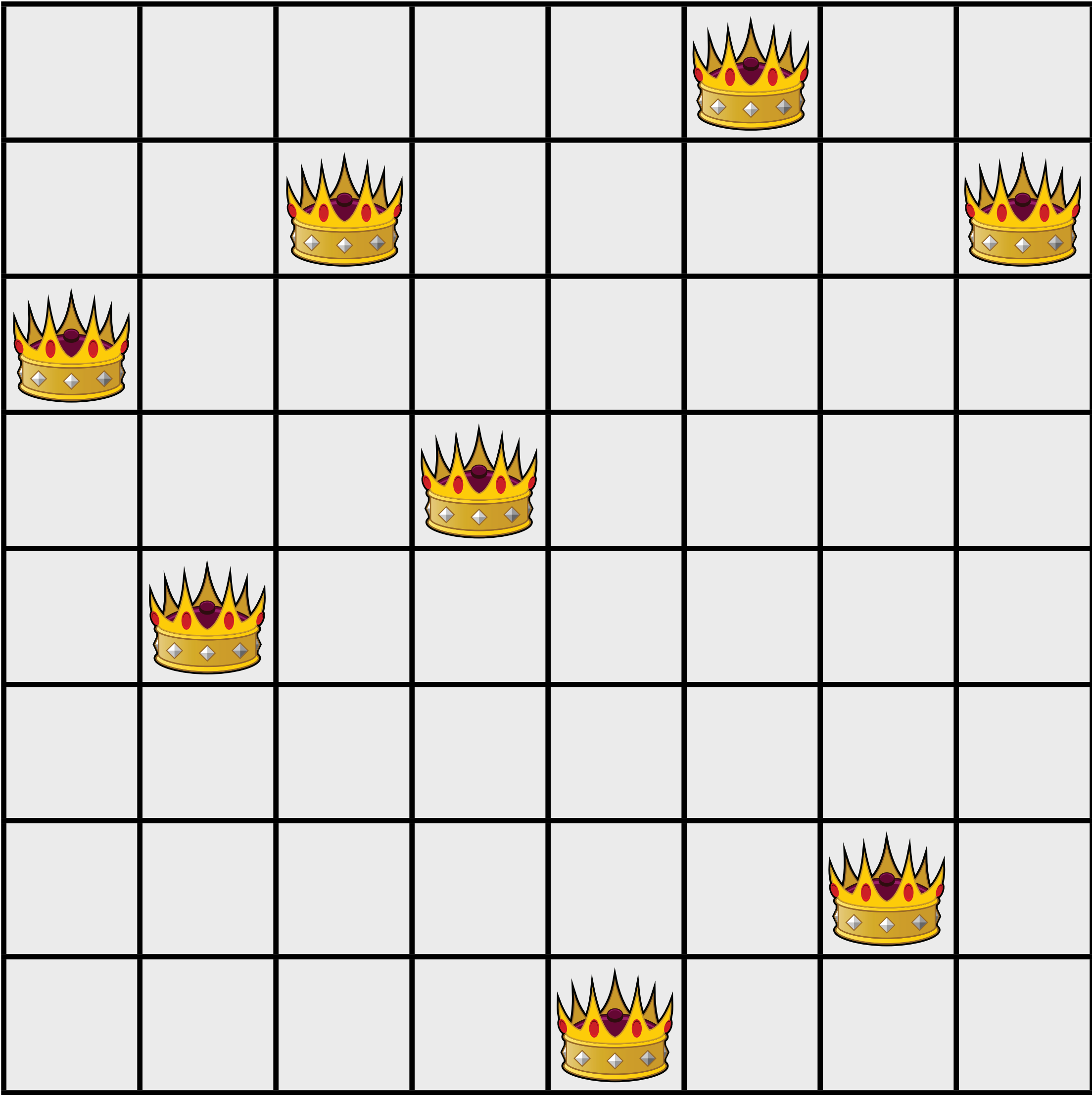
# Goals of the Lecture

- ▶ Local search
  - introduction
  - max/min-conflict search

# Local Search

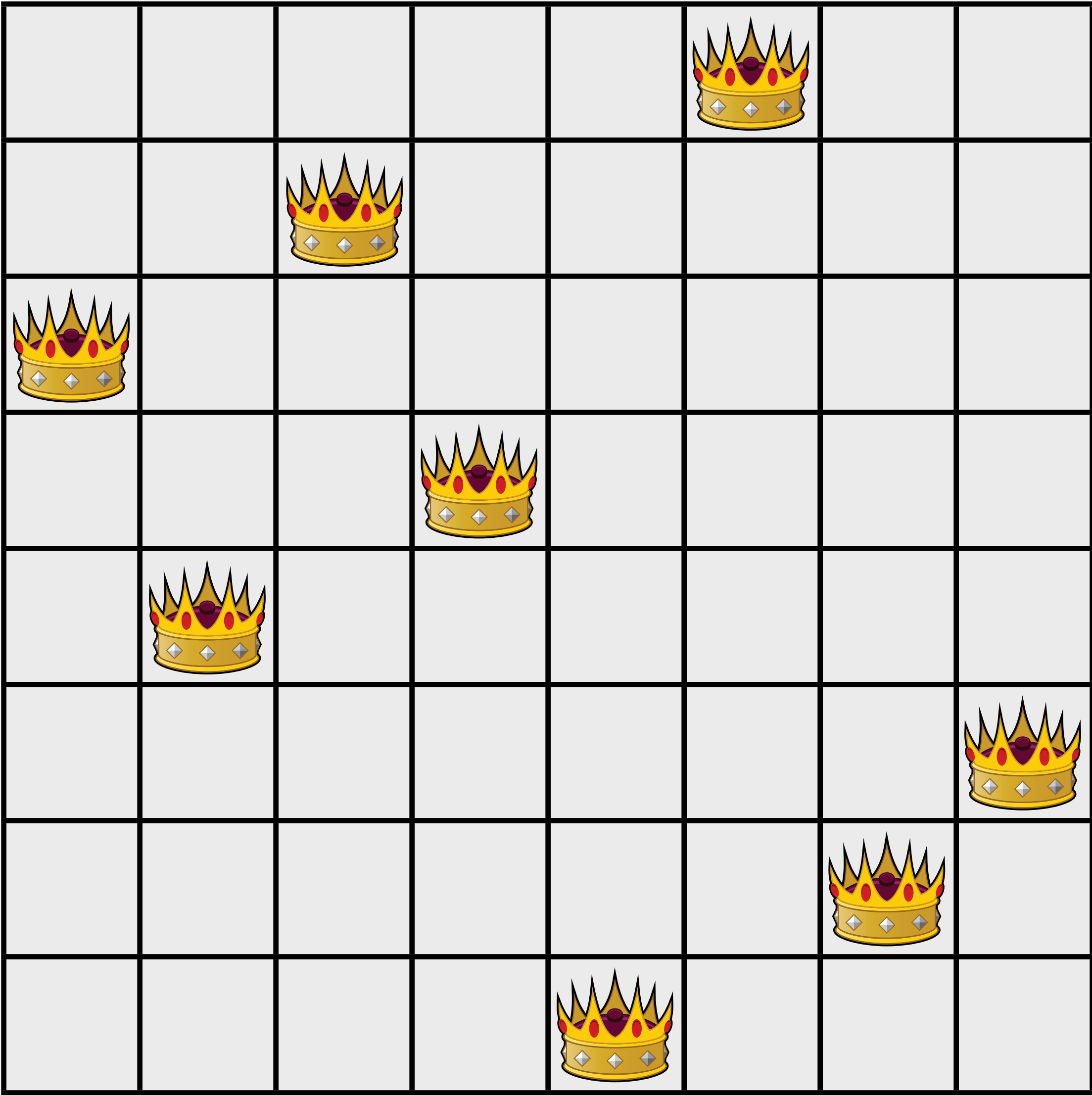


# Local Search

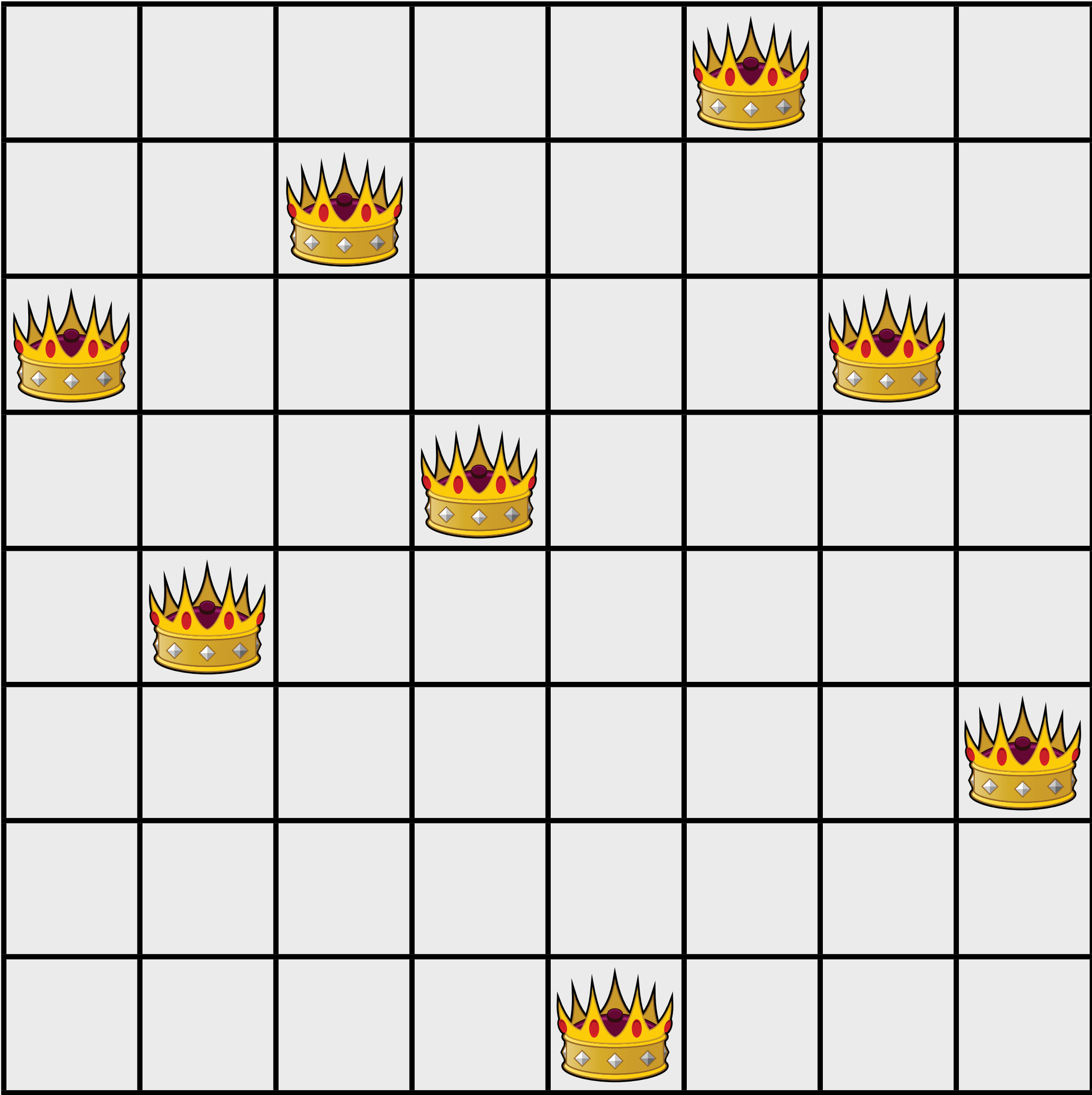




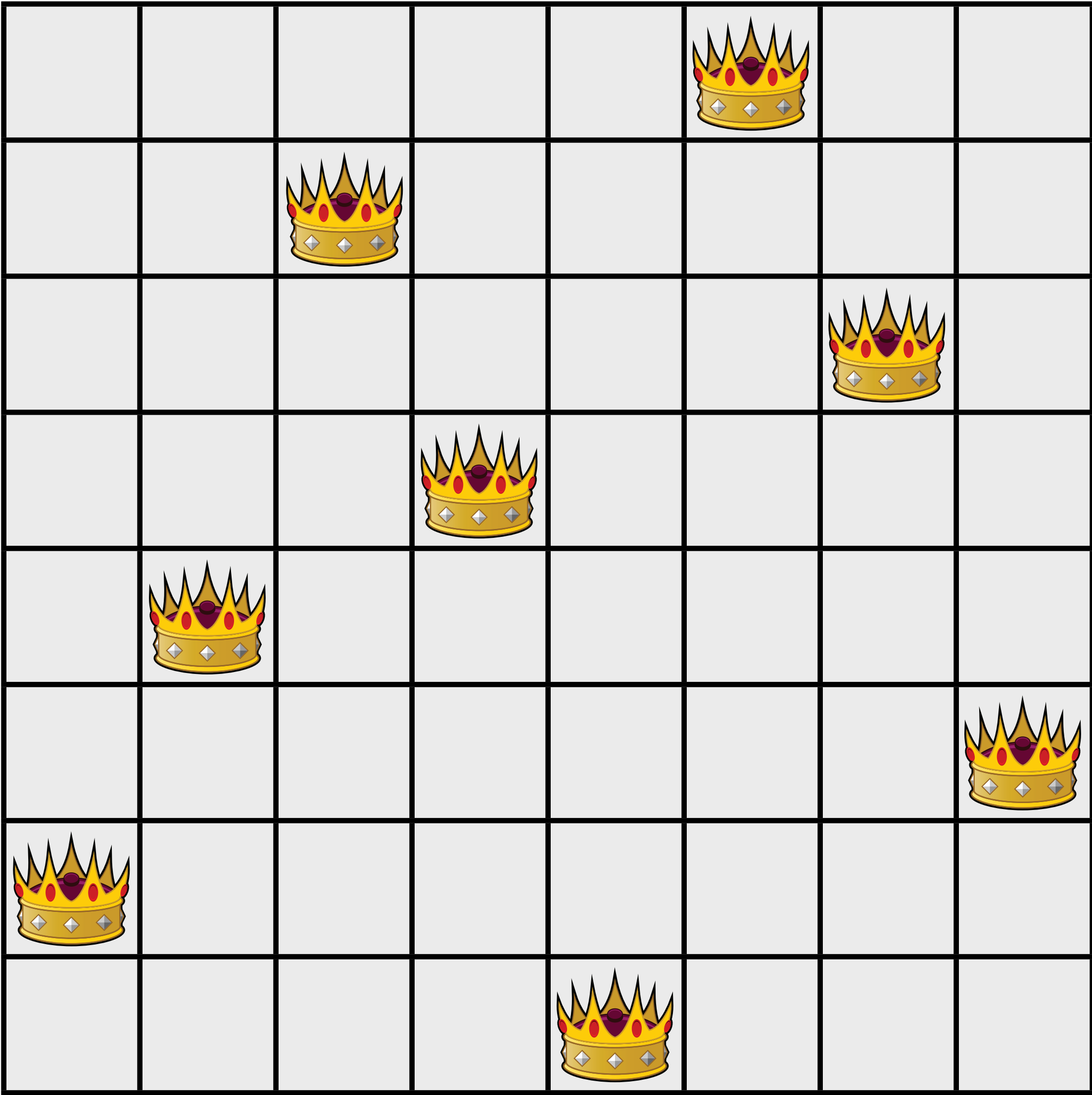
# Local Search



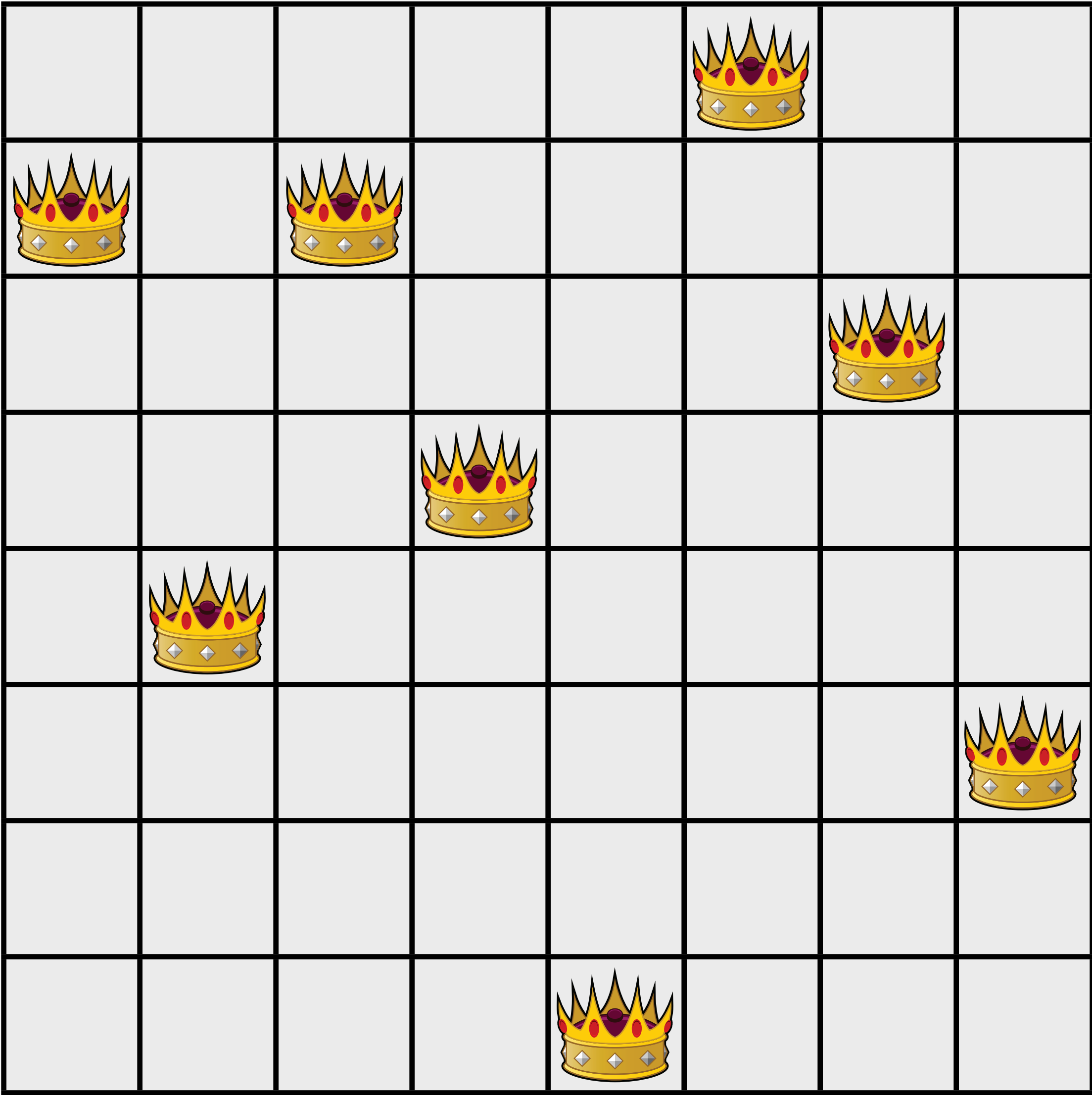
# Local Search



# Local Search

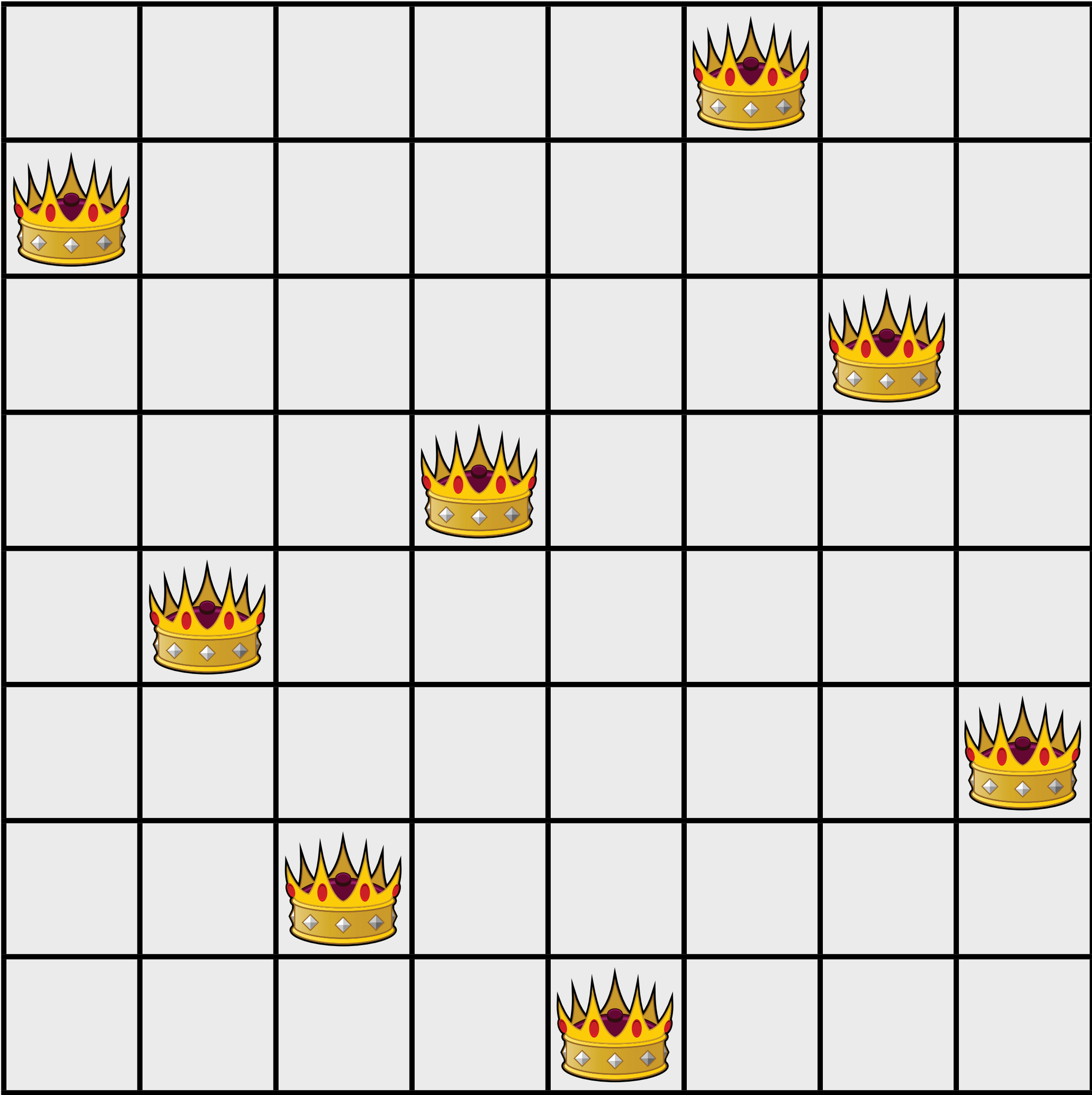


# Local Search





# Local Search



# Local Search

- ▶ Local search
  - move from configurations to configurations by performing local moves



# Local Search

- ▶ Local search
  - move from configurations to configurations by performing local moves
- ▶ Local search
  - works with complete assignments to the decision variables and modify them

# Local Search

- ▶ Local search
  - move from configurations to configurations by performing local moves
- ▶ Local search
  - works with complete assignments to the decision variables and modify them
- ▶ Contrast with constraint programming
  - CP works with partial assignments that are being extended

# Local Search

- ▶ How to drive a local search?

# Local Search

- ▶ How to drive a local search?
- ▶ Satisfaction problems
  - start with an infeasible configuration
  - move towards feasibility

# Local Search

- ▶ How to drive a local search?
- ▶ Satisfaction problems
  - start with an infeasible configuration
  - move towards feasibility
- ▶ Pure optimization problems
  - start with suboptimal solutions
  - move towards optimal configurations



# Local Search

- ▶ How to drive a local search?
- ▶ Satisfaction problems
  - start with an infeasible configuration
  - move towards feasibility
- ▶ Pure optimization problems
  - start with suboptimal solutions
  - move towards optimal configurations
- ▶ Constrained optimization problems
  - many options ... (TBDL)



# Local Search for Satisfaction Problems

- ▶ How to drive the search toward feasibility?
  - transform the problem into an optimization problem

# Local Search for Satisfaction Problems

- ▶ How to drive the search toward feasibility?
  - transform the problem into an optimization problem
- ▶ Use the concept of violations
  - e.g., how many constraints are violated by a configuration

# Local Search for Satisfaction Problems

- ▶ How to drive the search toward feasibility?
  - transform the problem into an optimization problem
- ▶ Use the concept of violations
  - e.g., how many constraints are violated by a configuration
- ▶ Minimize violations
  - move toward configurations with fewer violations

# Local Search for Satisfaction Problems

- ▶ What is a local move?
  - many choices
  - assign a value to a decision variable

# Local Search for Satisfaction Problems

- ▶ What is a local move?
  - many choices
  - assign a value to a decision variable
- ▶ How to select a move?
  - many choices
  - max/min conflict



# Local Search for Satisfaction Problems

- ▶ What is a local move?
  - many choices
  - assign a value to a decision variable
- ▶ How to select a move?
  - many choices
  - max/min conflict
- ▶ Max/Min-Conflict
  - choose a decision variable that appear in the most violations
  - assign to a value that minimizes its violations



# The Queens Model

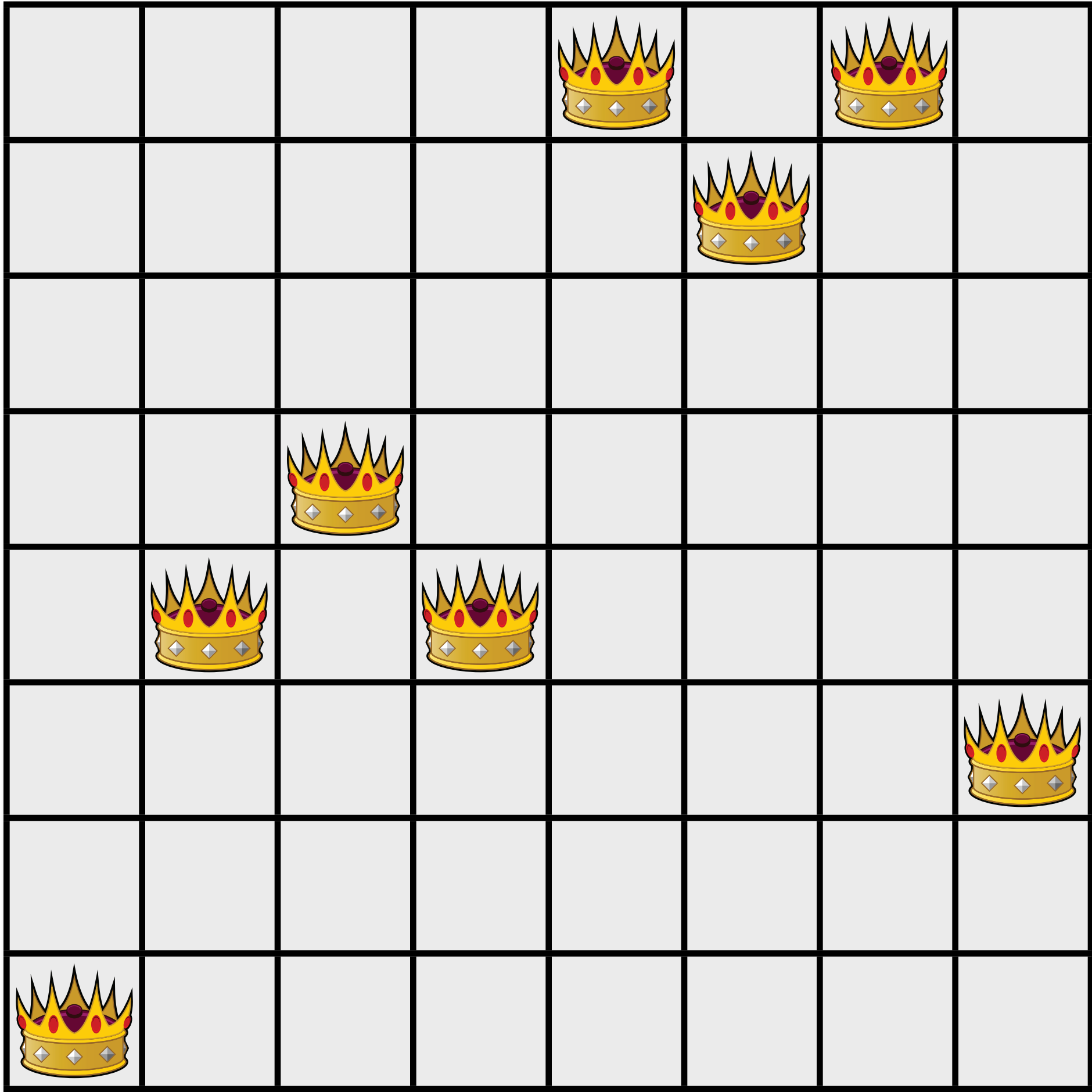
- ▶ Constraint violations

# The Queens Model

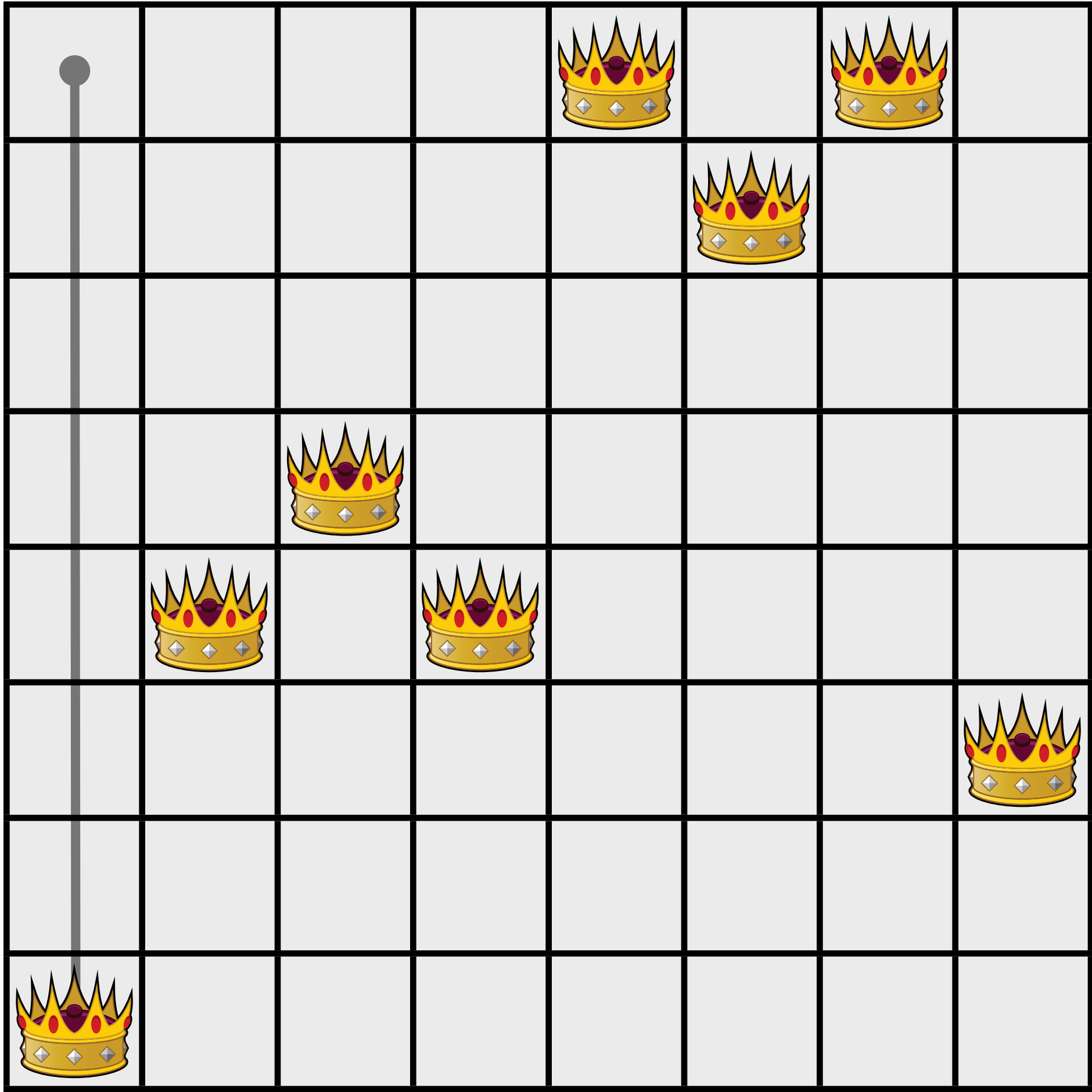
## ► Constraint violations

```
range R = 1..8;  
var{int} row[R] in R;  
solve {  
    forall(i in R, j in R: i < j) {  
        row[i] ≠ row[j];  
        row[i] ≠ row[j] + (j - i);  
        row[i] ≠ row[j] - (j - i);  
    }  
}
```

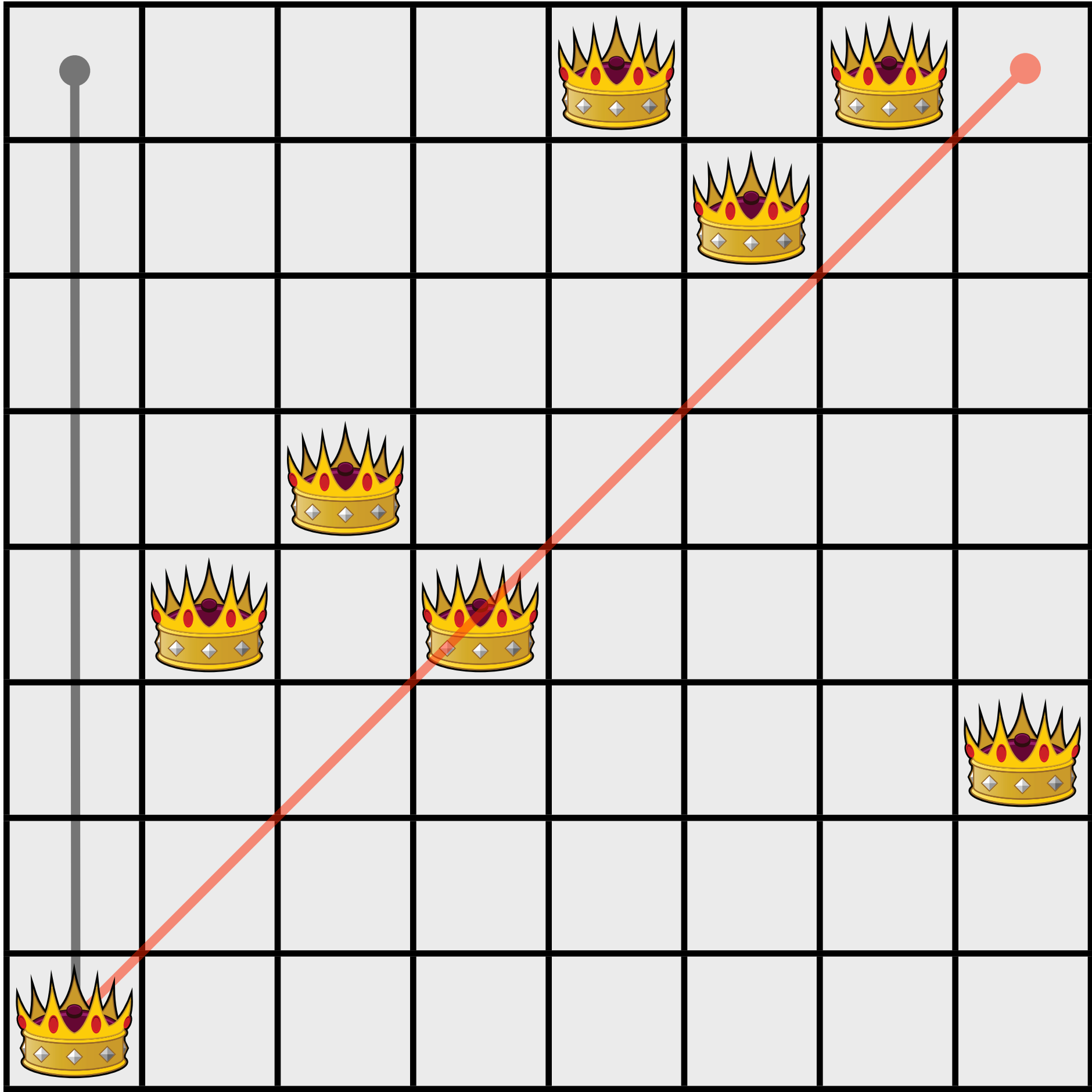
# Intuition



# Intuition

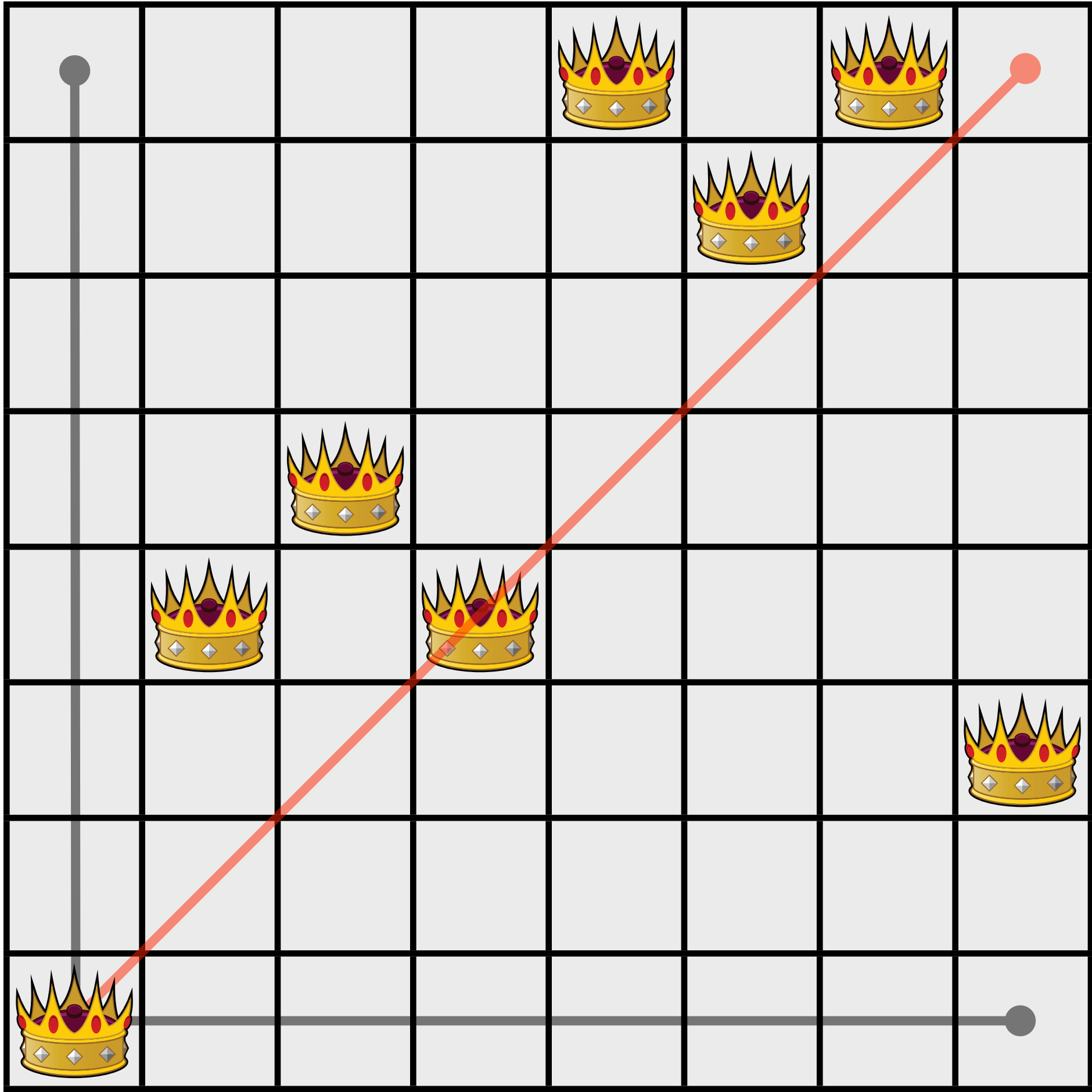


# Intuition



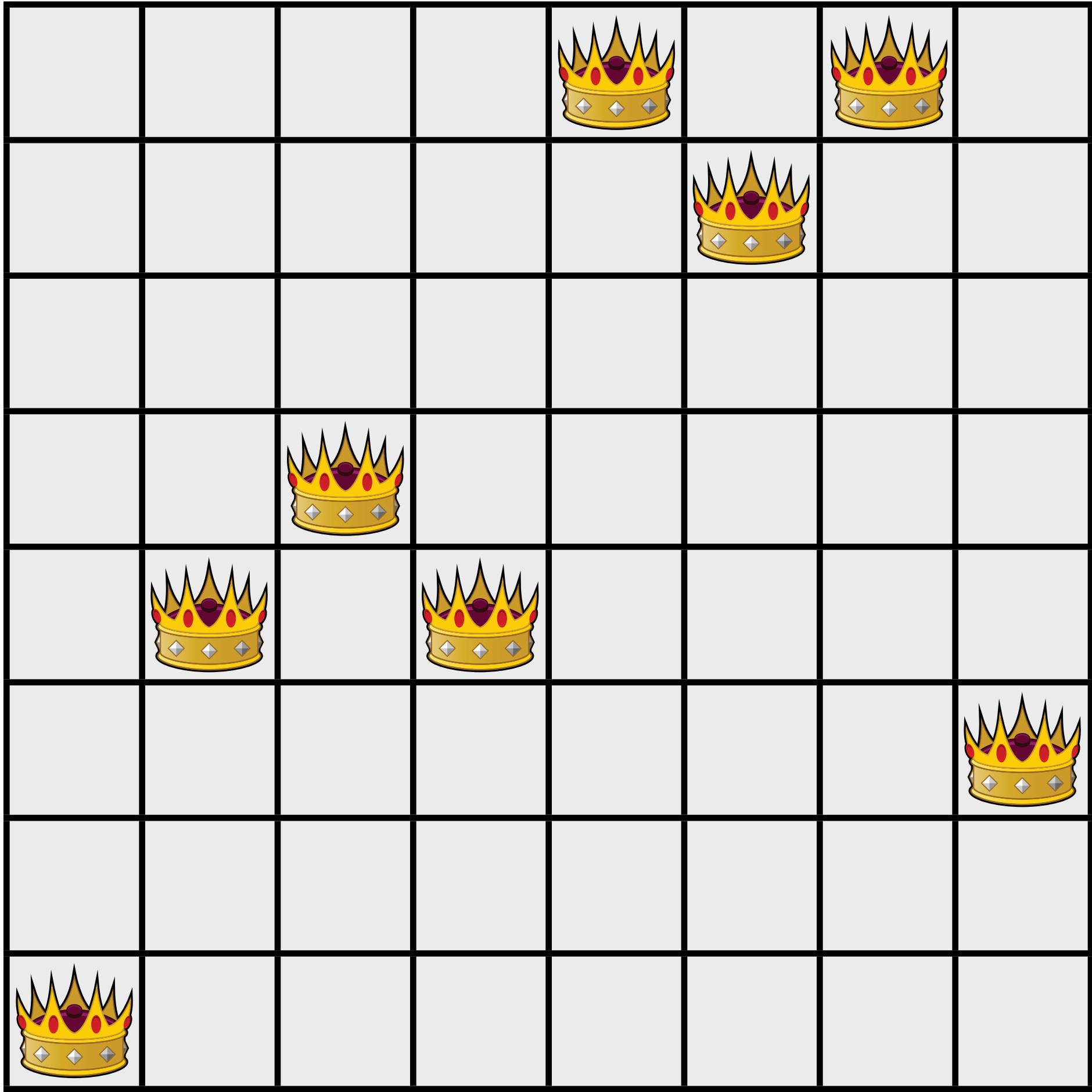


# Intuition



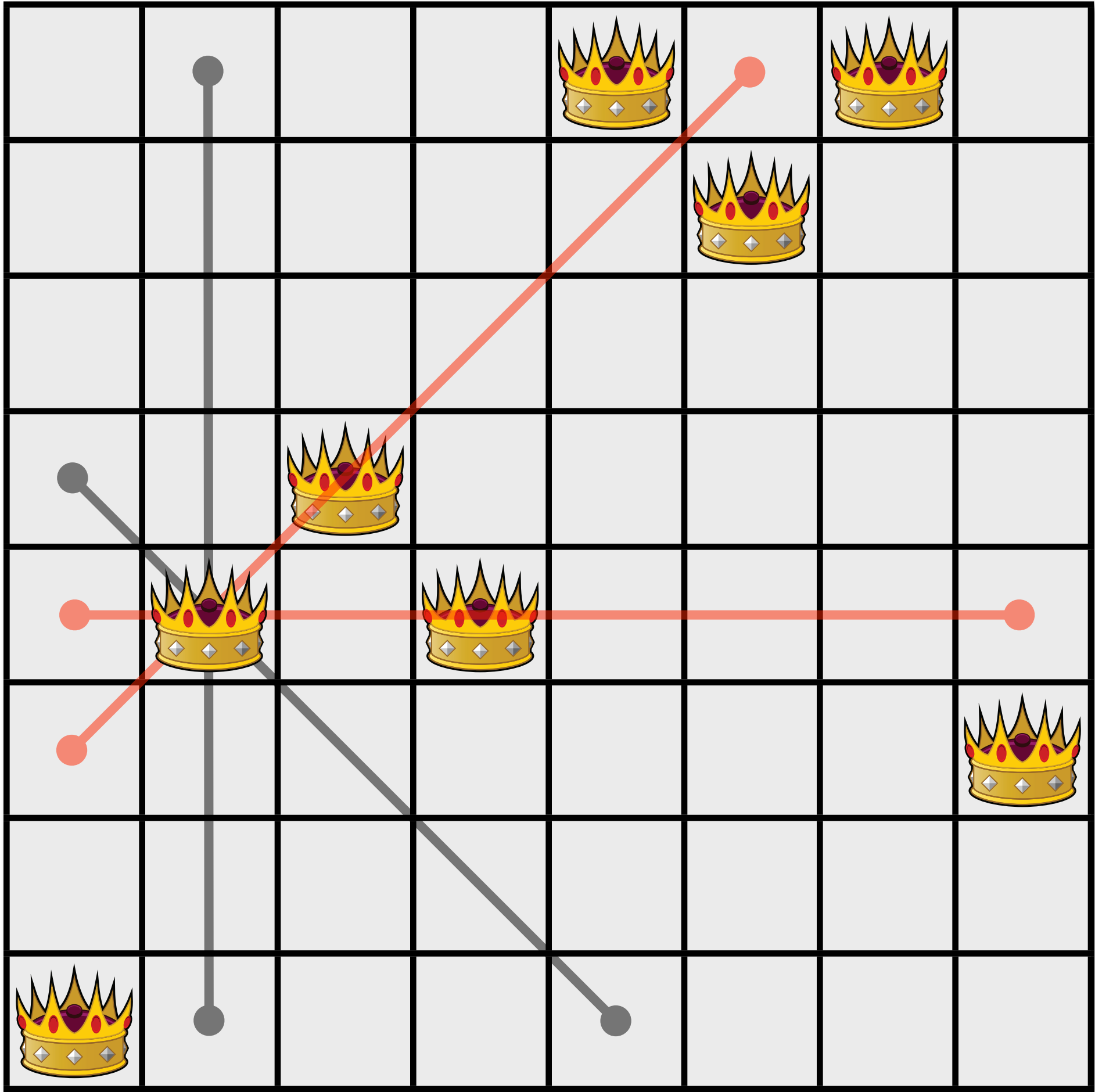


# Intuition



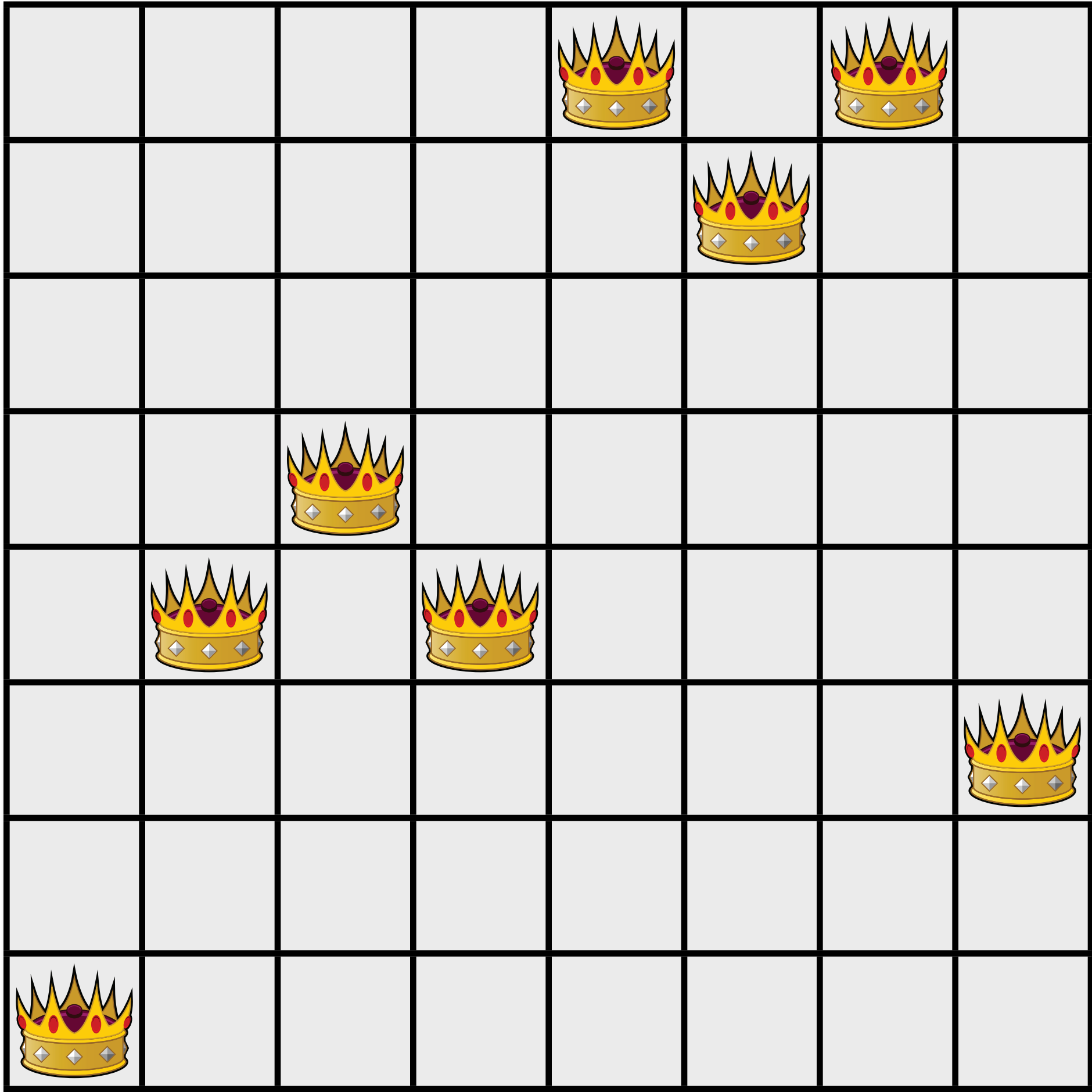
1

# Intuition



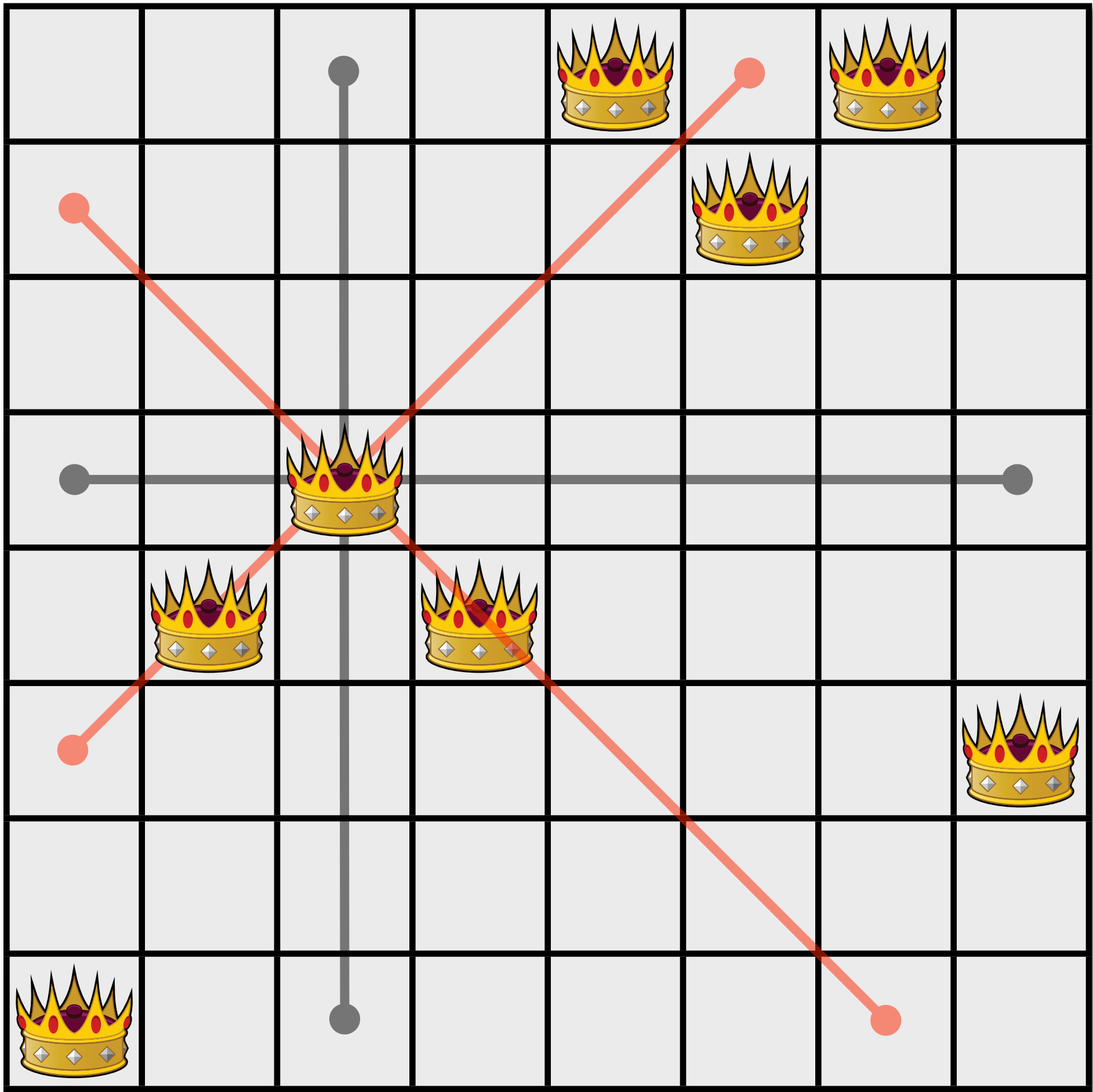
1

# Intuition



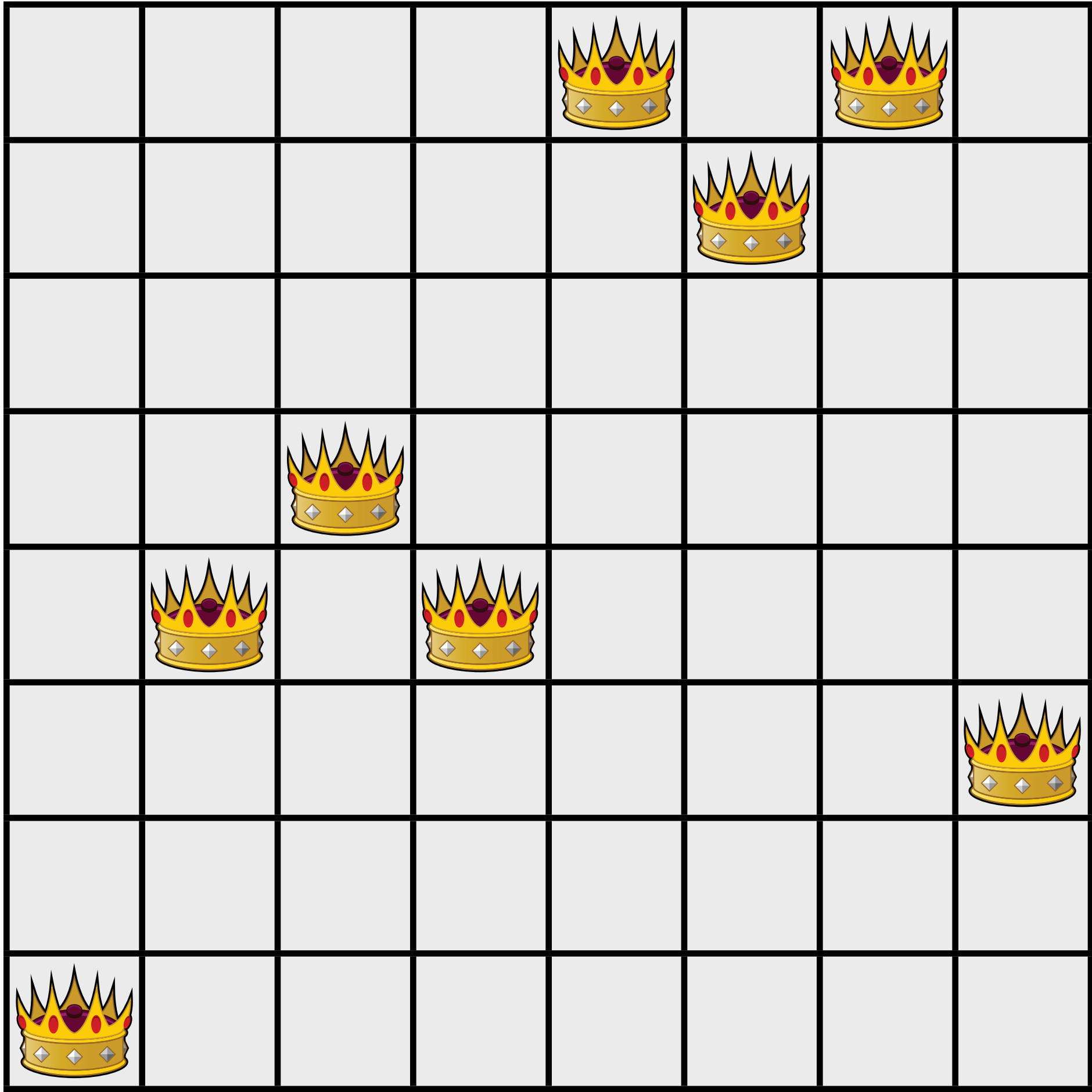
1 2

# Intuition



1 2

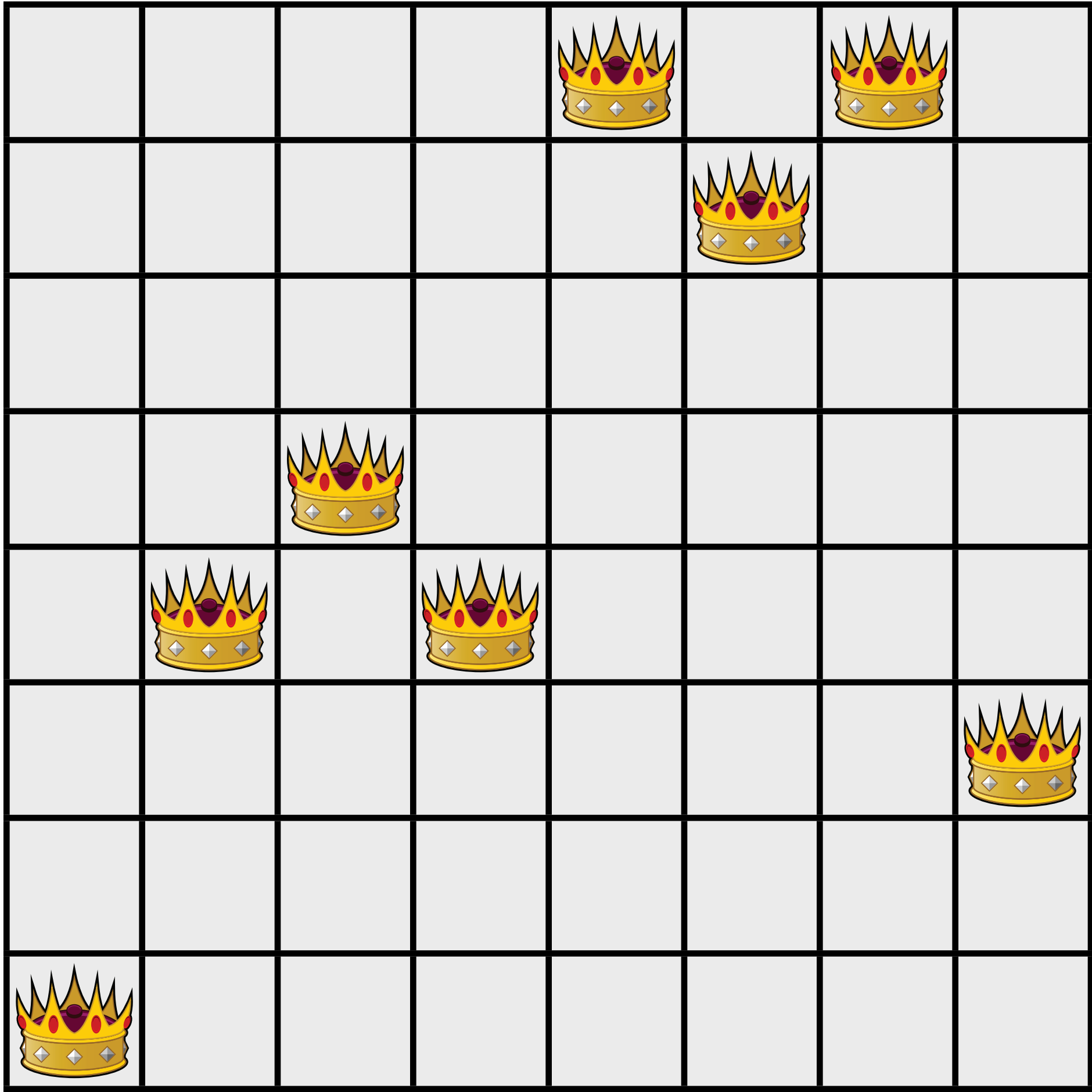
# Intuition



1   2   2

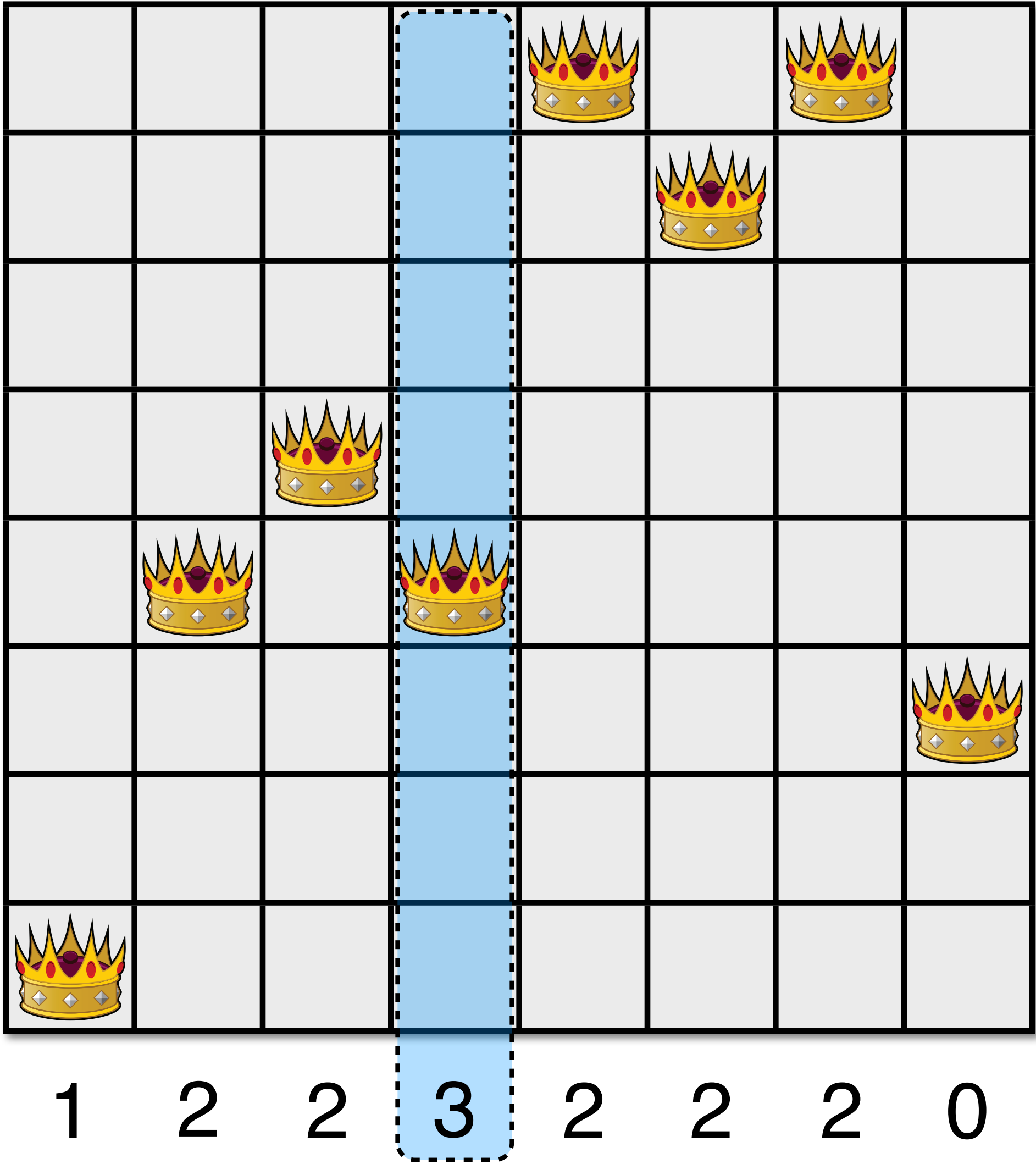


# Intuition

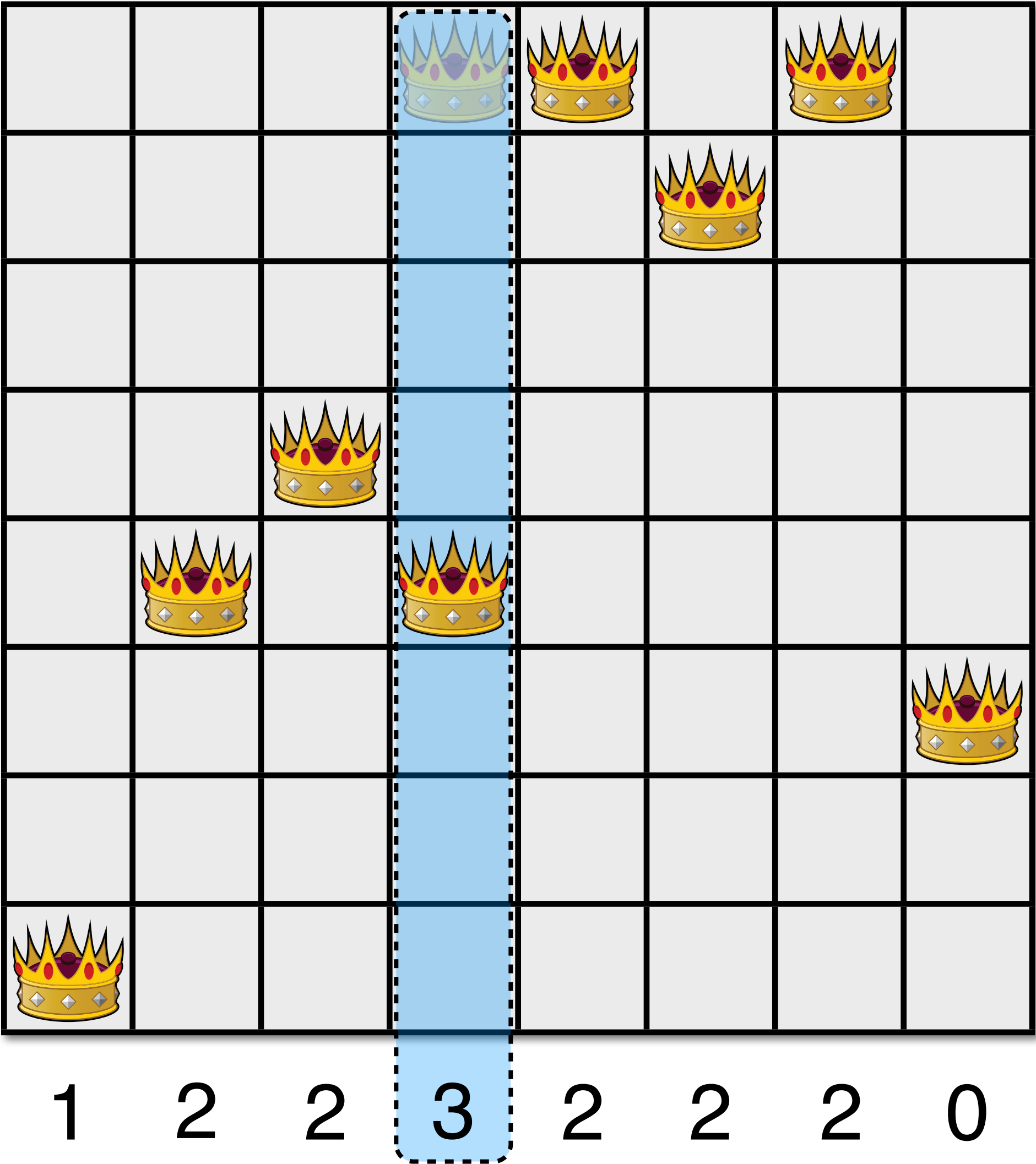


1 2 2 3 2 2 2 0 7

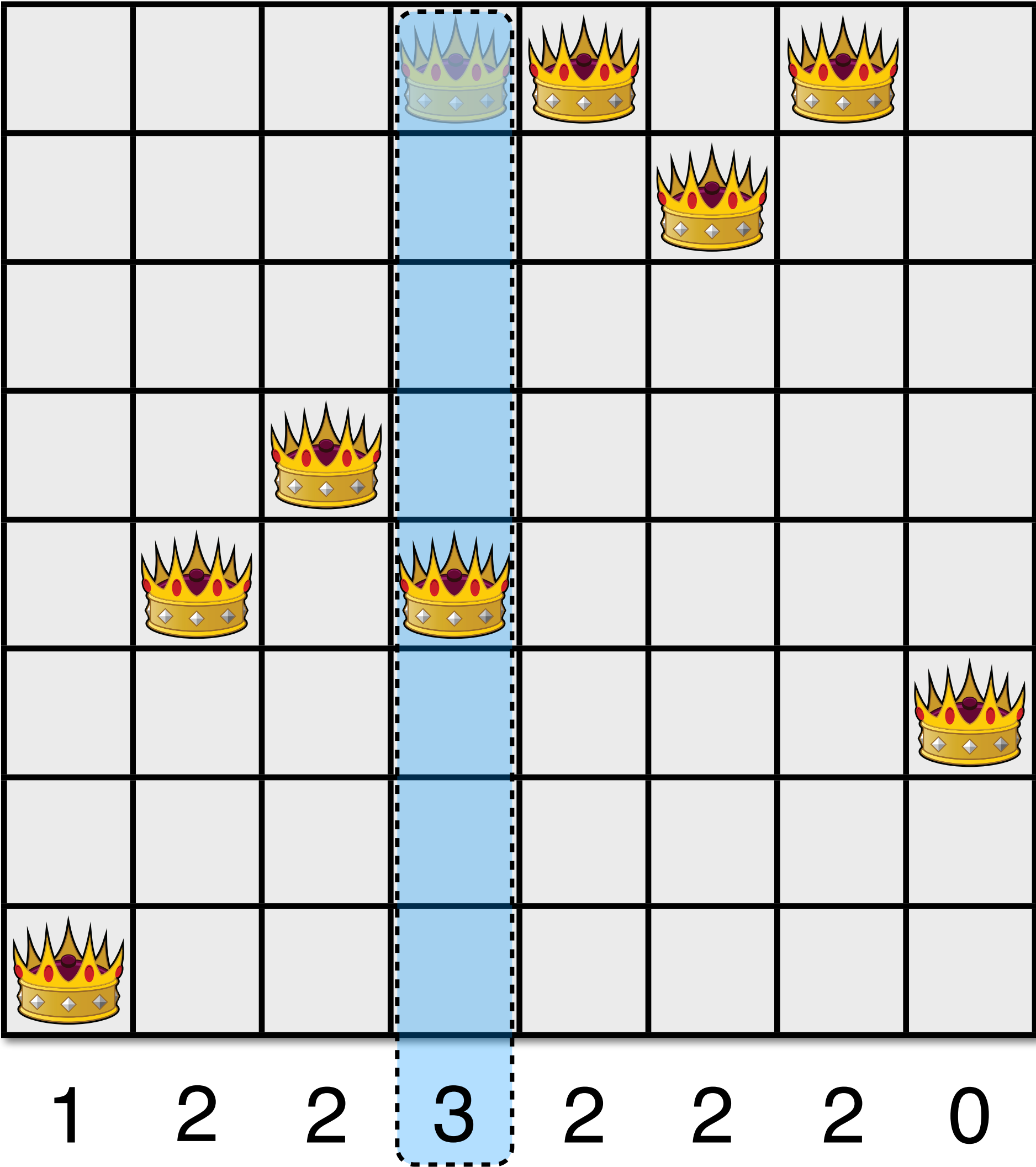
# Intuition



# Intuition



# Intuition

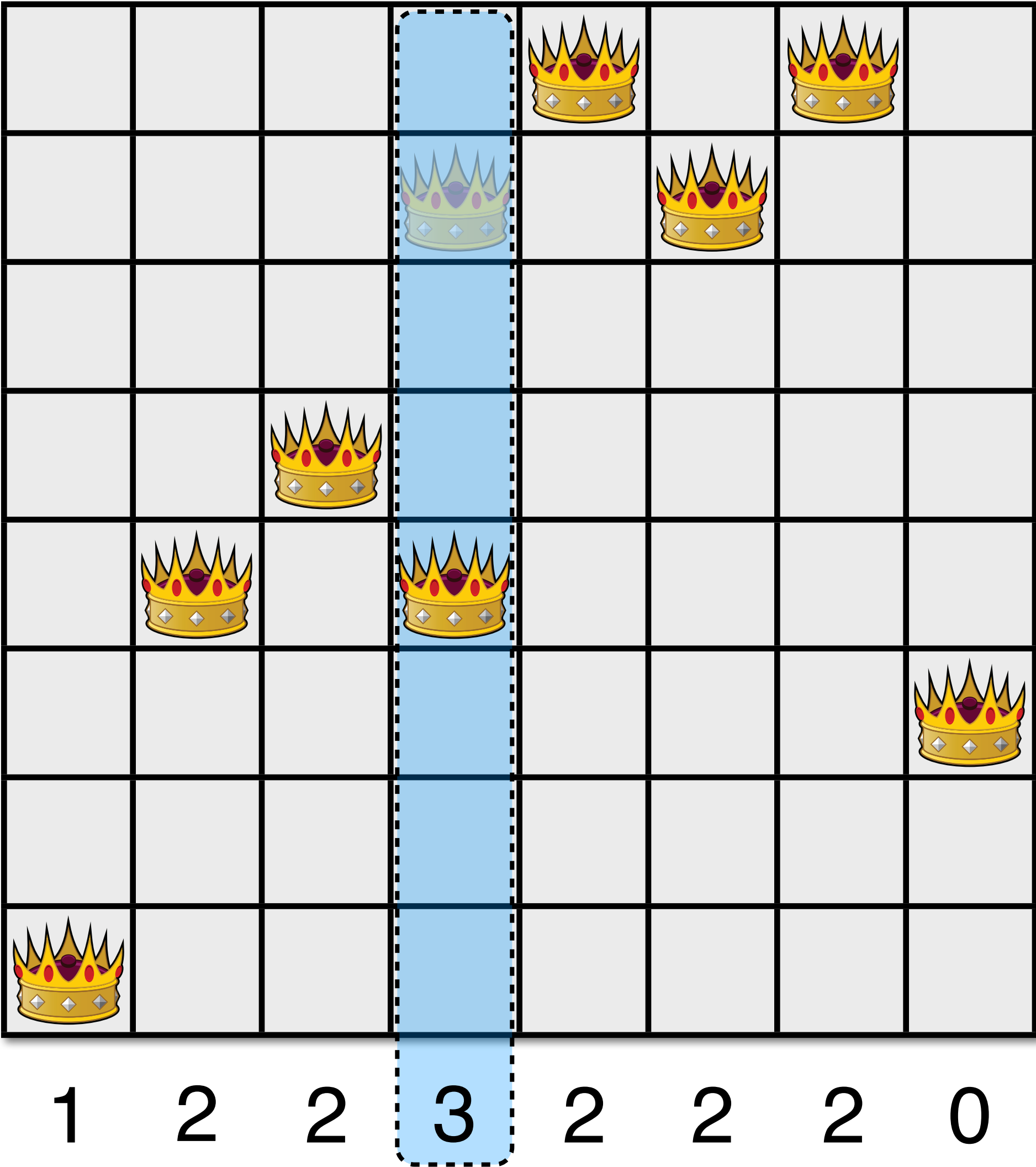


-1

7



# Intuition



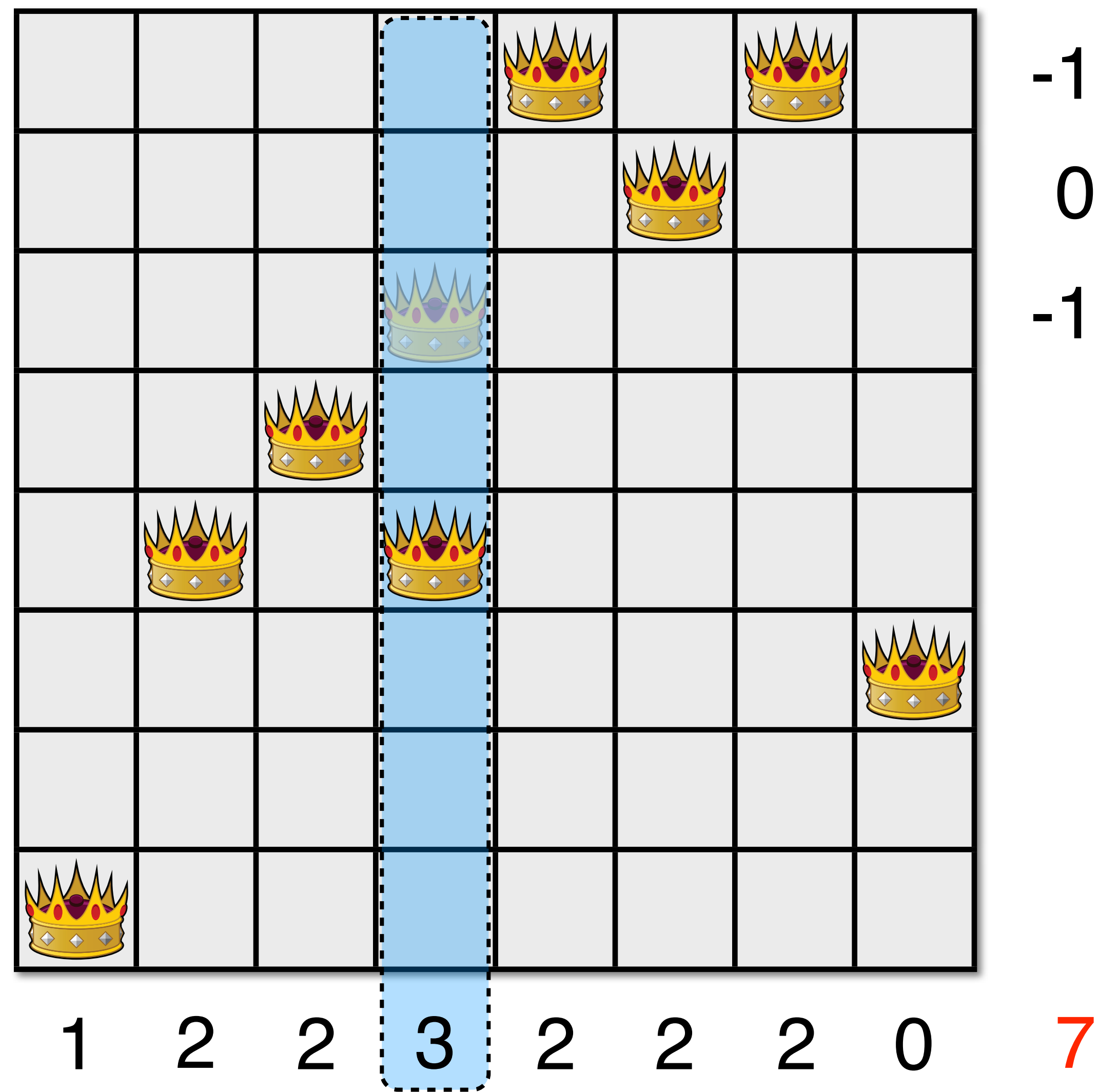
-1

0

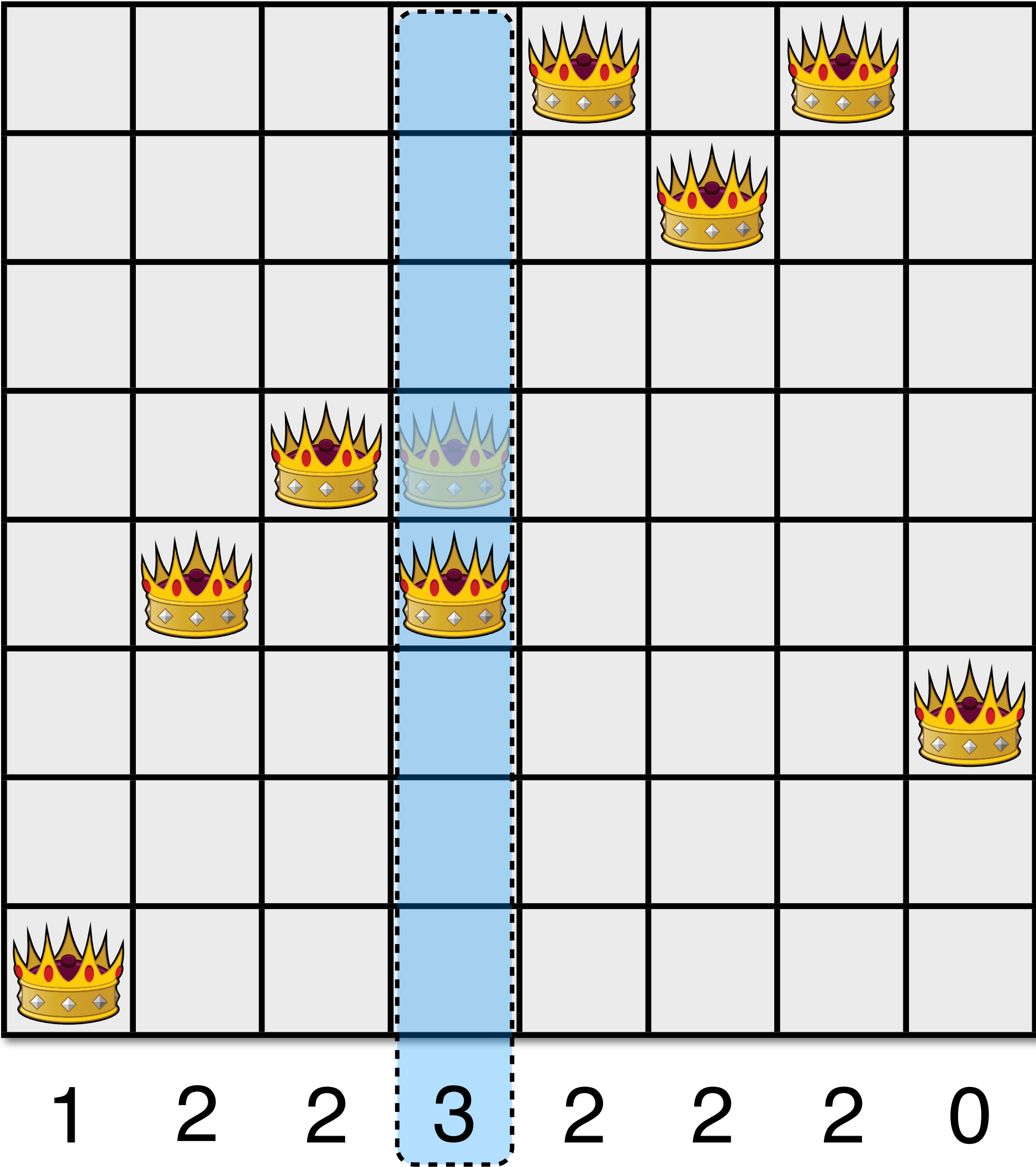
7



# Intuition



# Intuition



-1

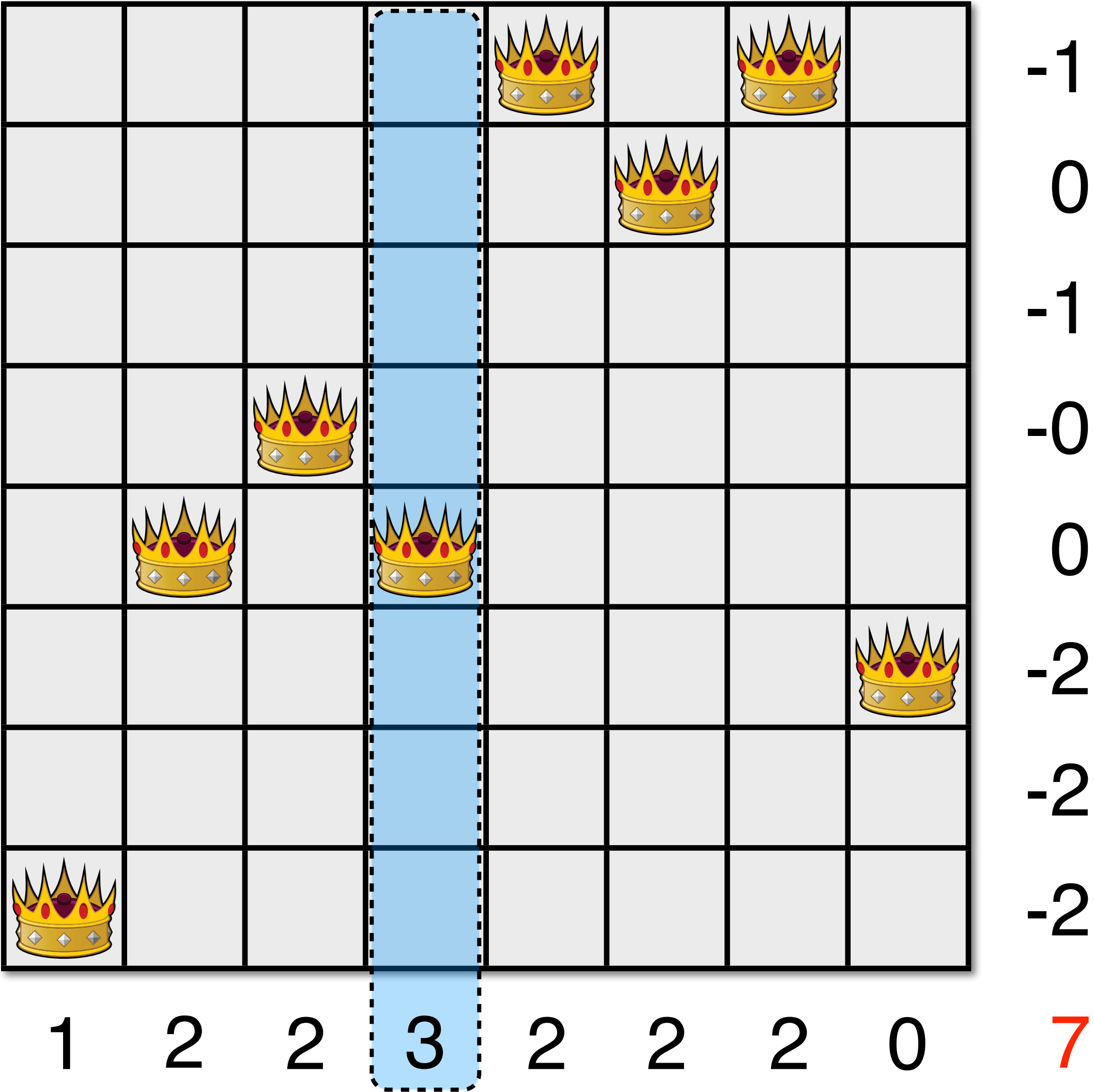
0

-1

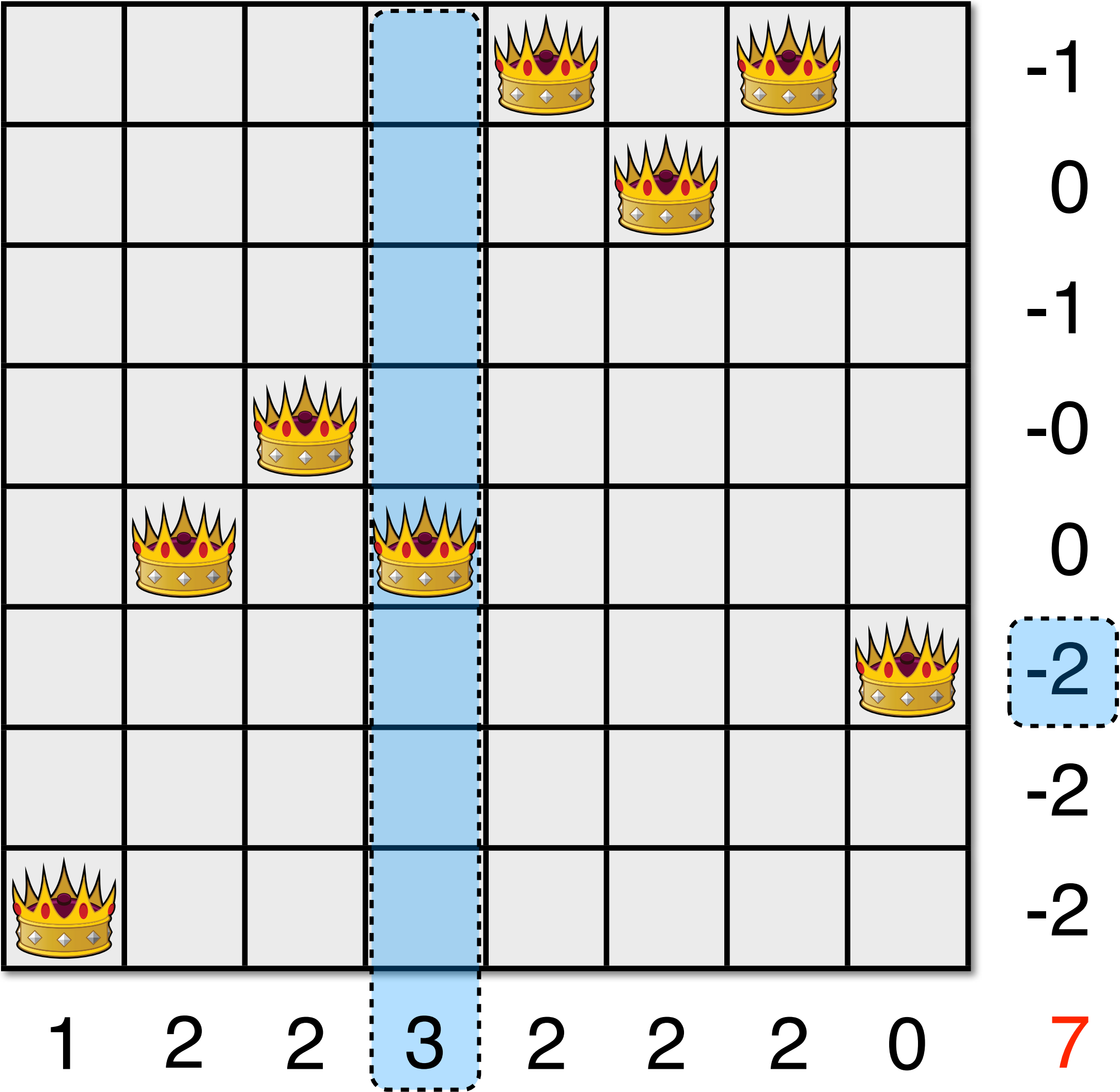
-0

7

# Intuition

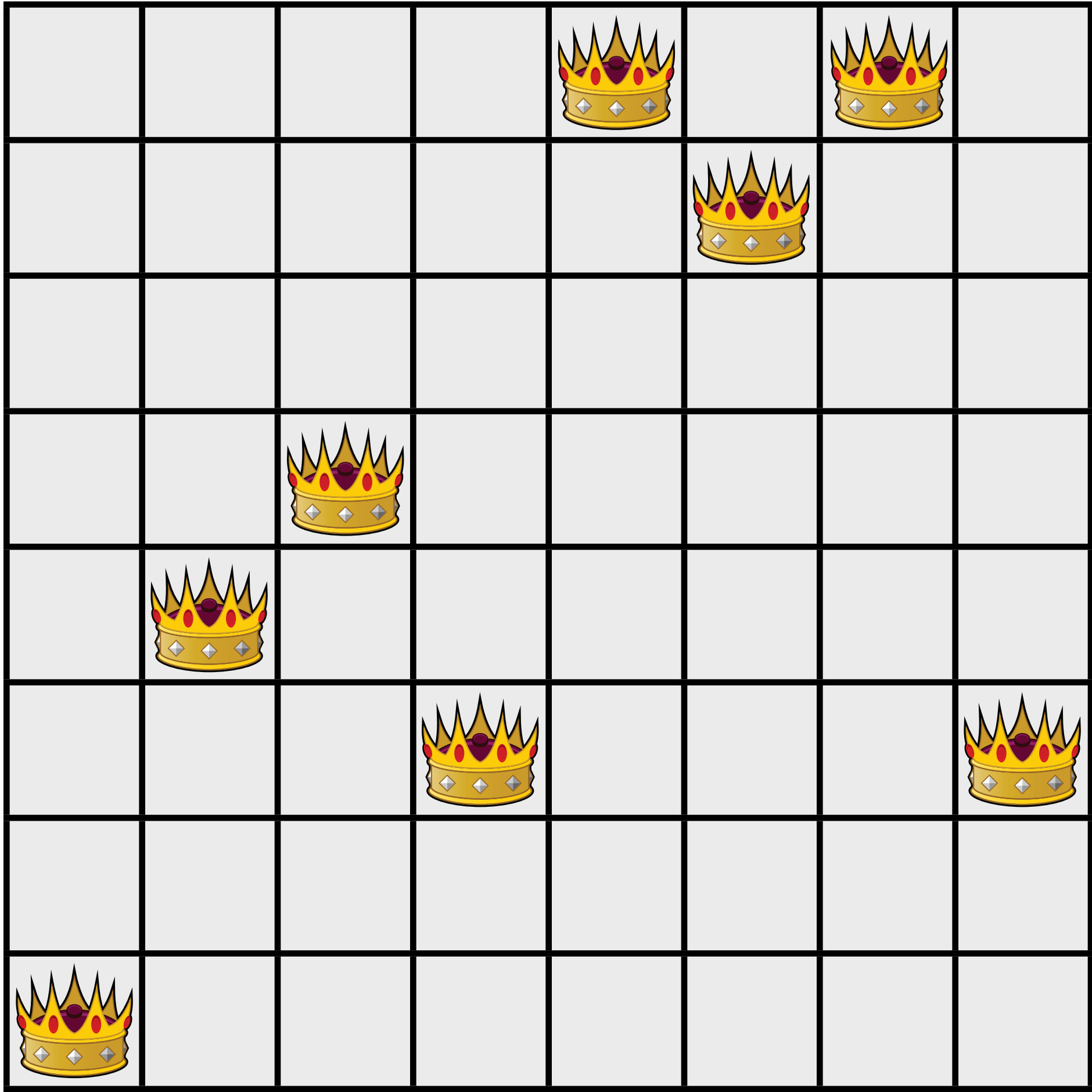


# Intuition



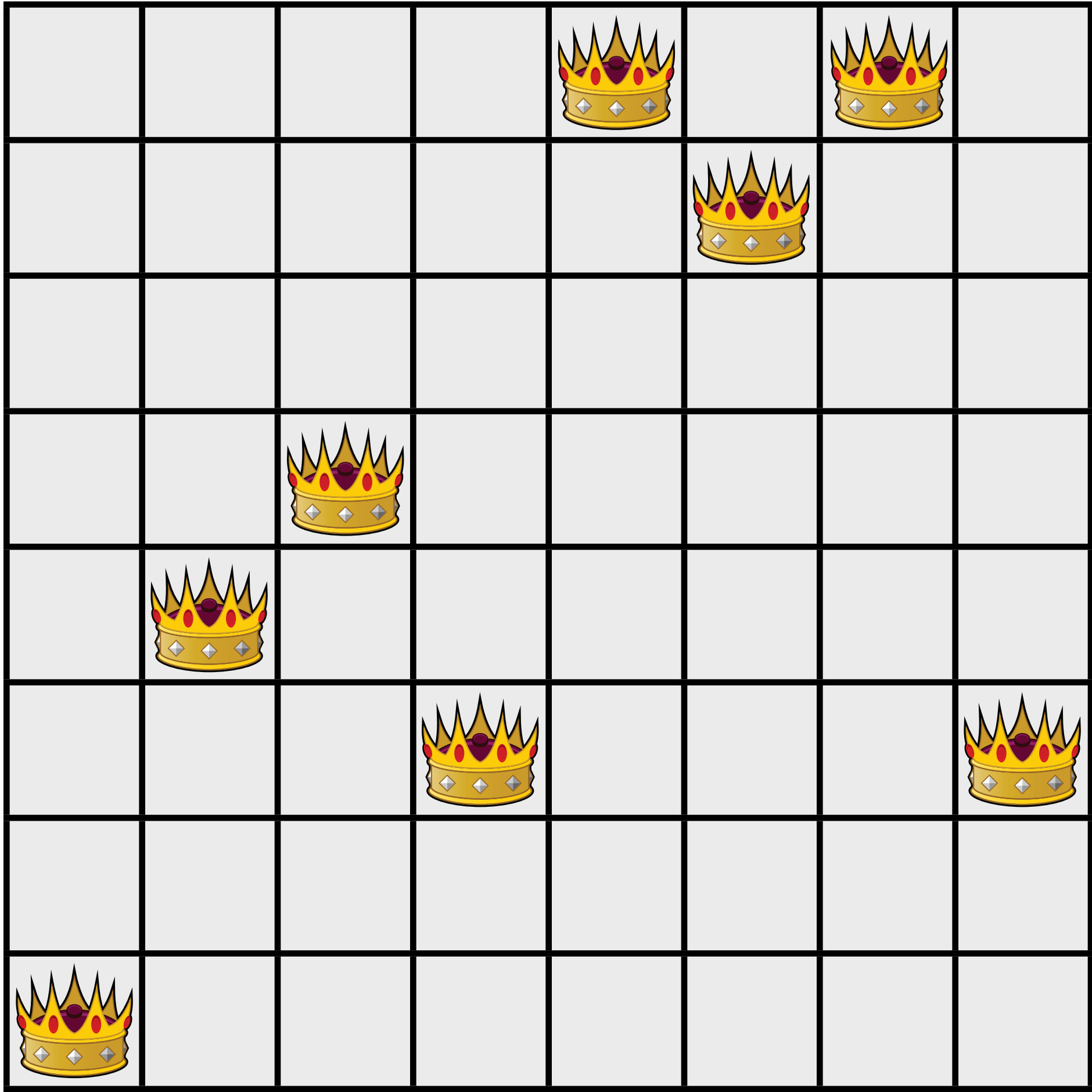


# Intuition

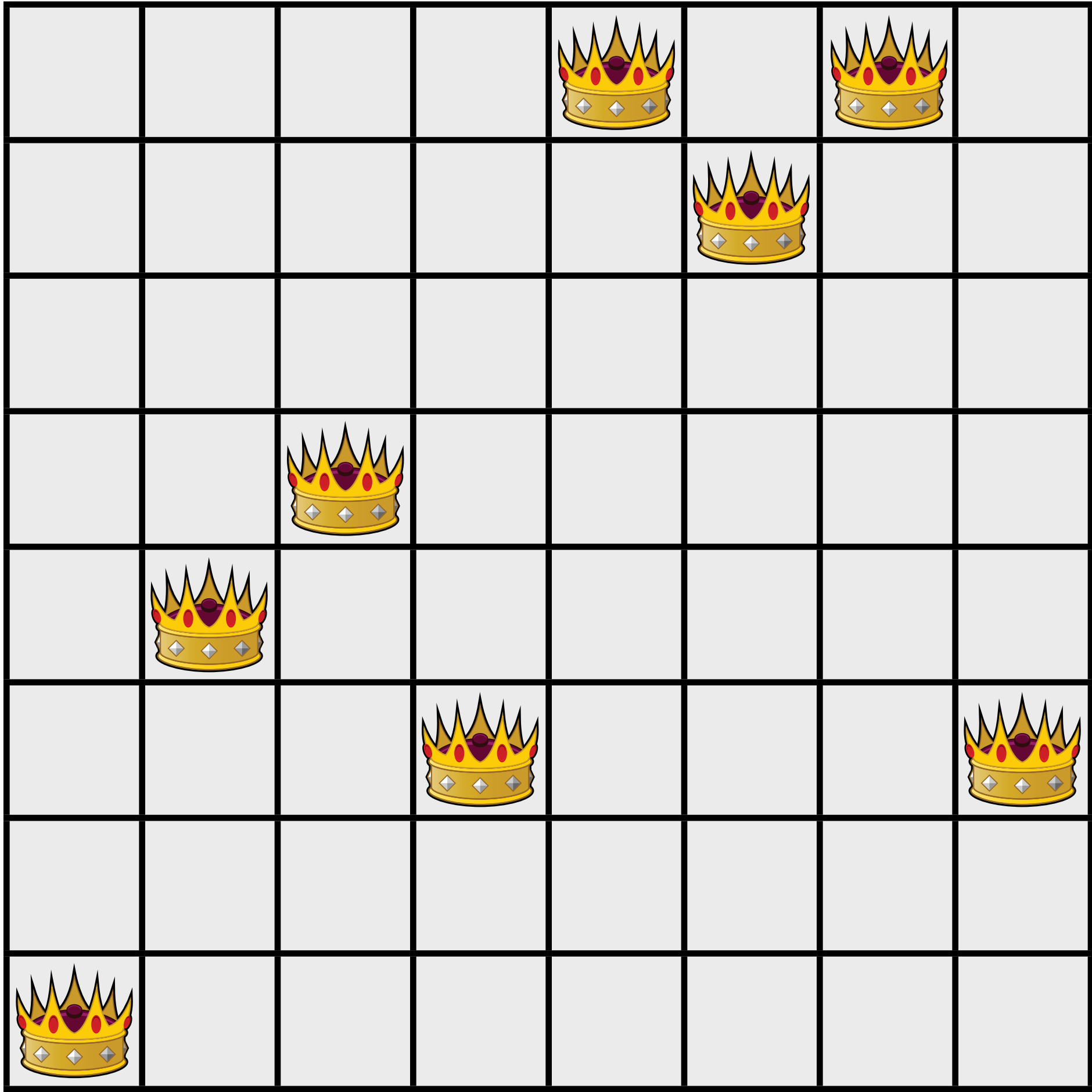




# Intuition

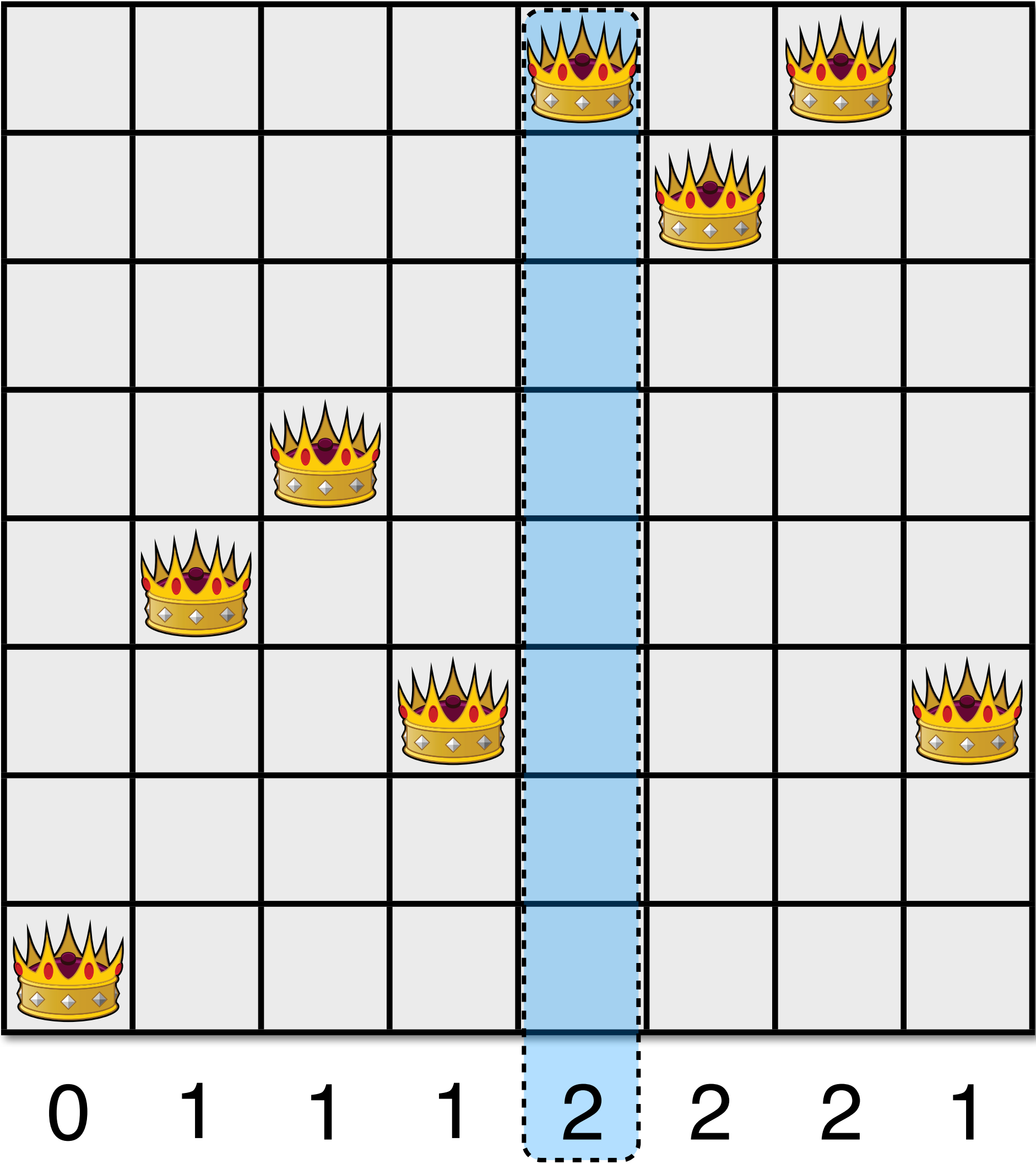


# Intuition

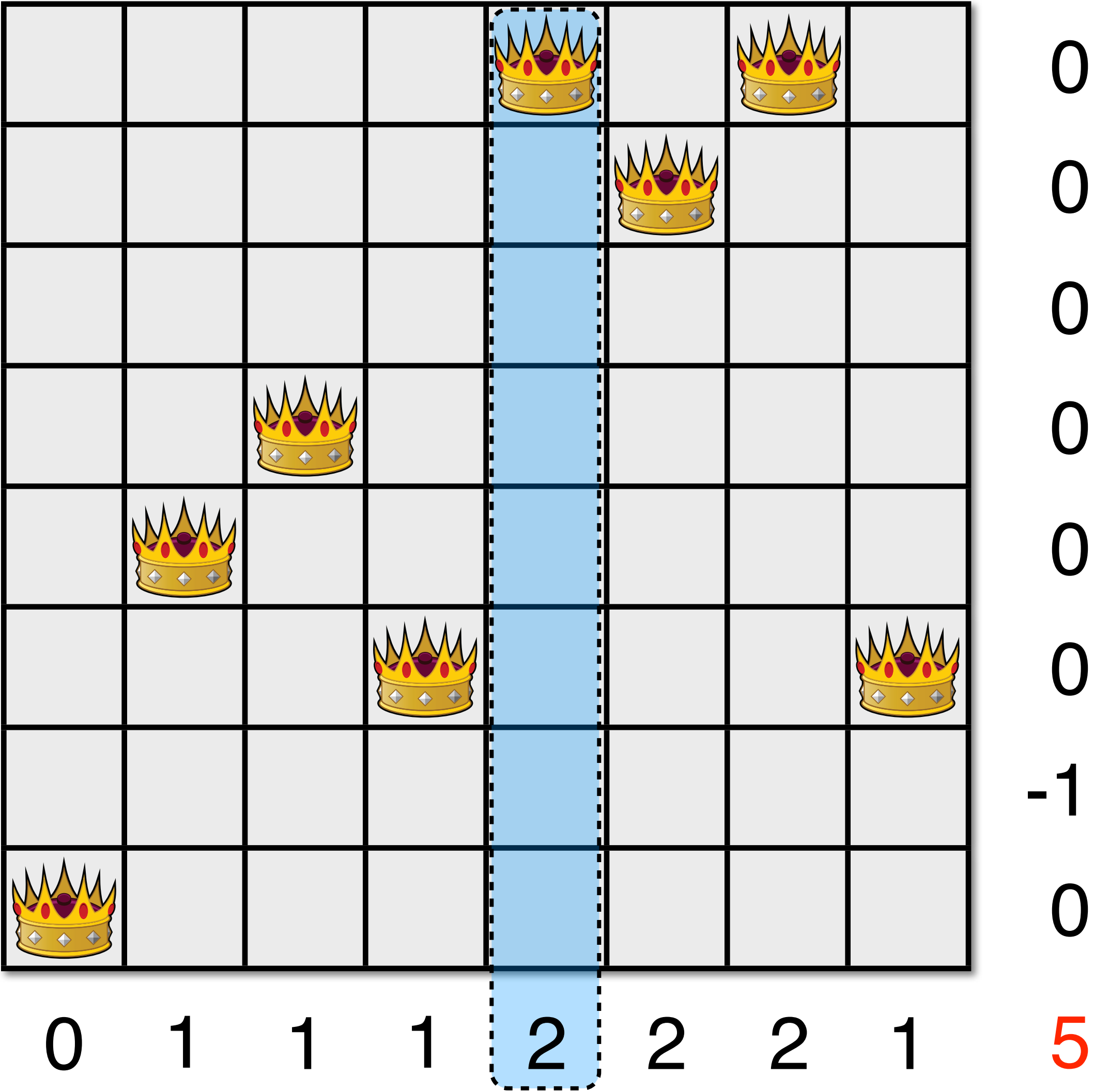


0 1 1 1 2 2 2 1 5

# Intuition

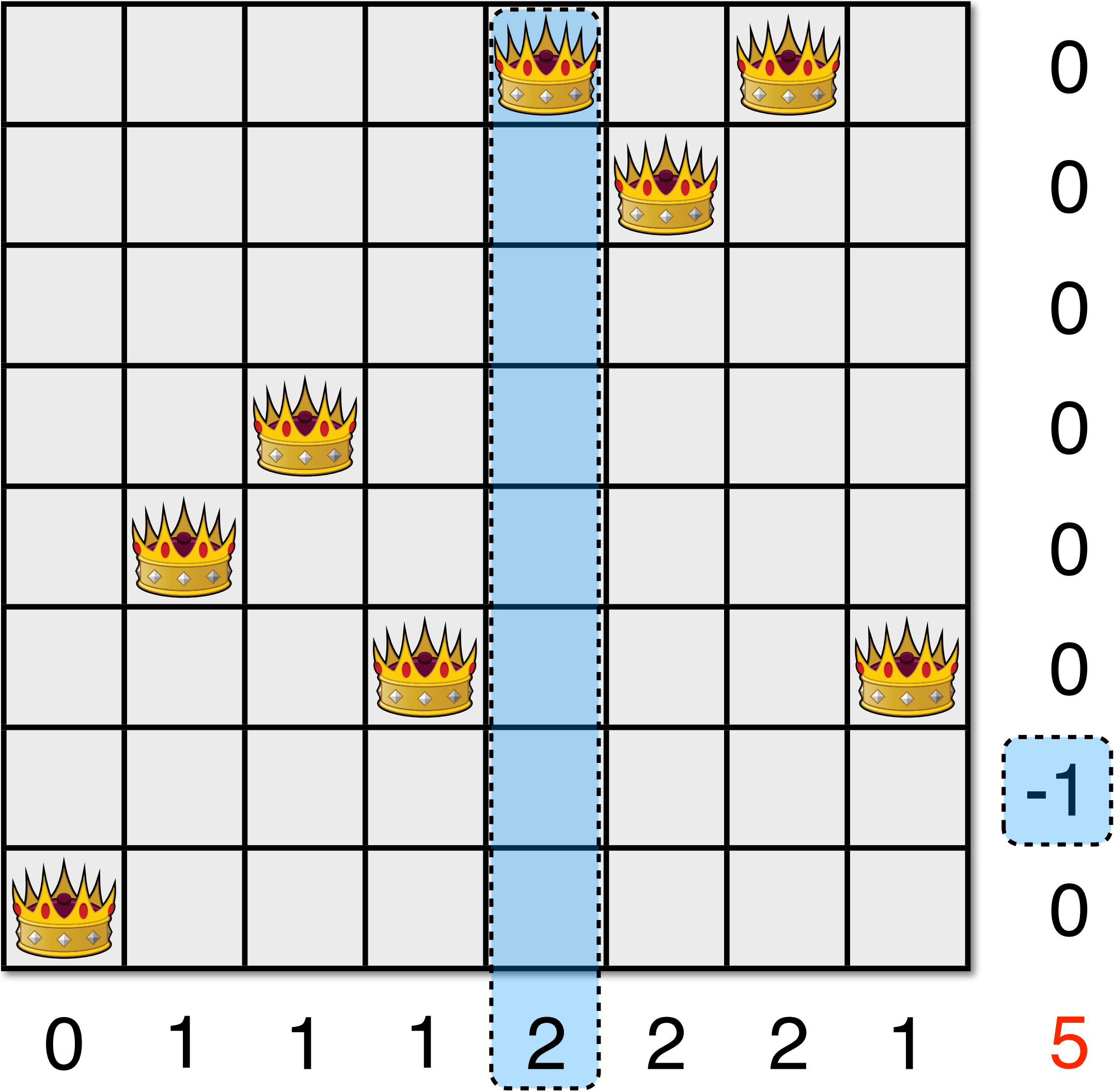


# Intuition



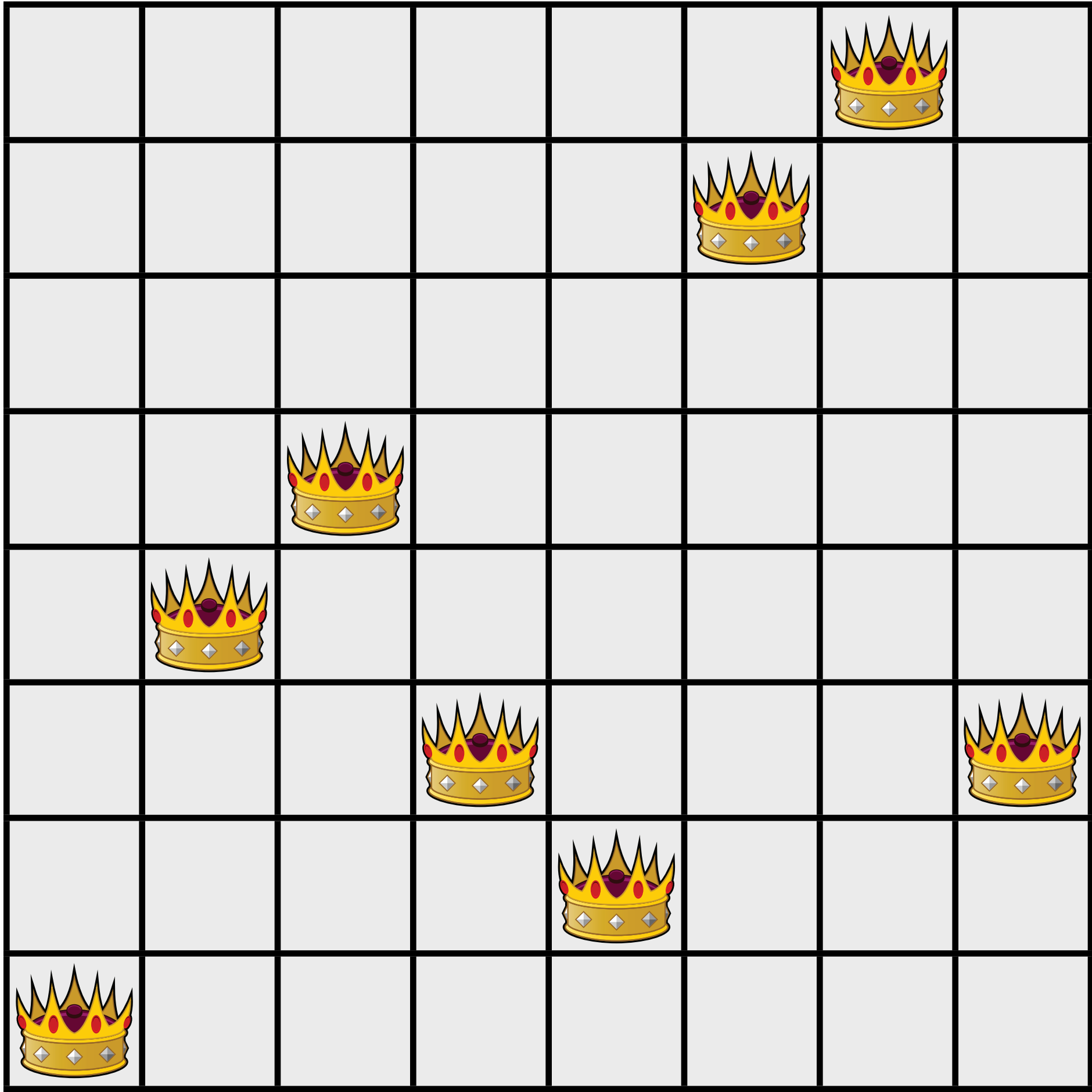


# Intuition

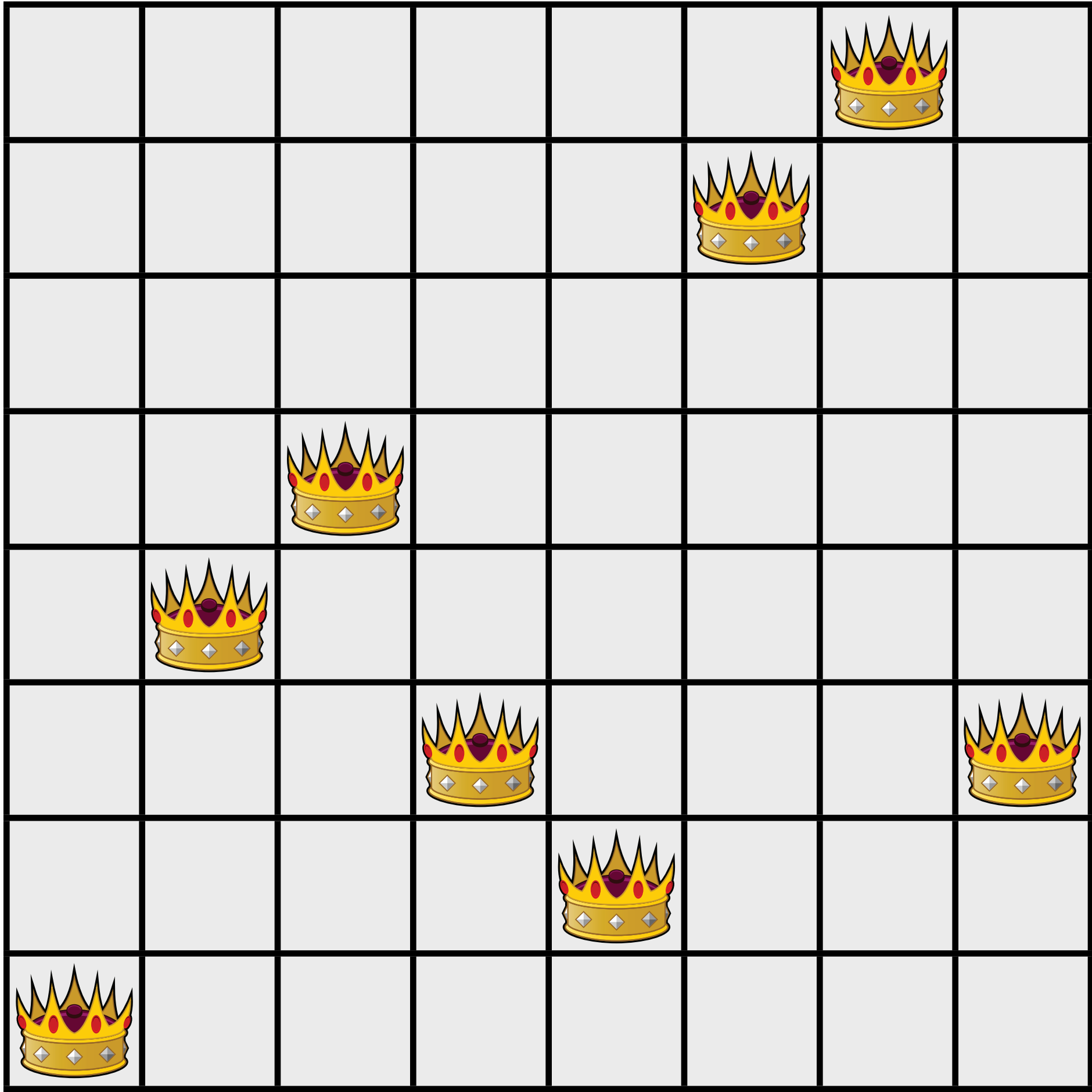




# Intuition

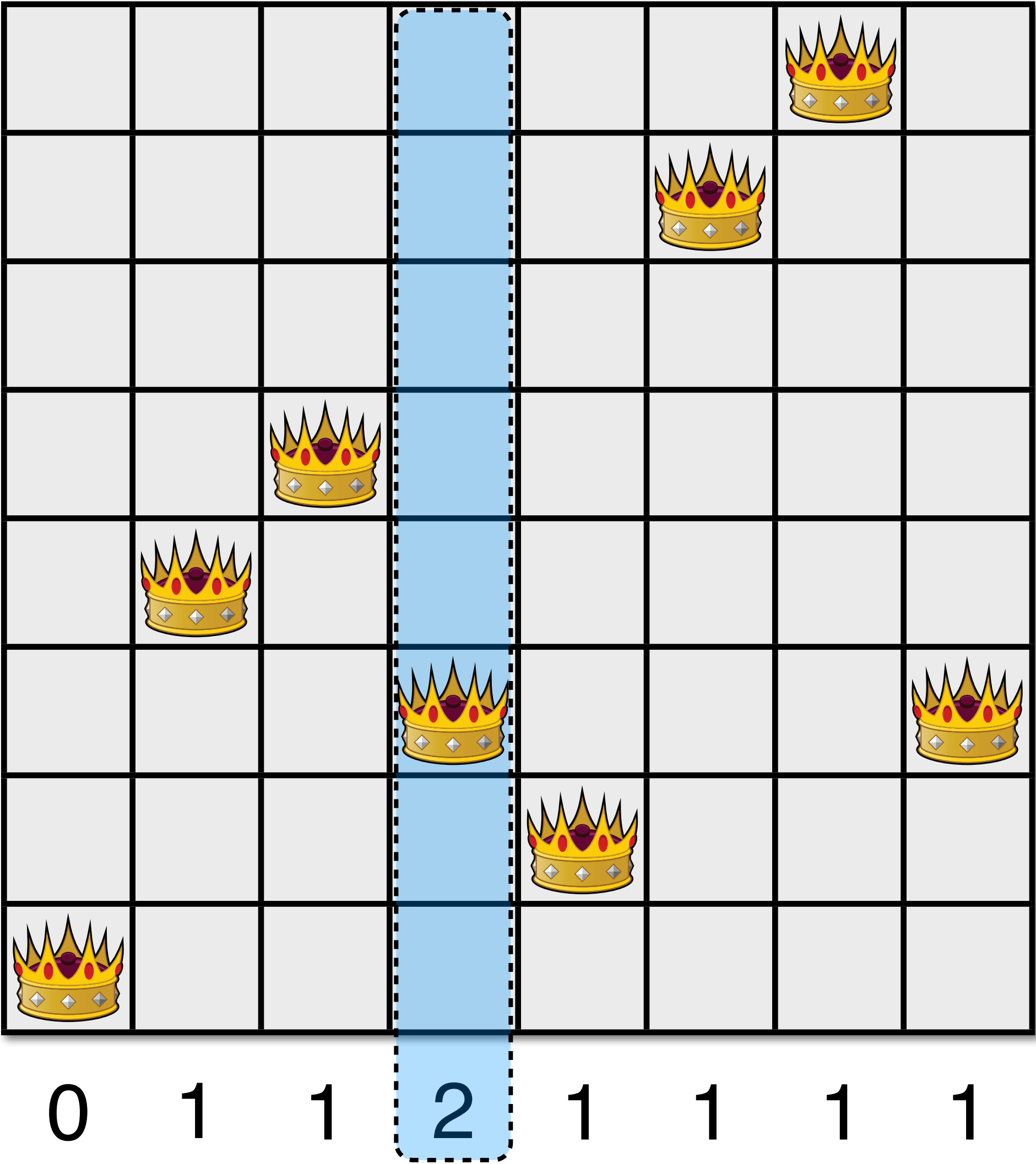


# Intuition

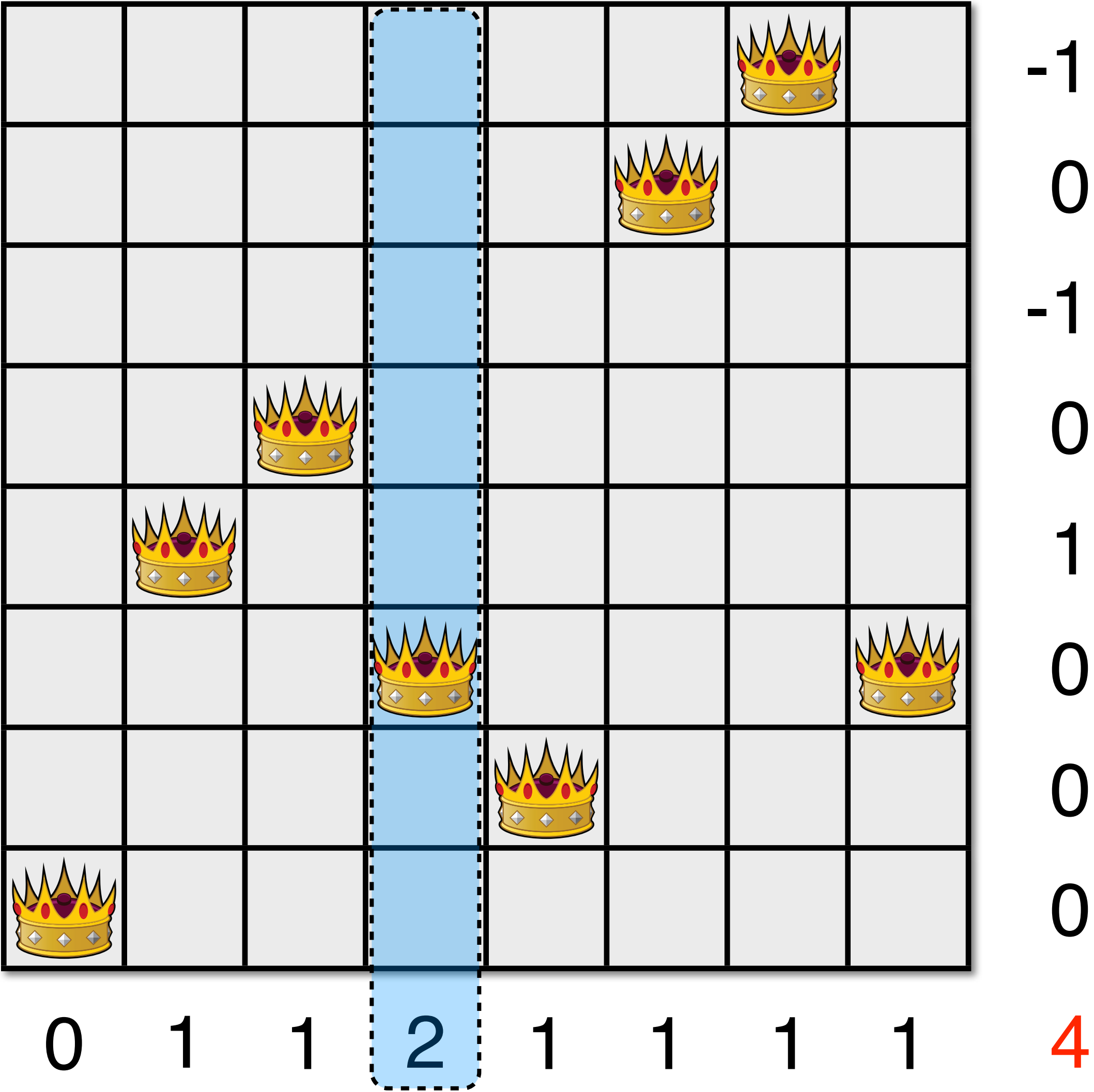


0 1 1 2 1 1 1 1 4

# Intuition

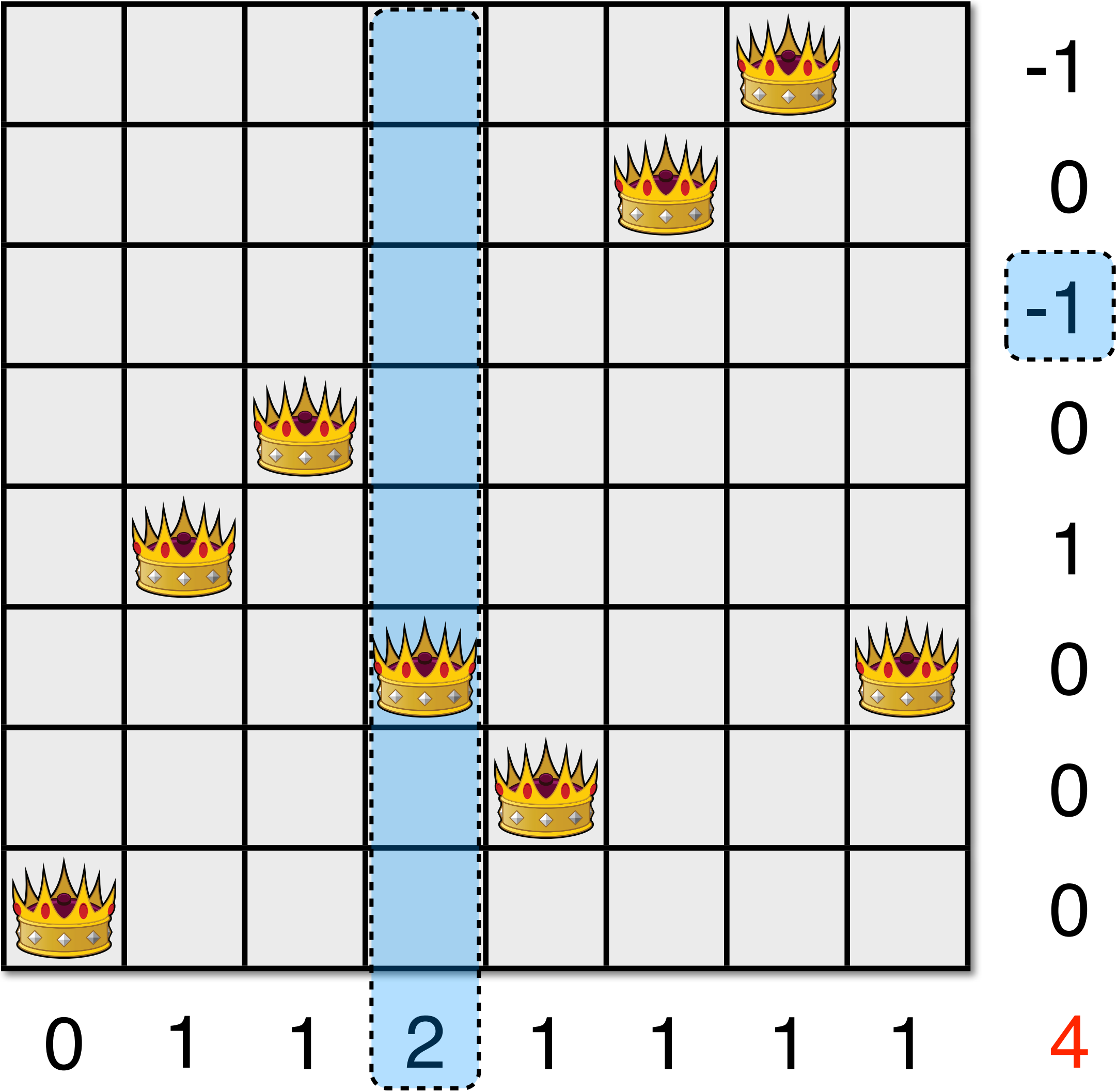


# Intuition



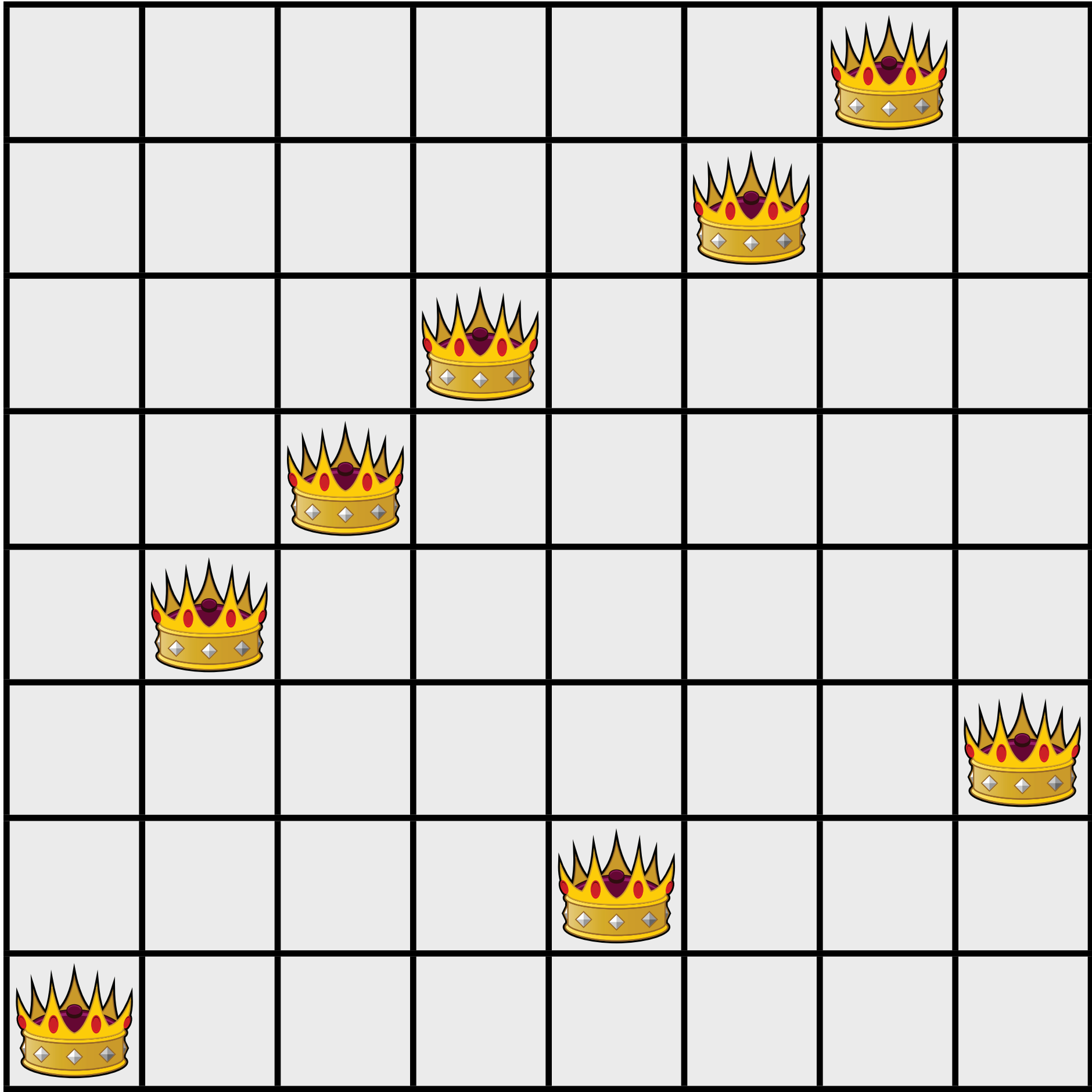


# Intuition

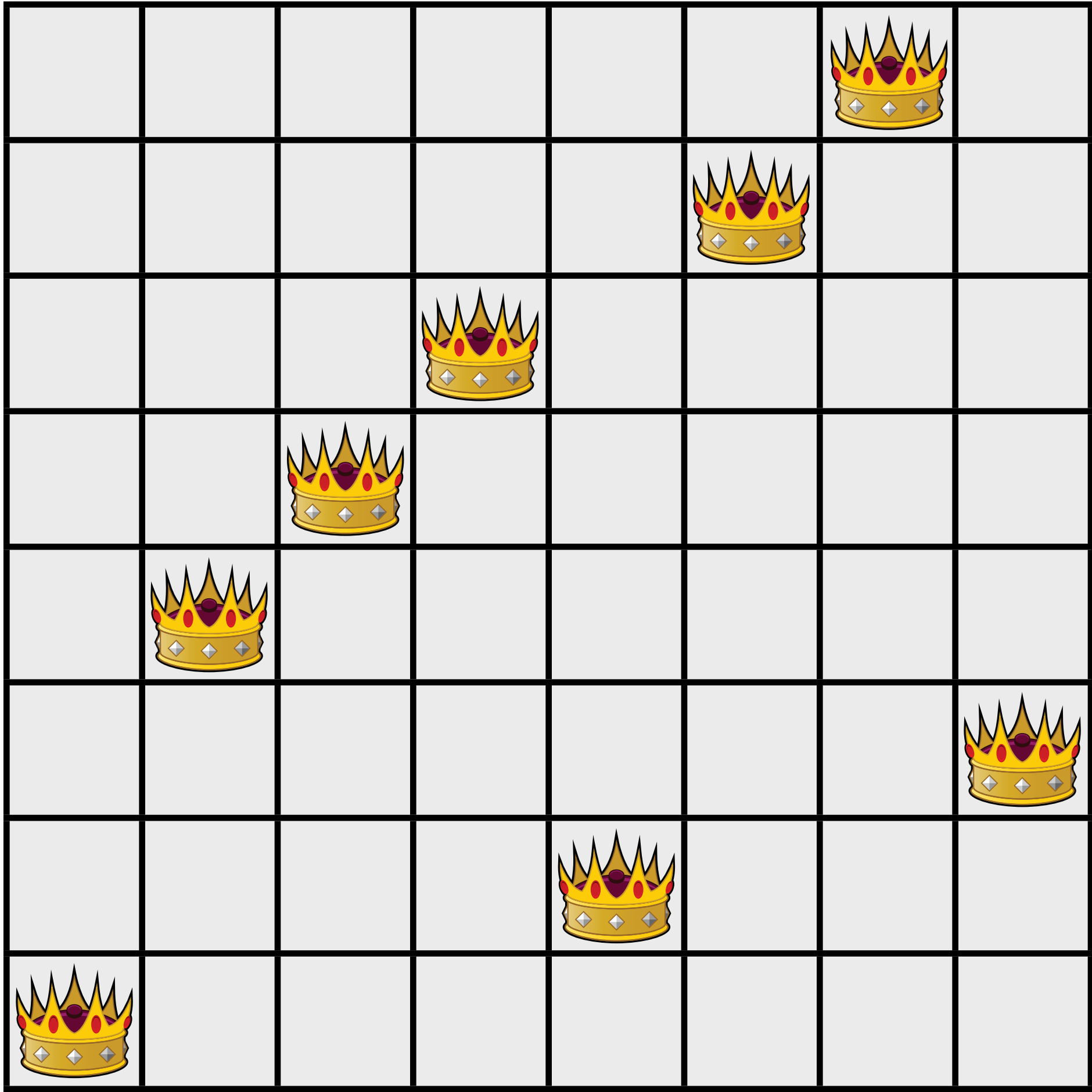




# Intuition

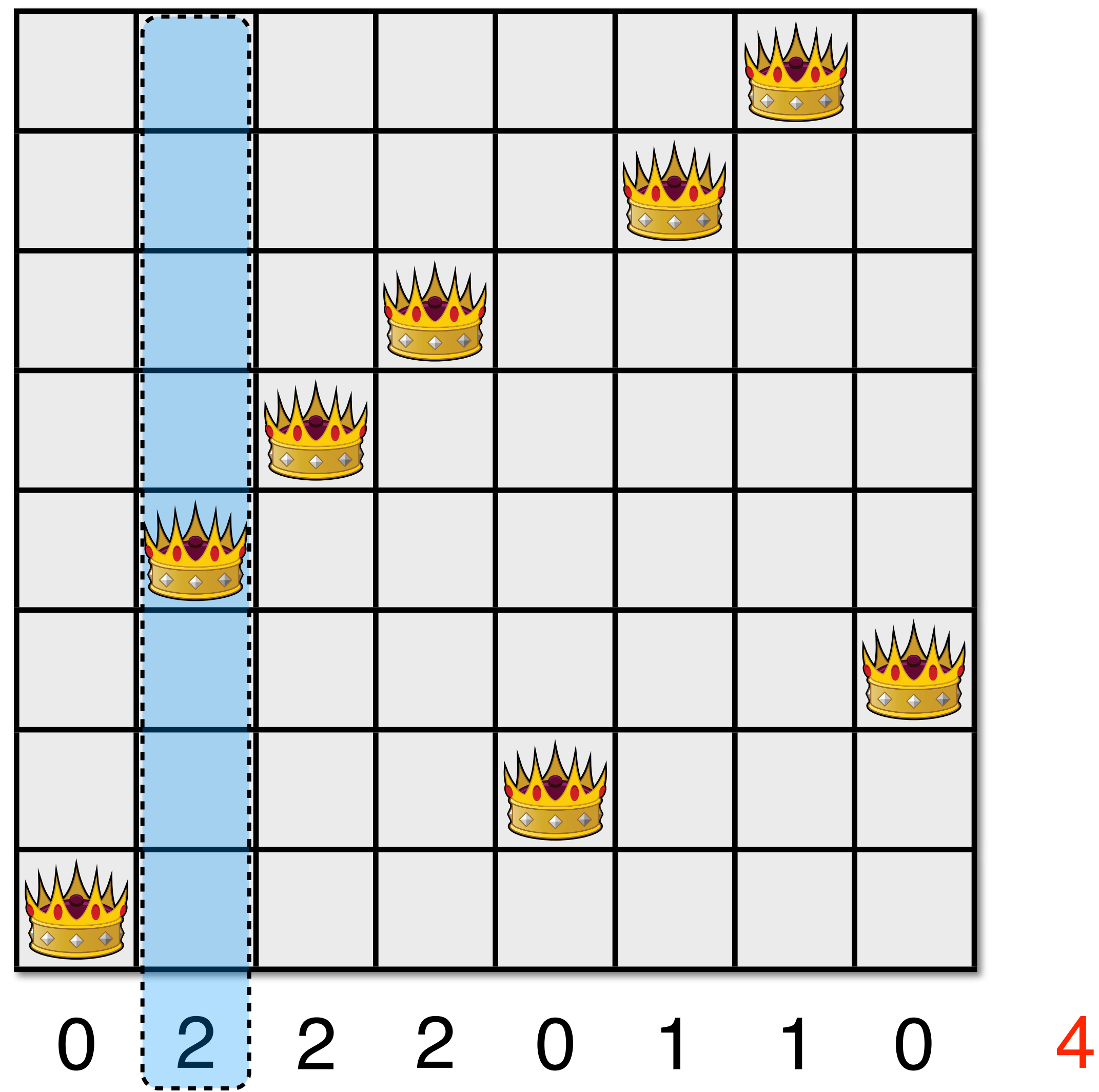


# Intuition

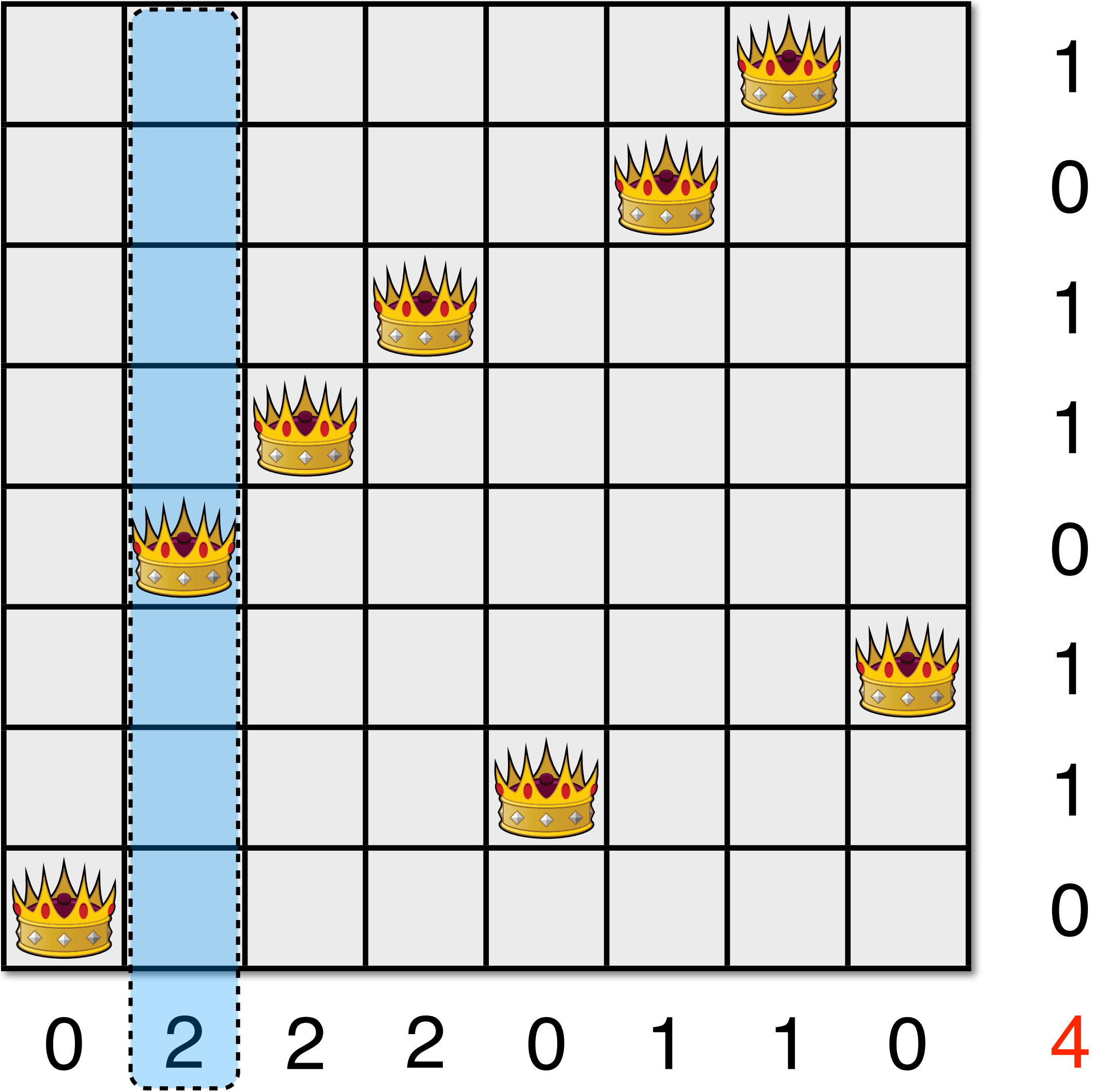


0 2 2 2 0 1 1 0 4

# Intuition

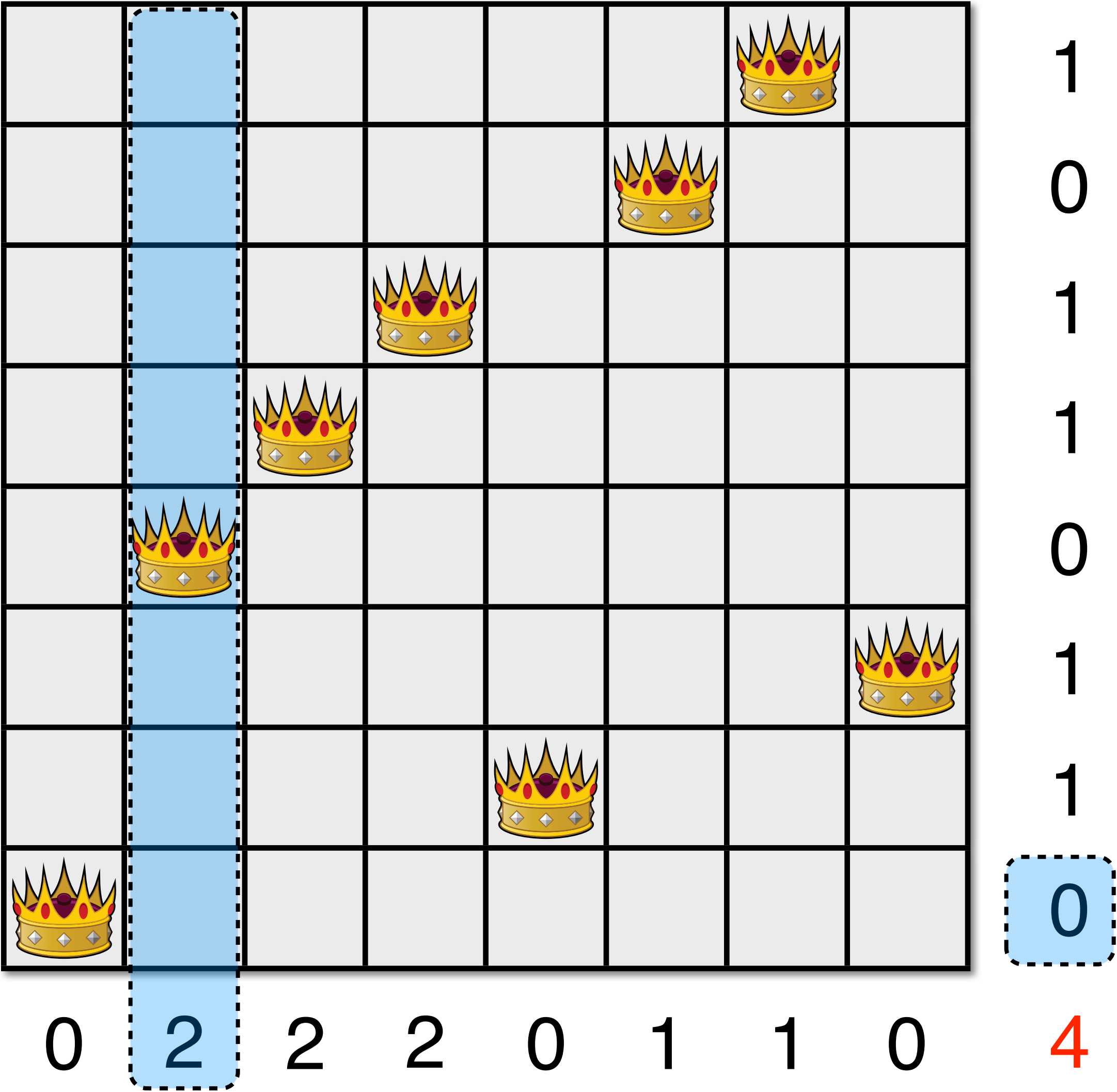


# Intuition



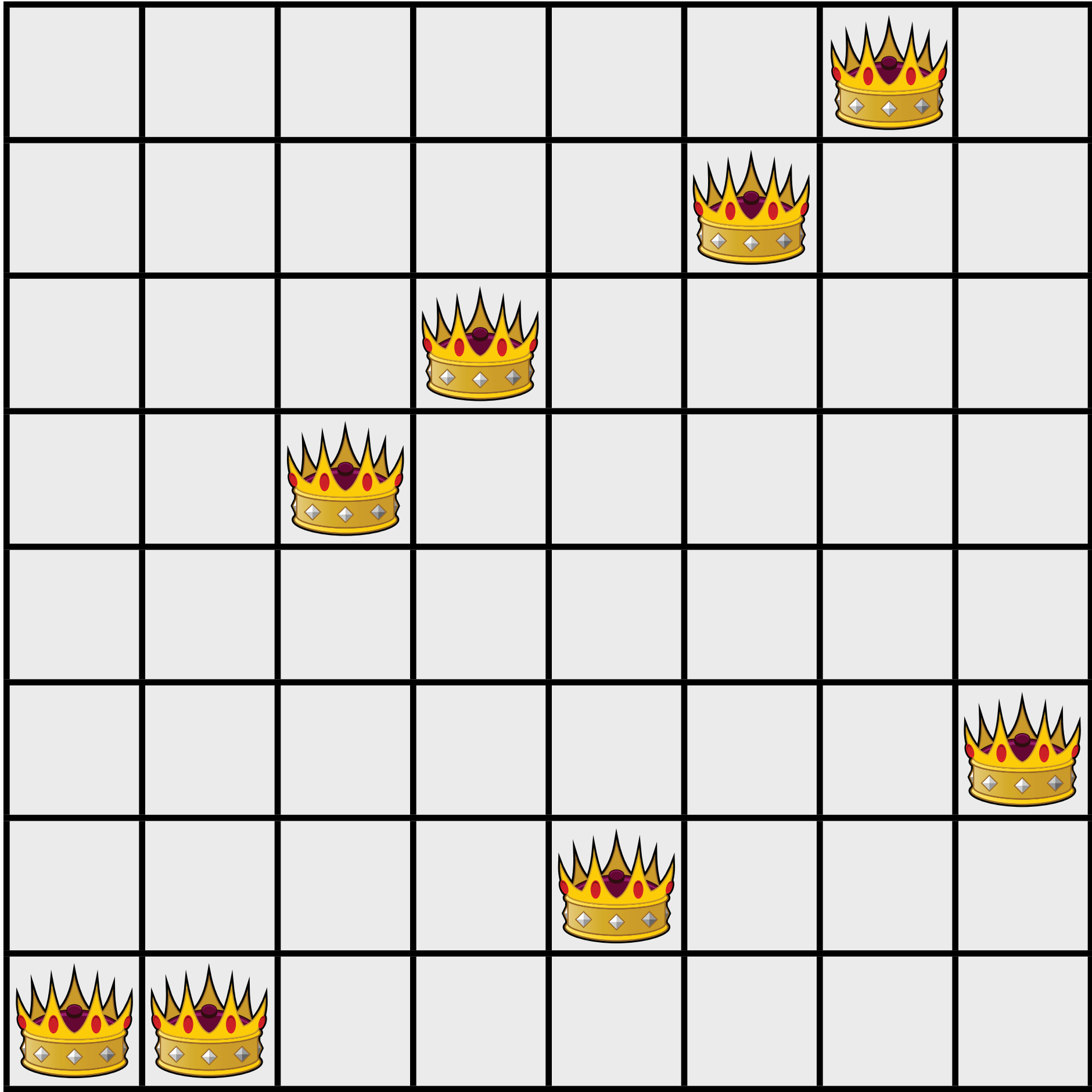


# Intuition

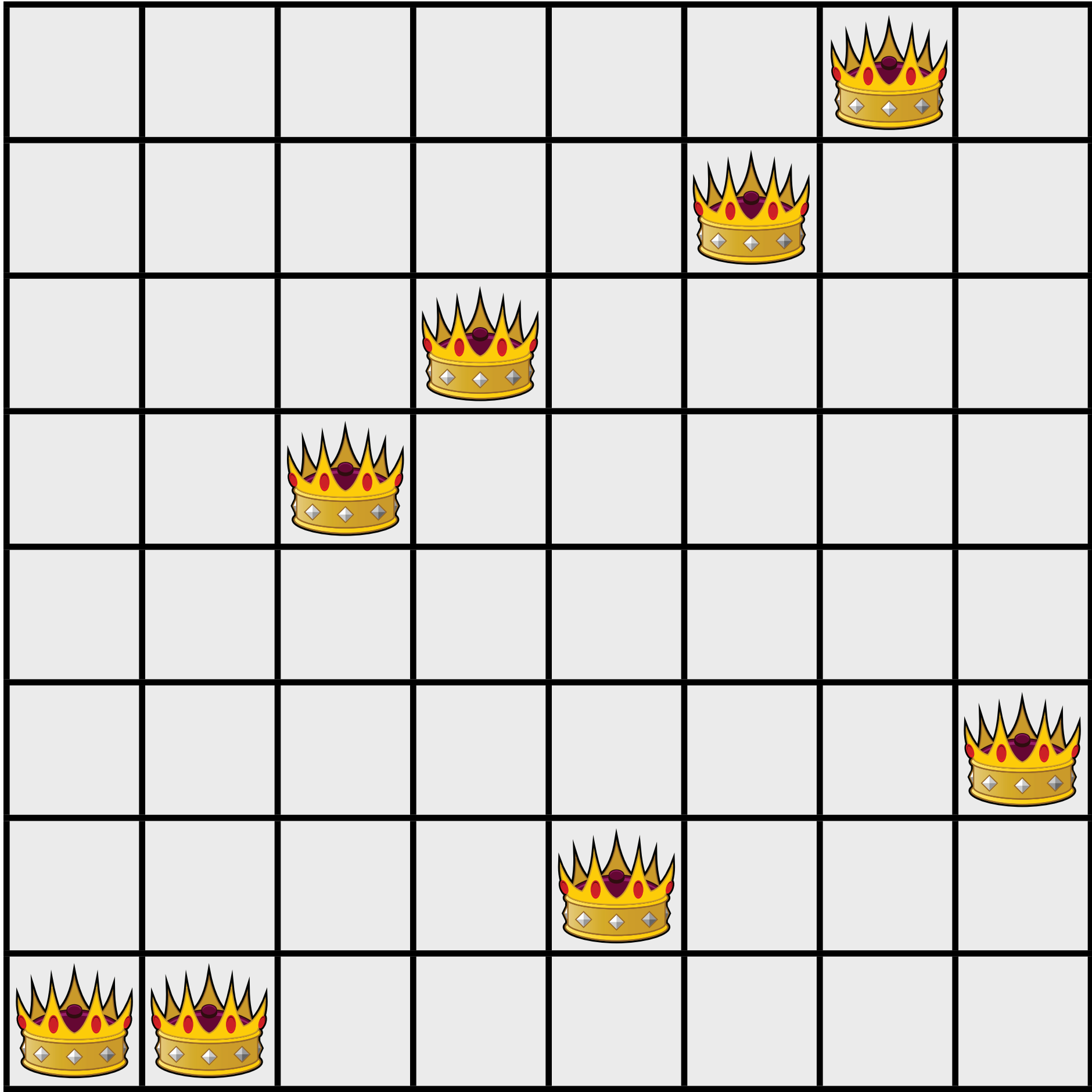




# Intuition

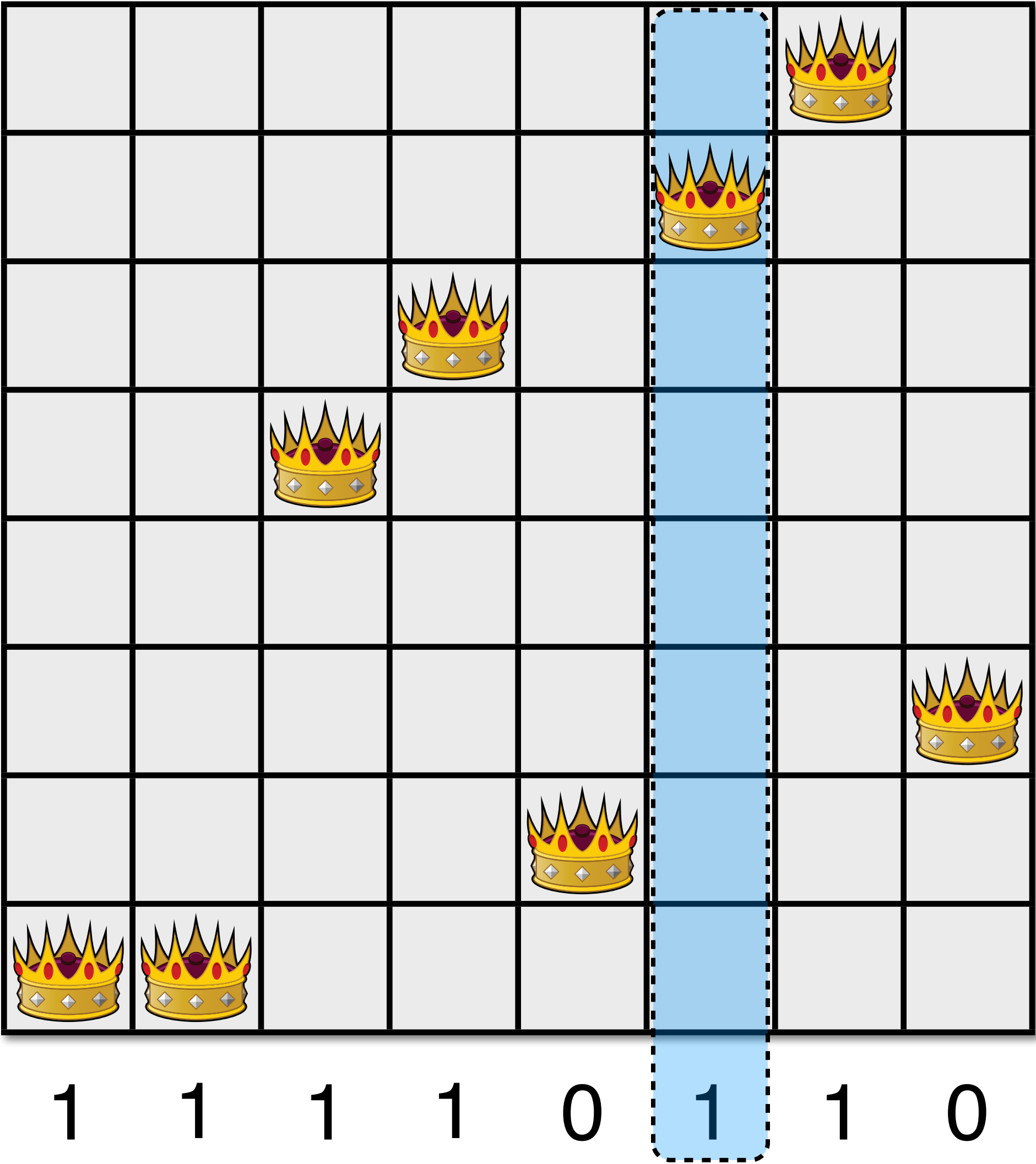


# Intuition

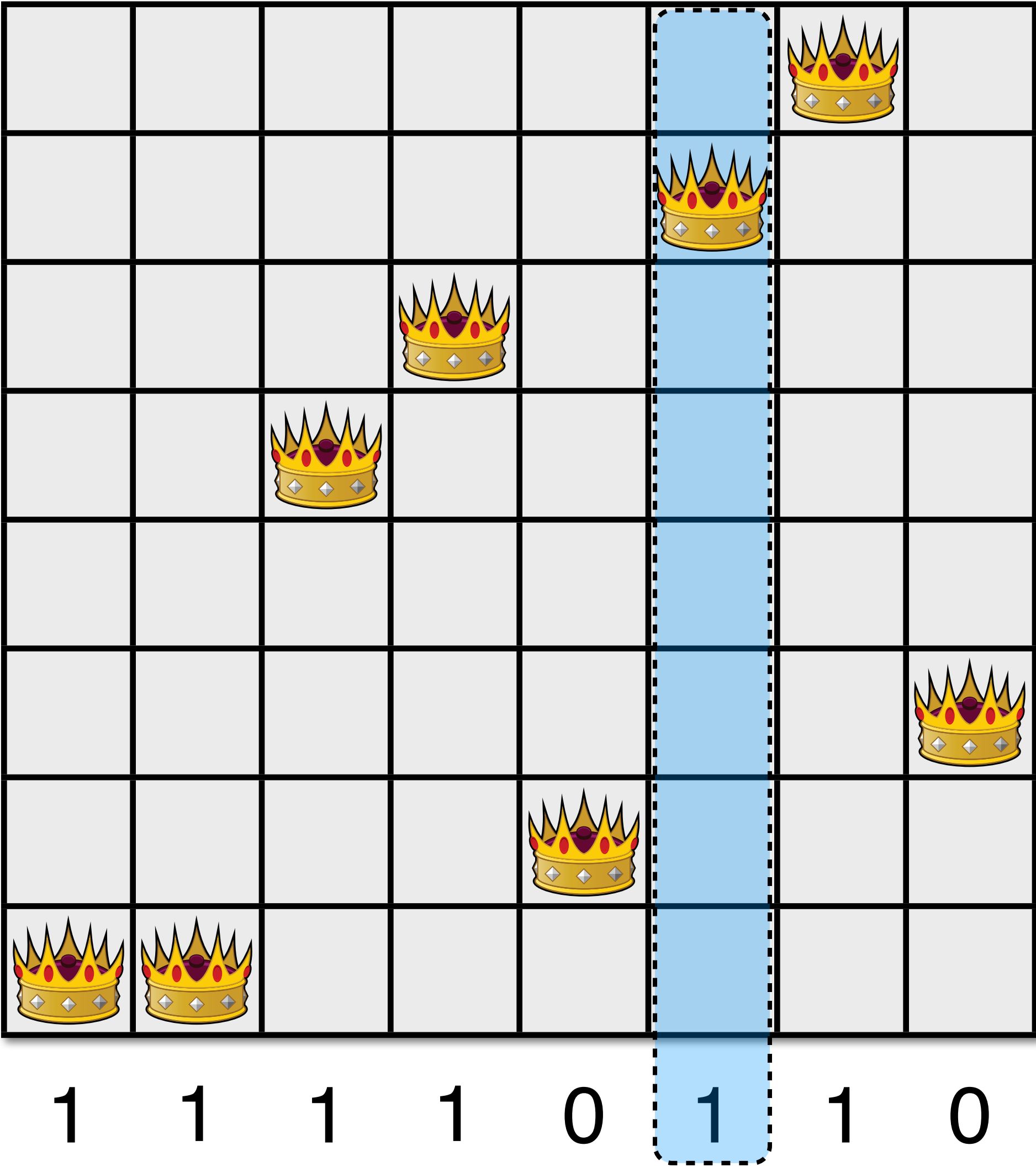


1 1 1 1 0 1 1 0 3

# Intuition

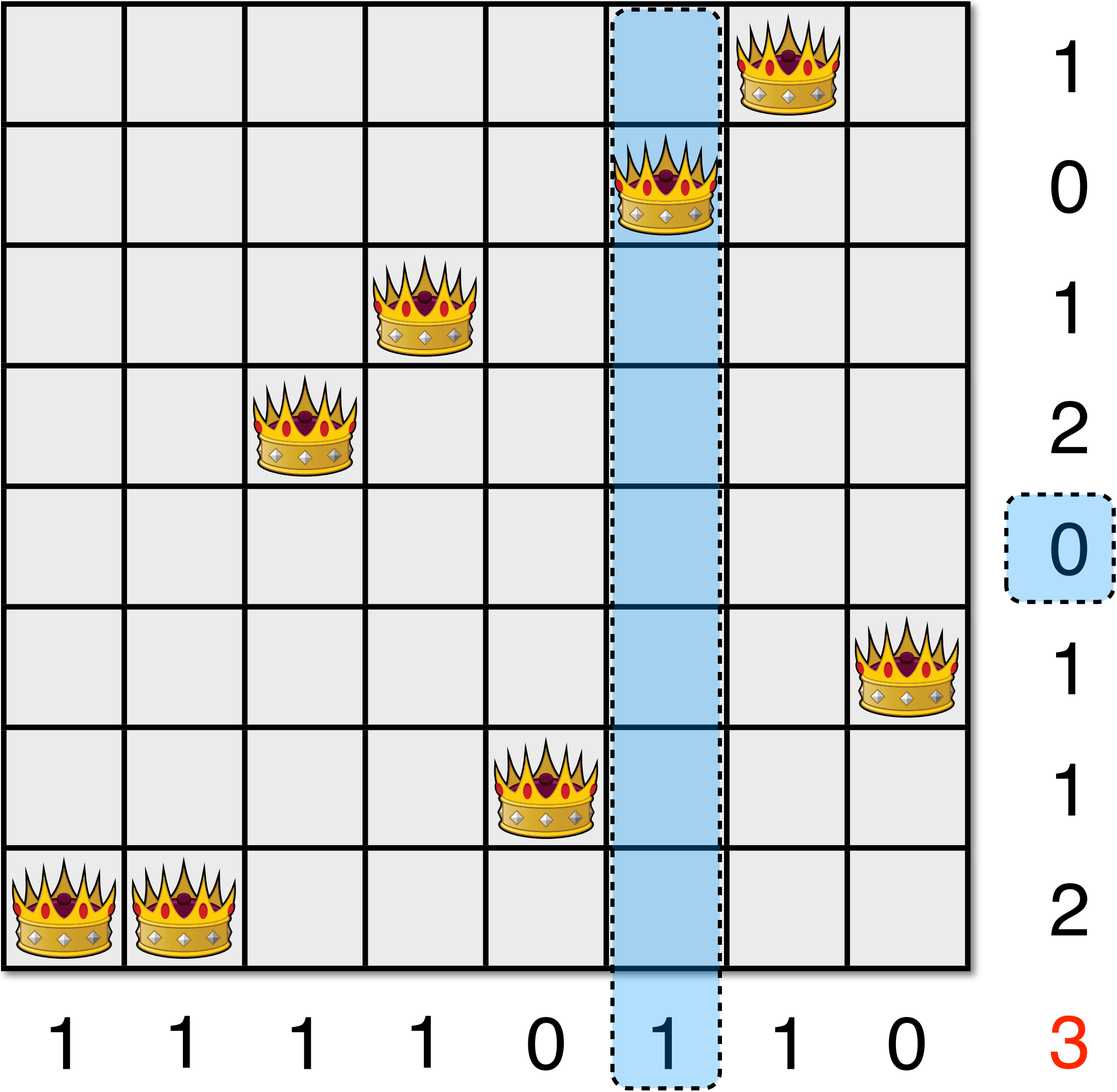


# Intuition



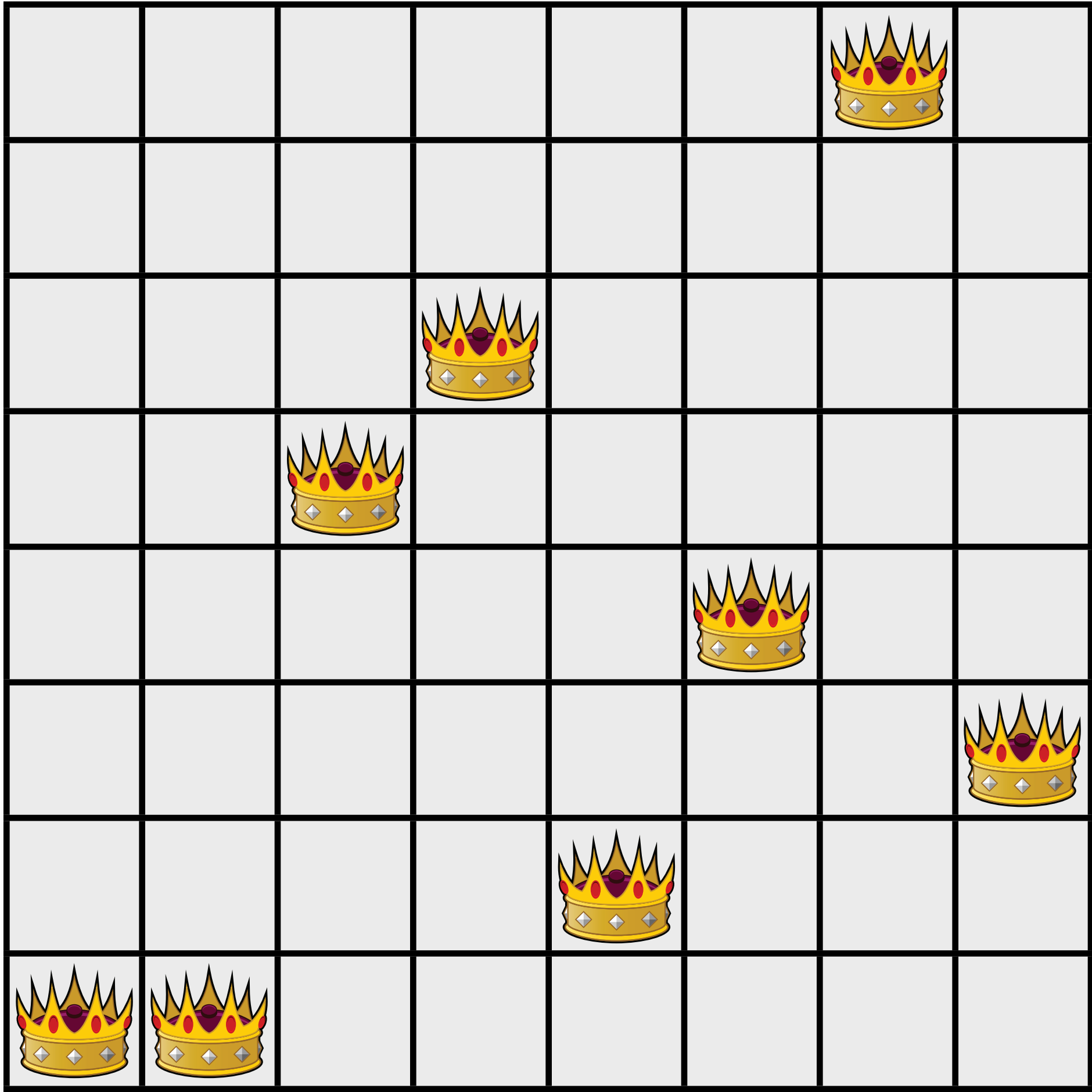
1  
0  
1  
2  
0  
1  
1  
2  
3

# Intuition

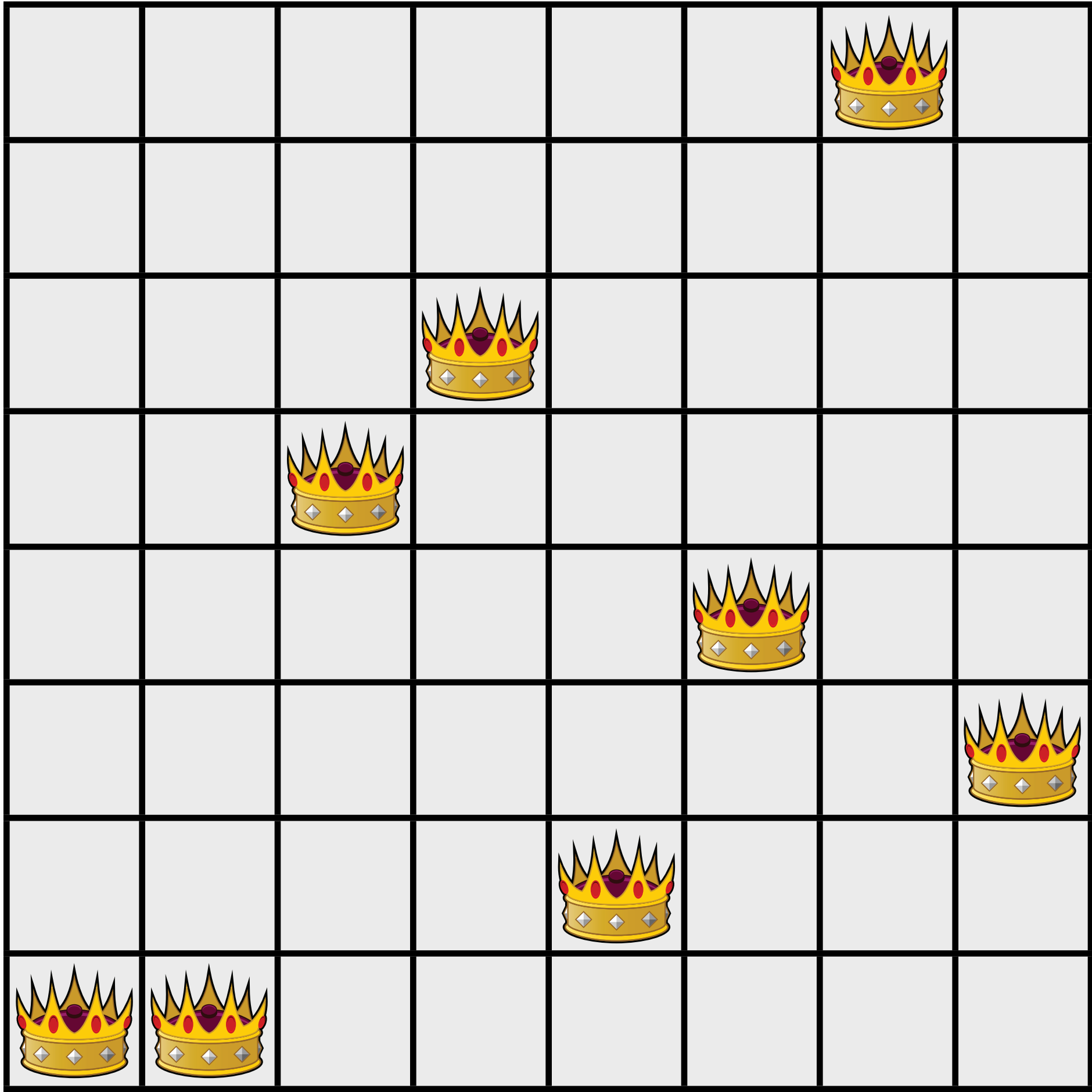




# Intuition

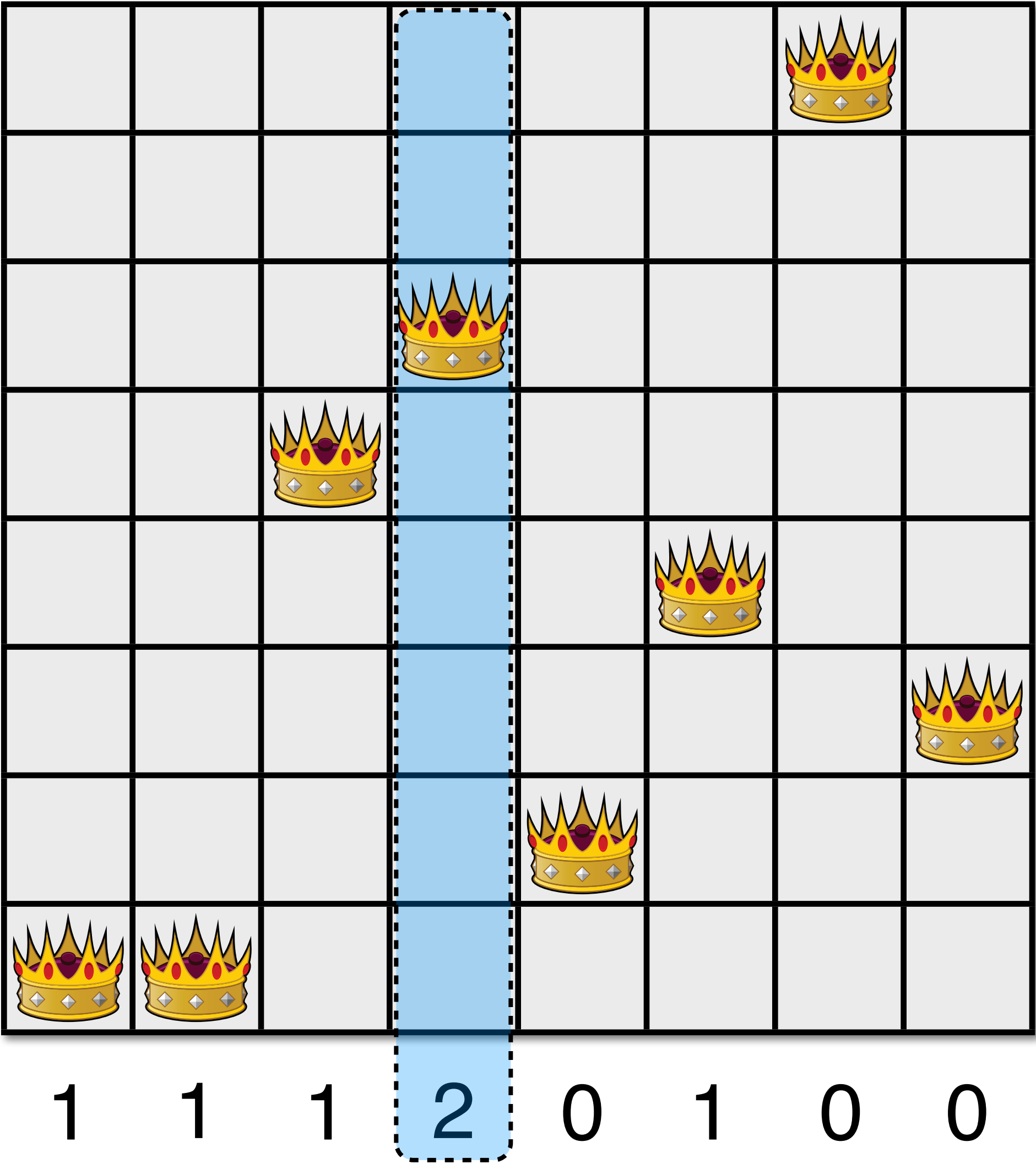


# Intuition



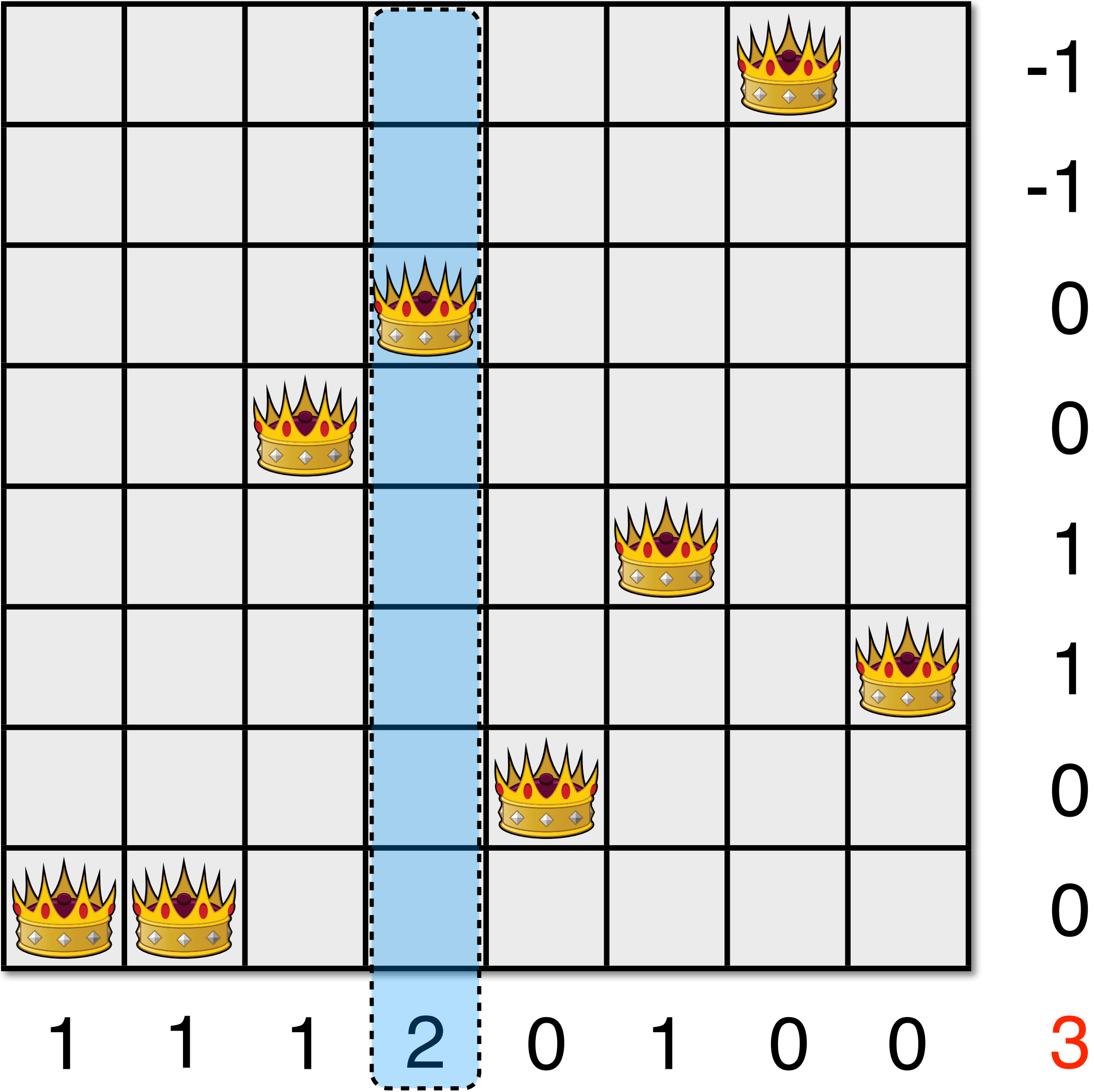
1 1 1 2 0 1 0 0 3

# Intuition

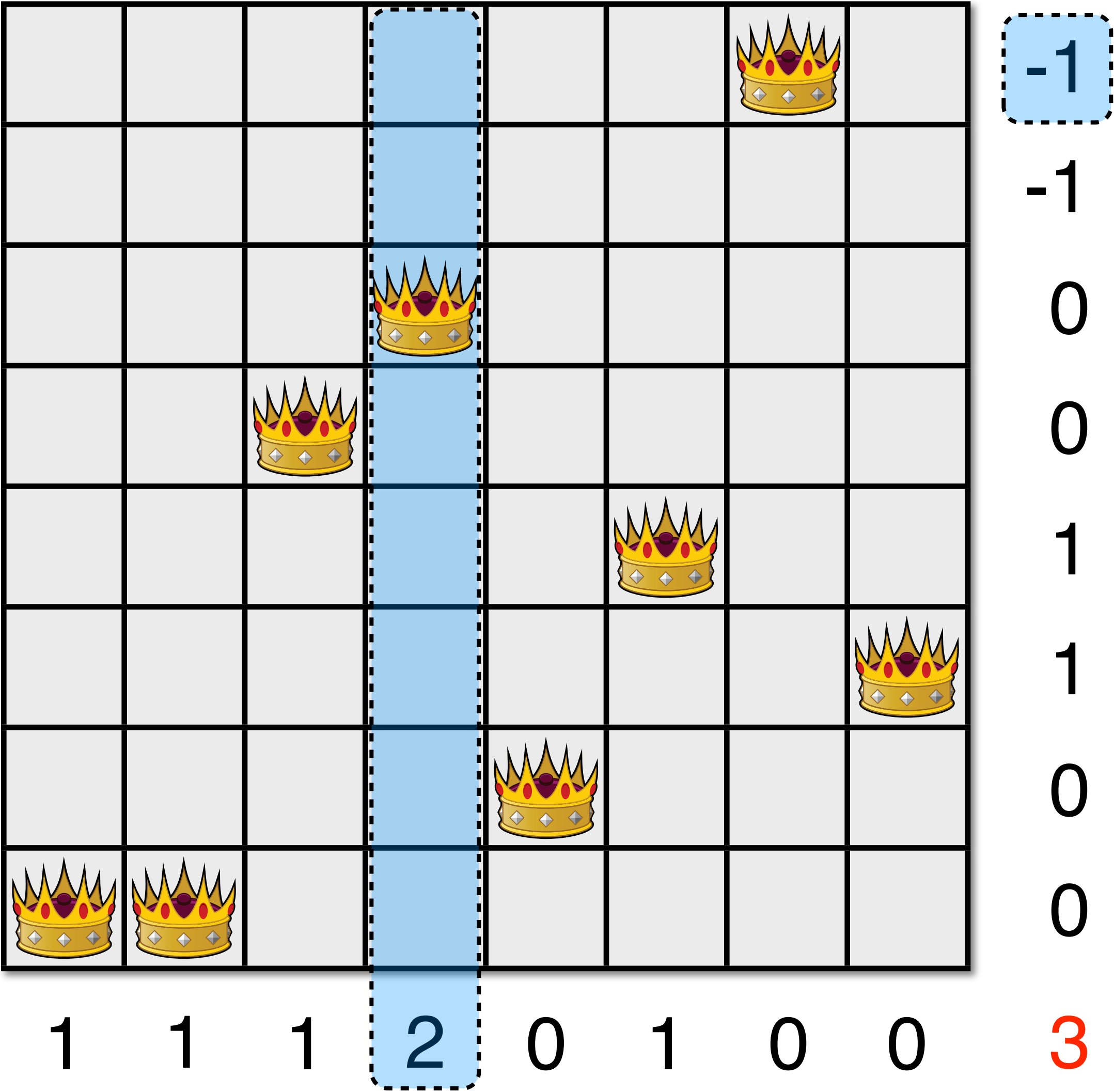


3

# Intuition

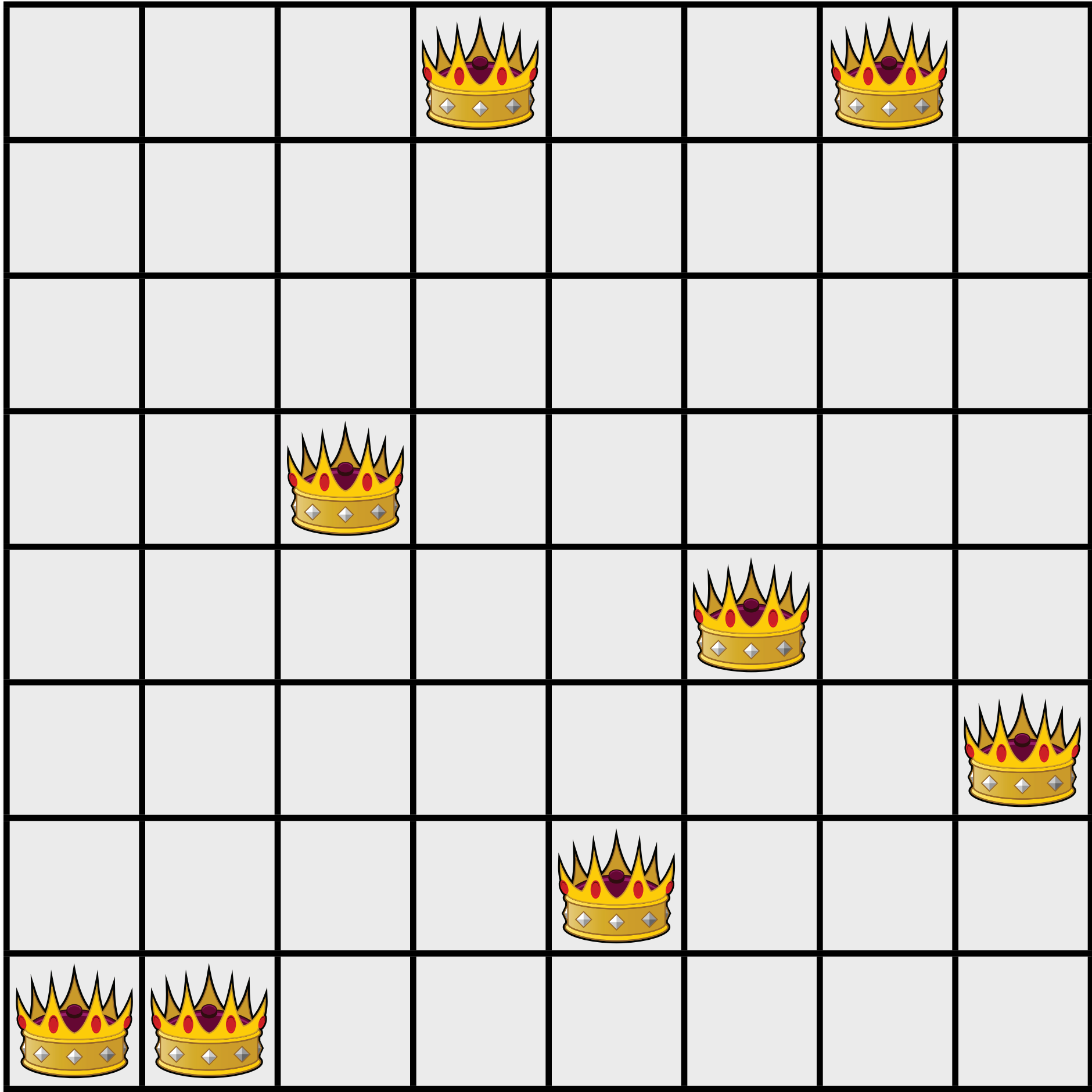


# Intuition

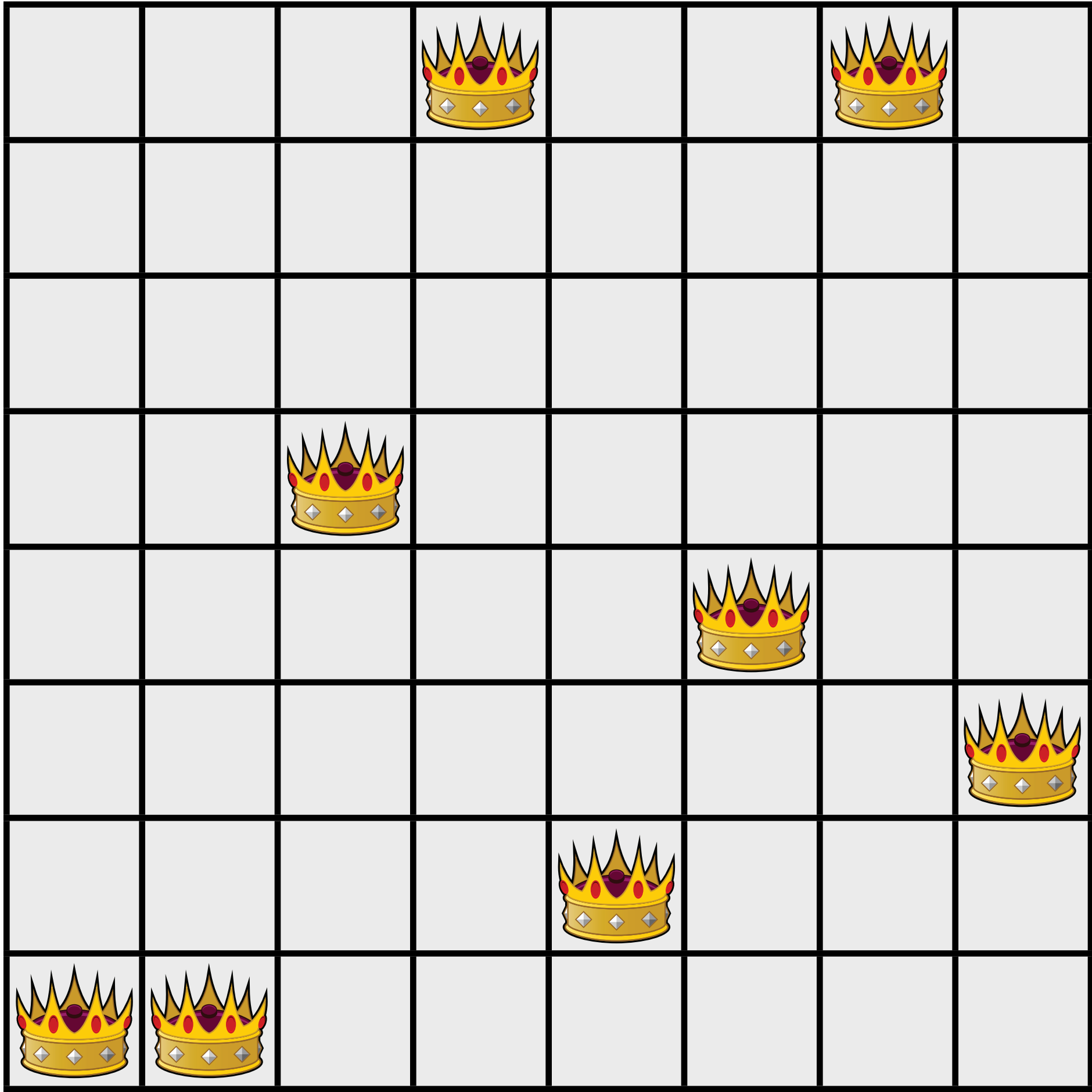




# Intuition

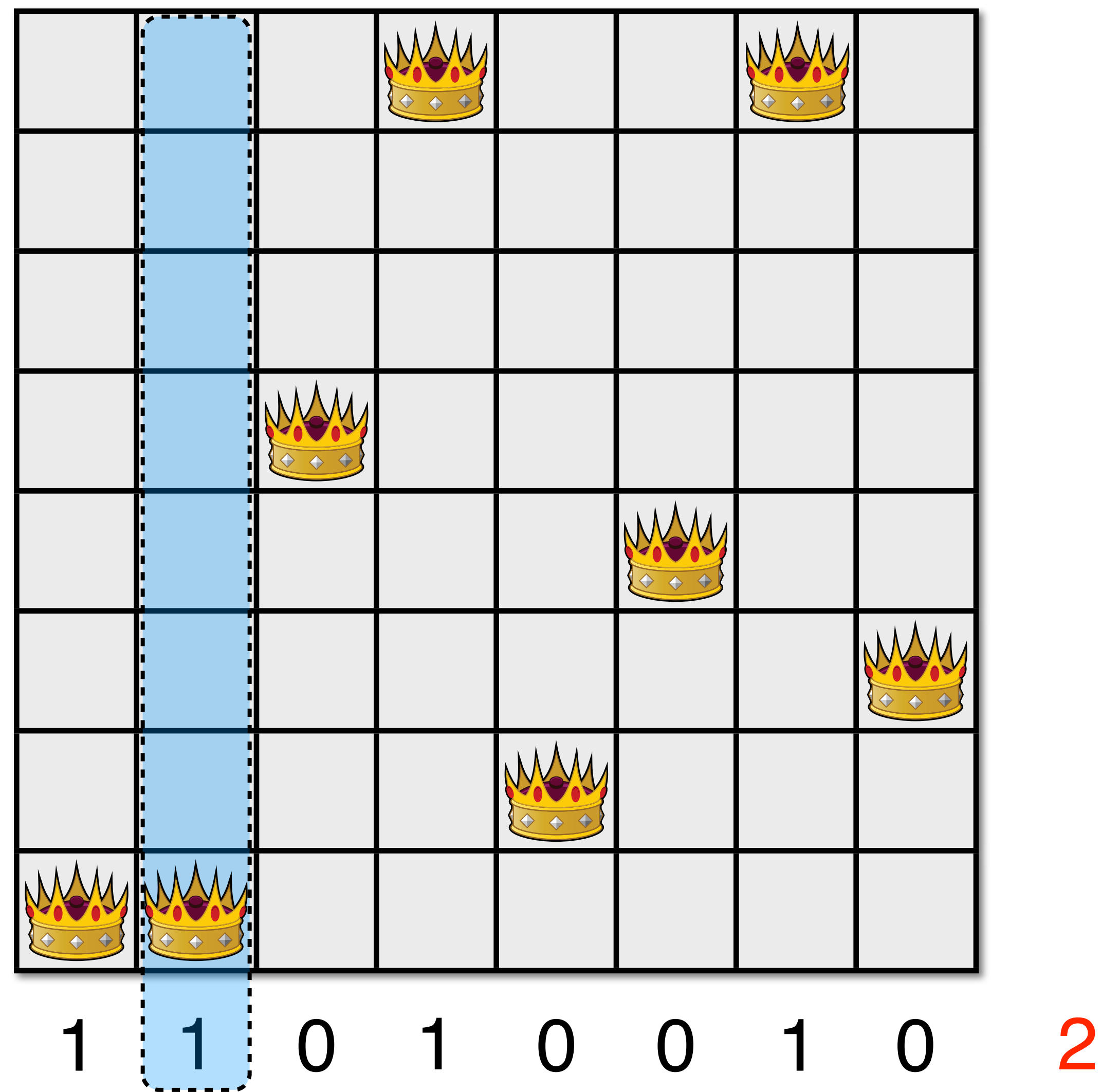


# Intuition

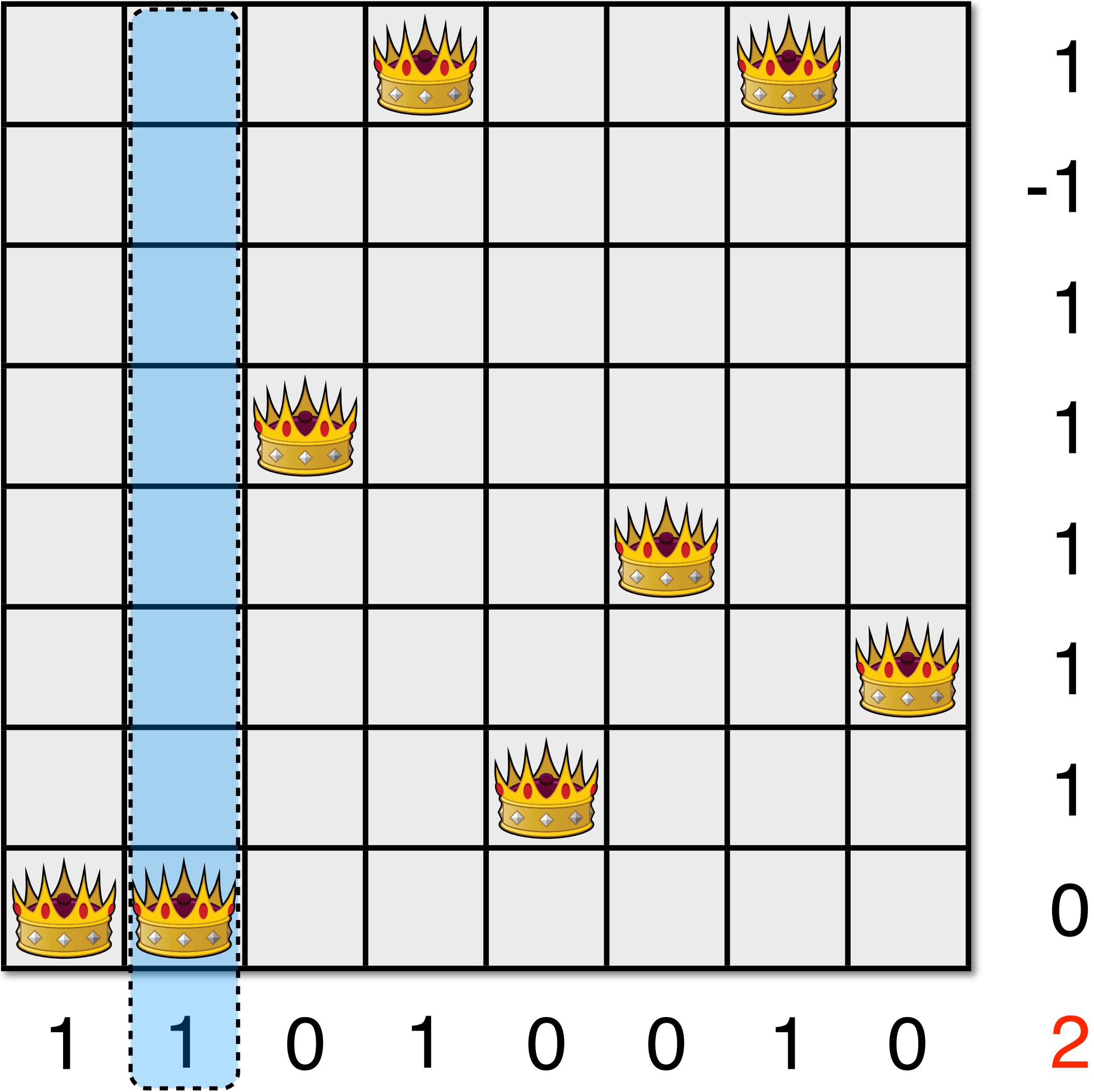


1 1 0 1 0 0 1 0 2

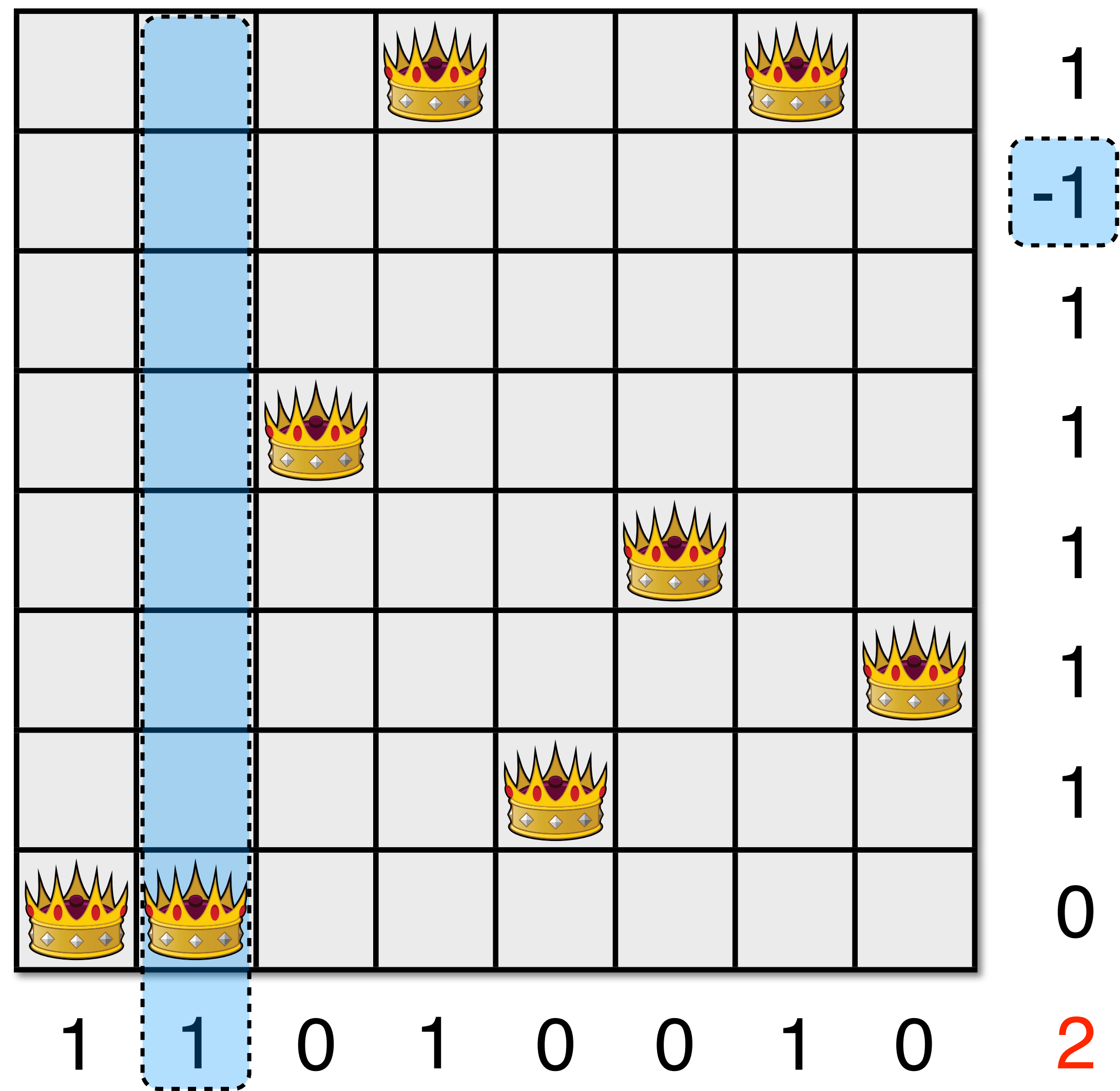
# Intuition



# Intuition

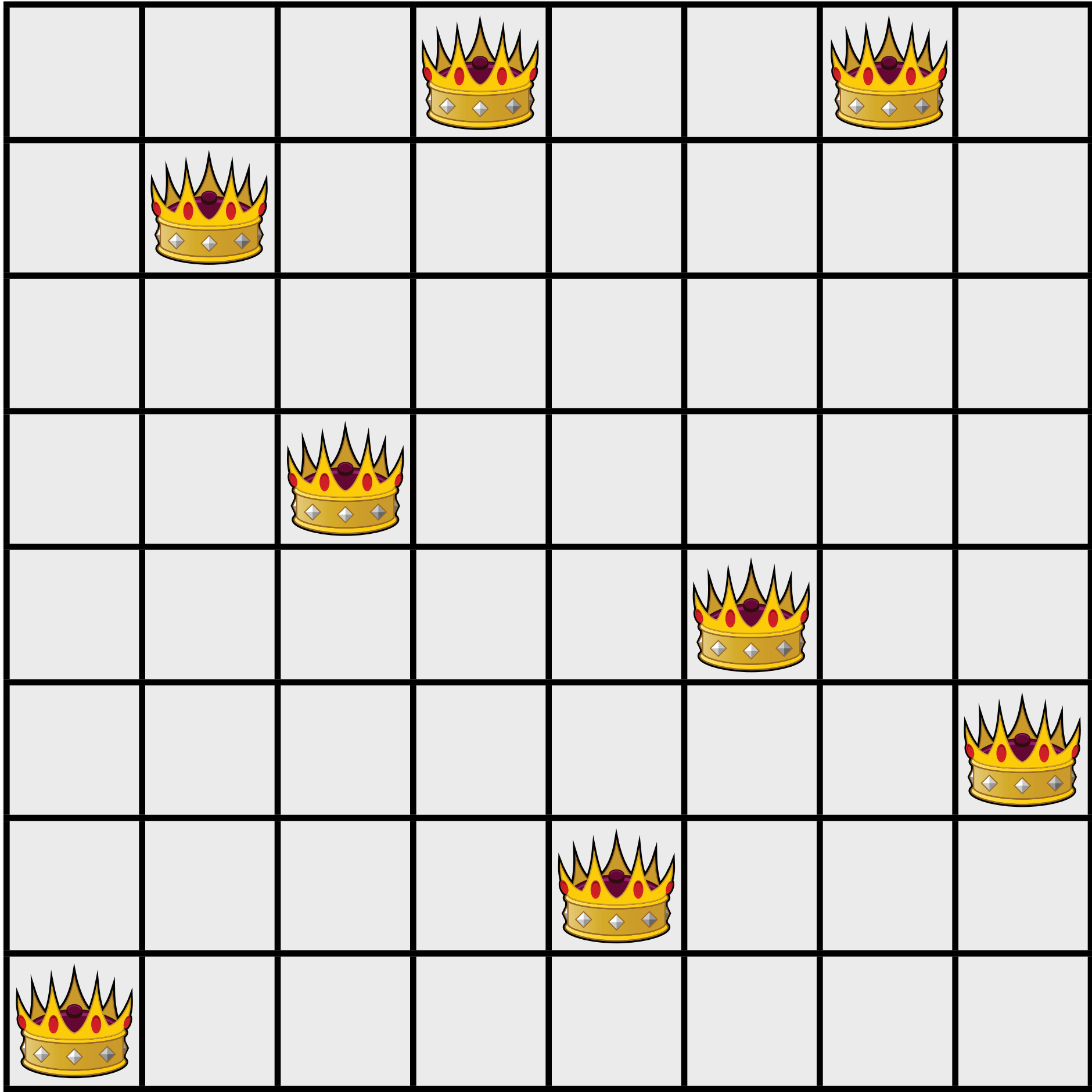


# Intuition

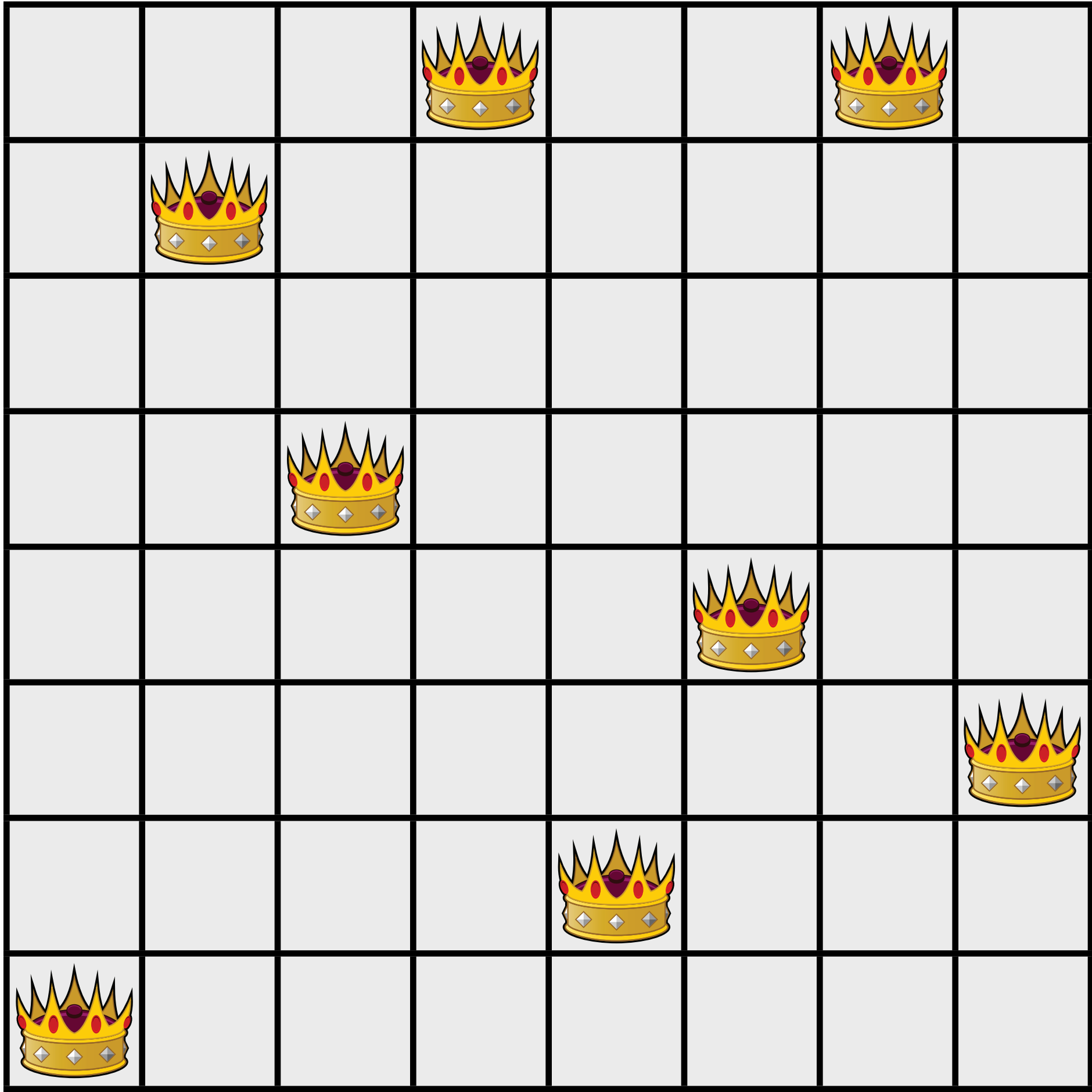




# Intuition

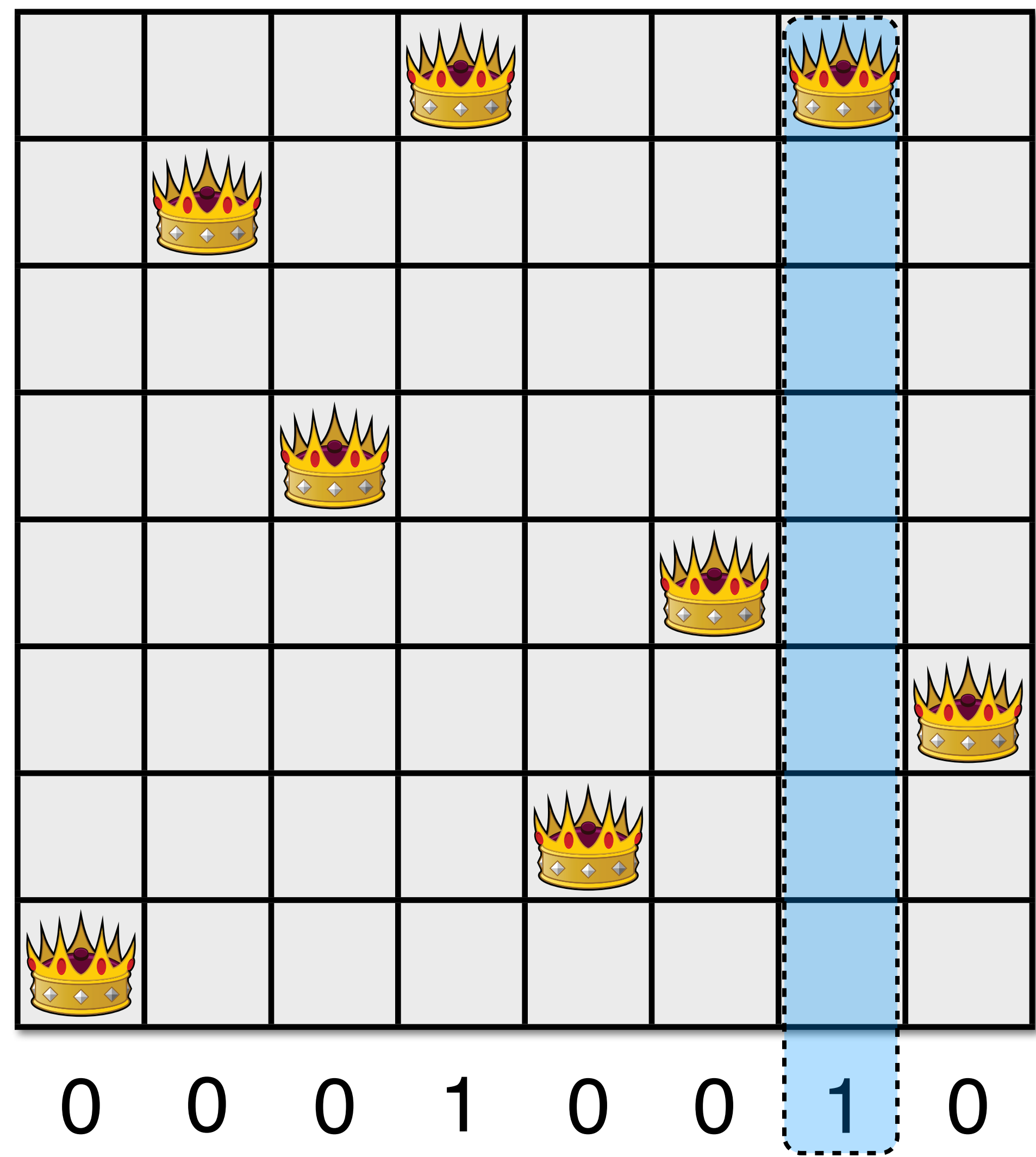


# Intuition



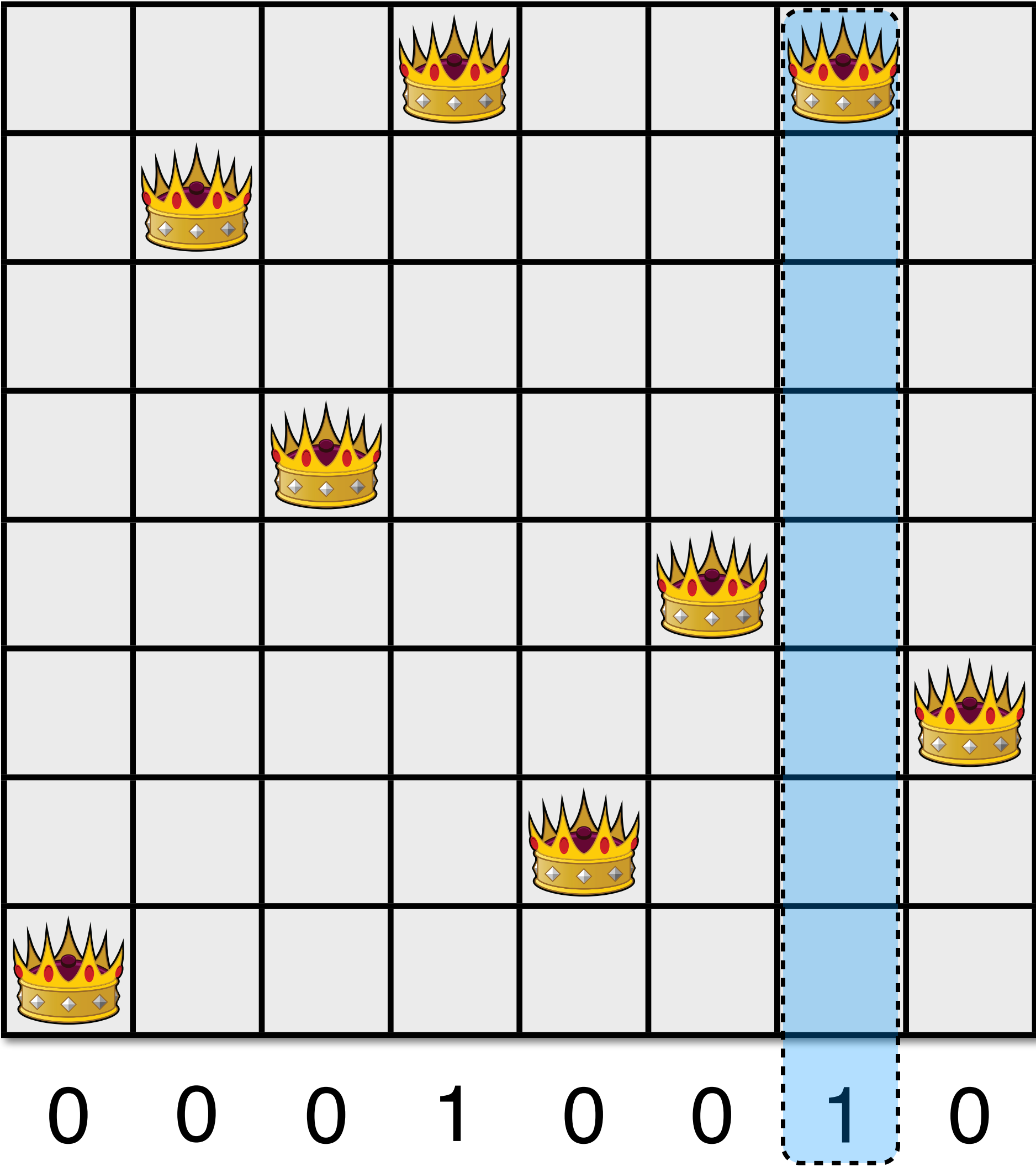
0 0 0 1 0 0 1 0 1

# Intuition



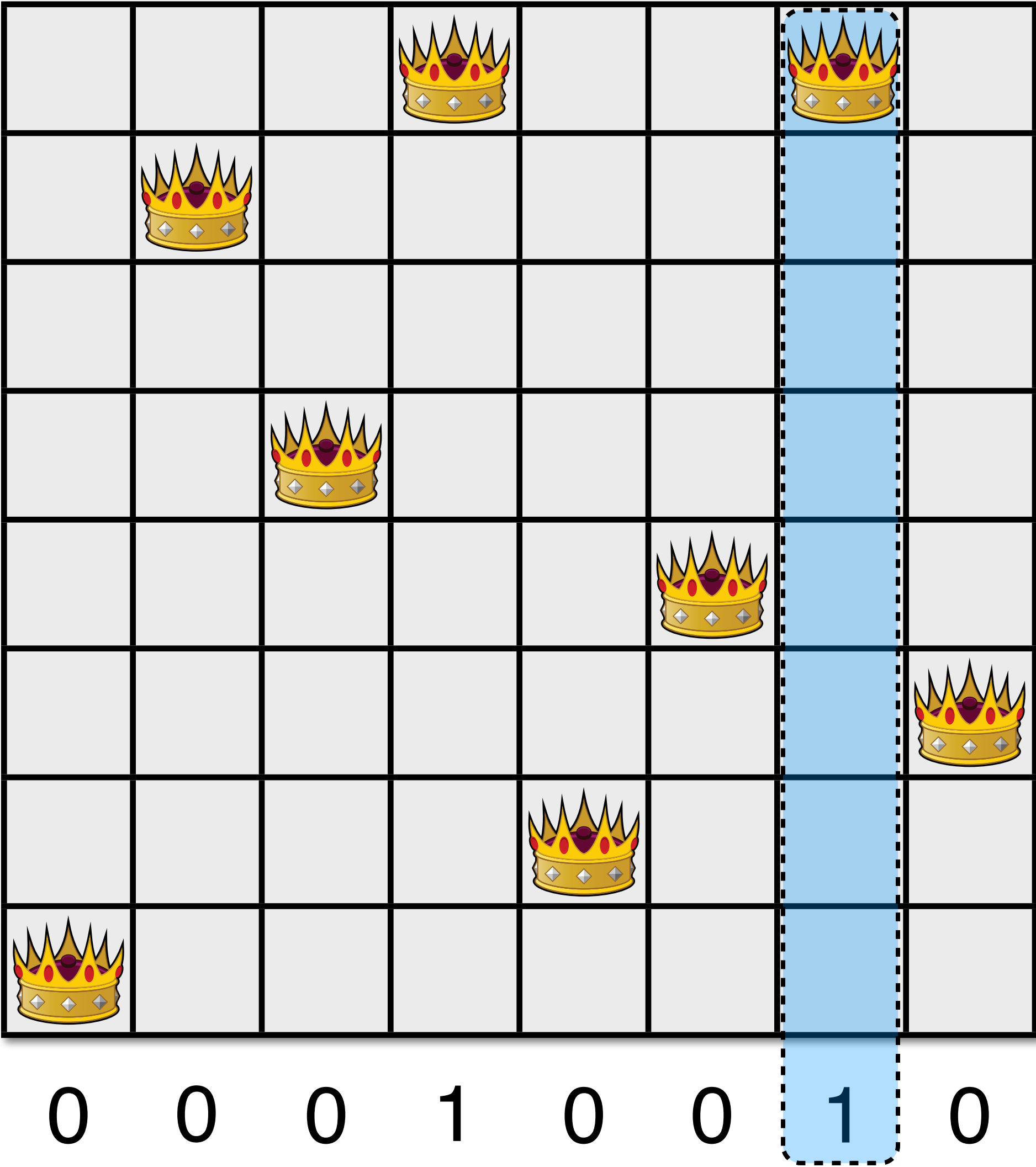
1

# Intuition



0  
1  
-1  
2  
2  
1  
2  
1  
**1**

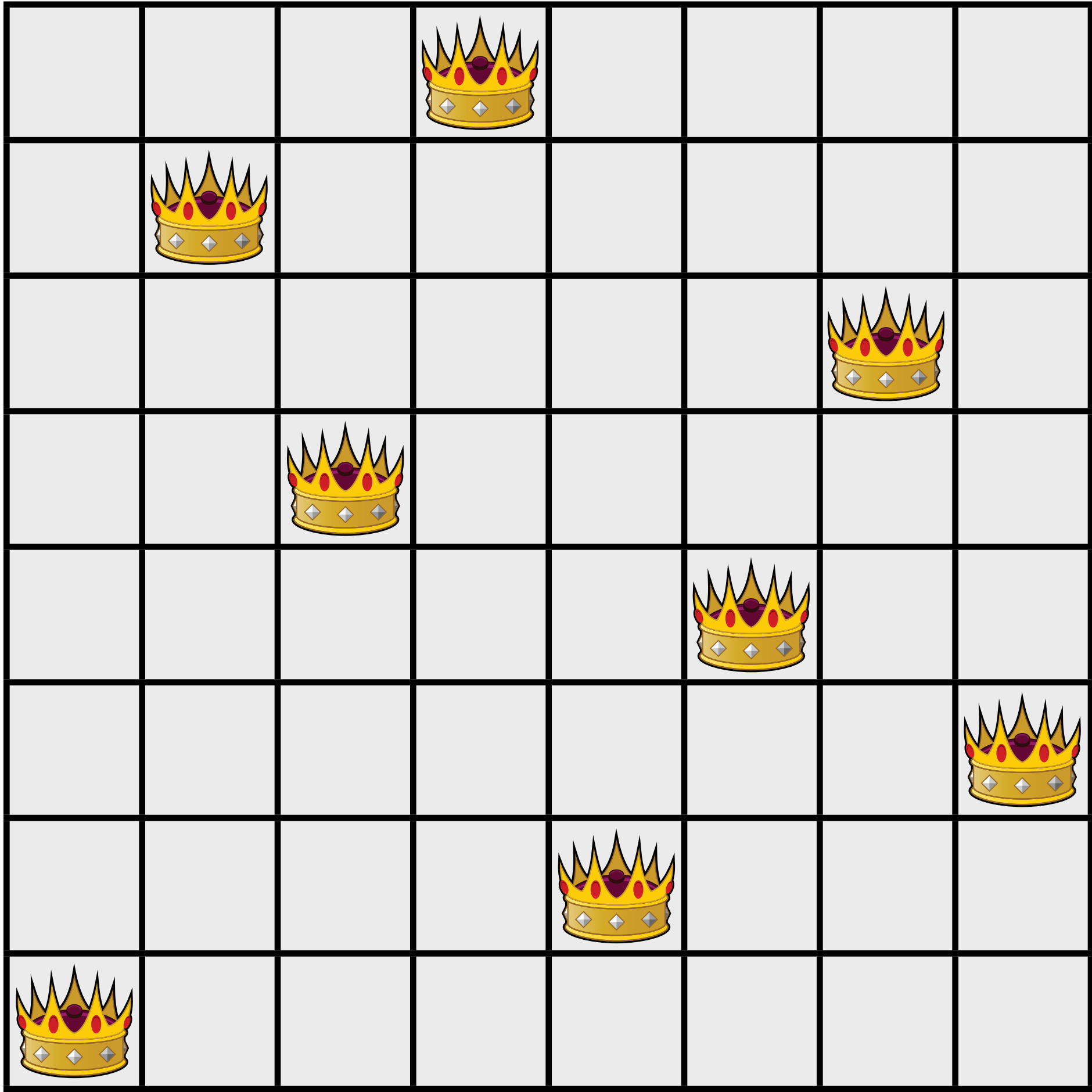
# Intuition



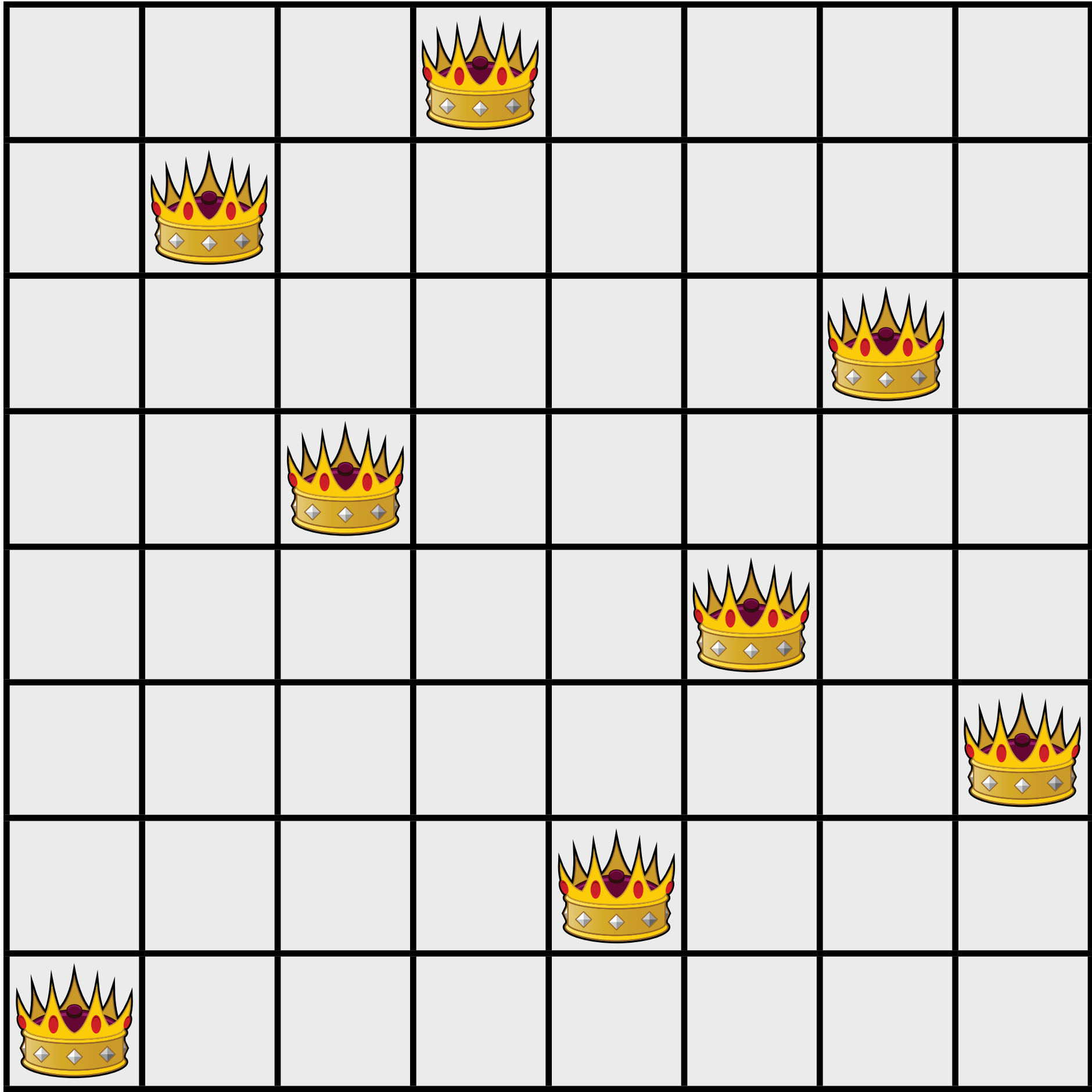
0  
1  
-1  
2  
2  
1  
2  
1  
1



# Intuition

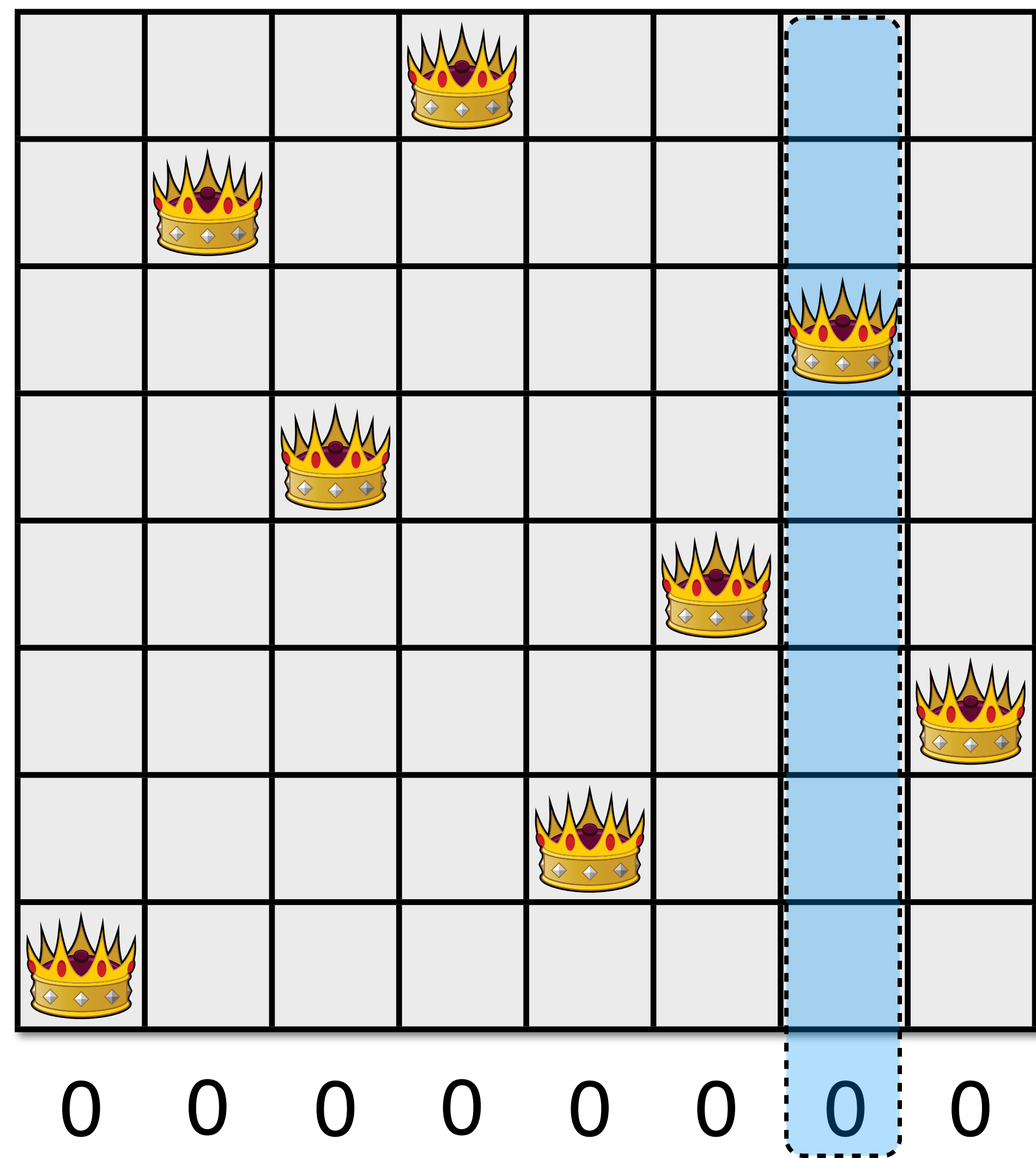


# Intuition

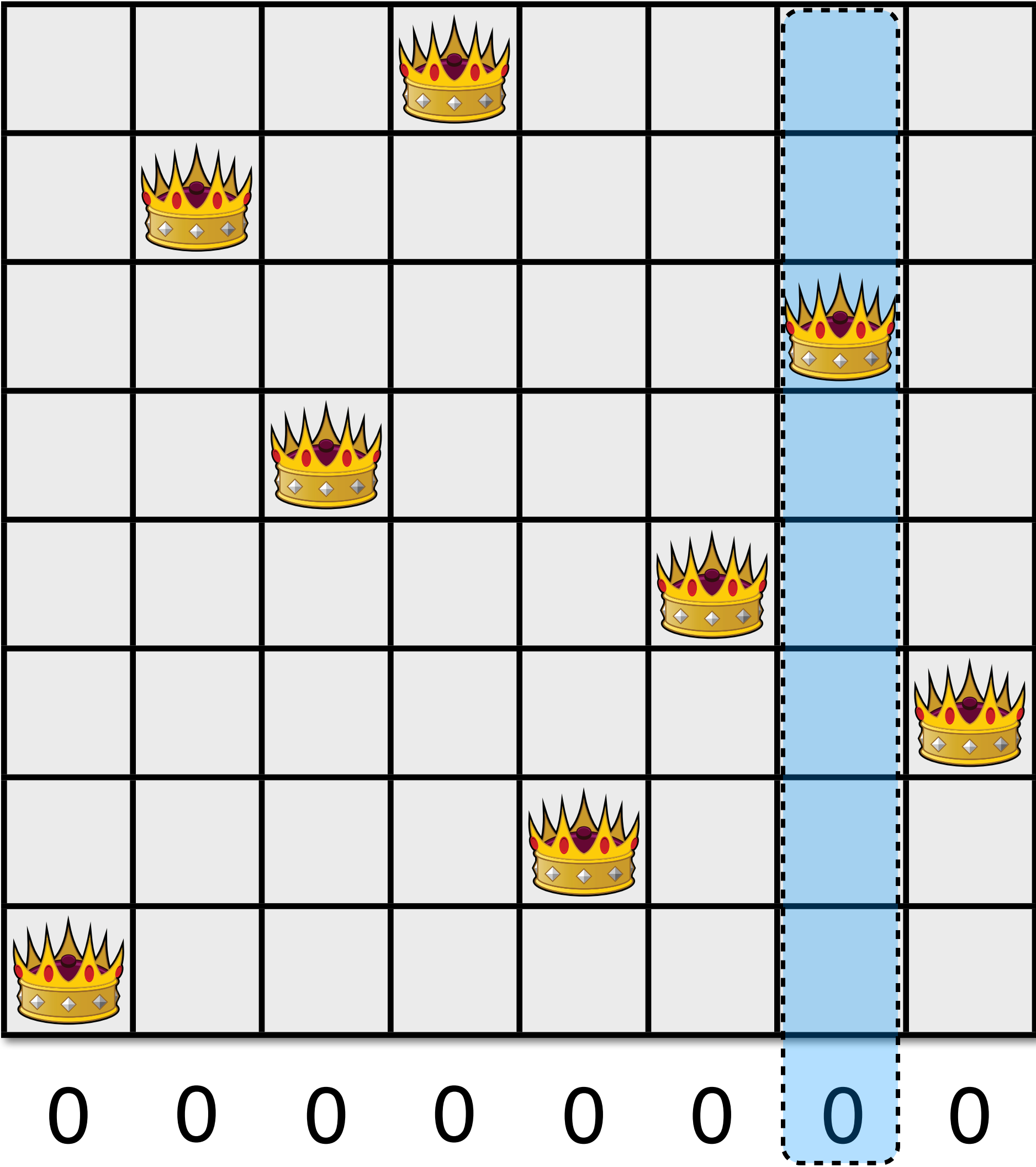


0 0 0 0 0 0 0 0 0

# Intuition



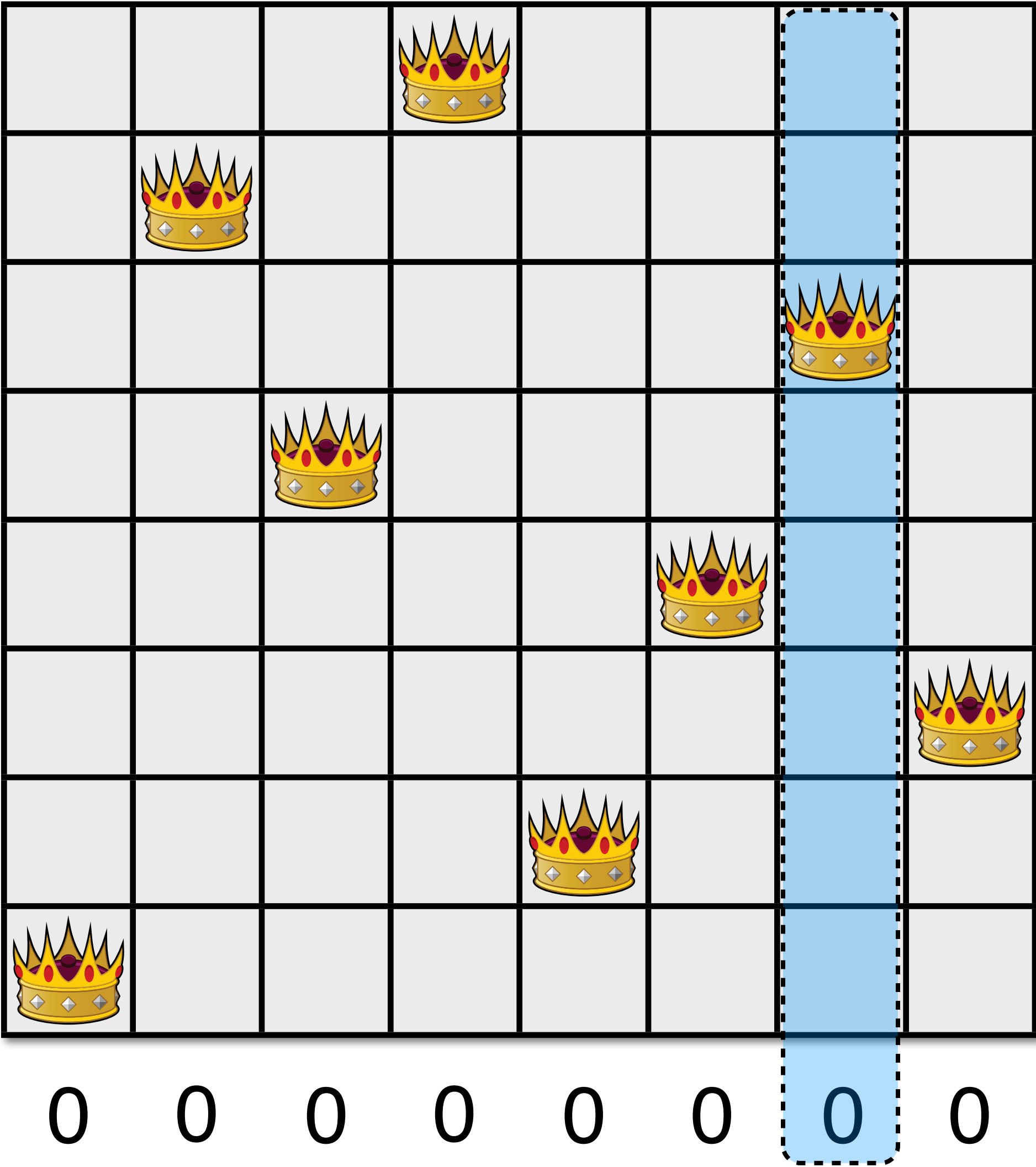
# Intuition



1  
2  
0  
3  
3  
2  
3  
2  
0



# Intuition



1  
2  
0  
3  
3  
2  
3  
2  
0



# Neighborhood

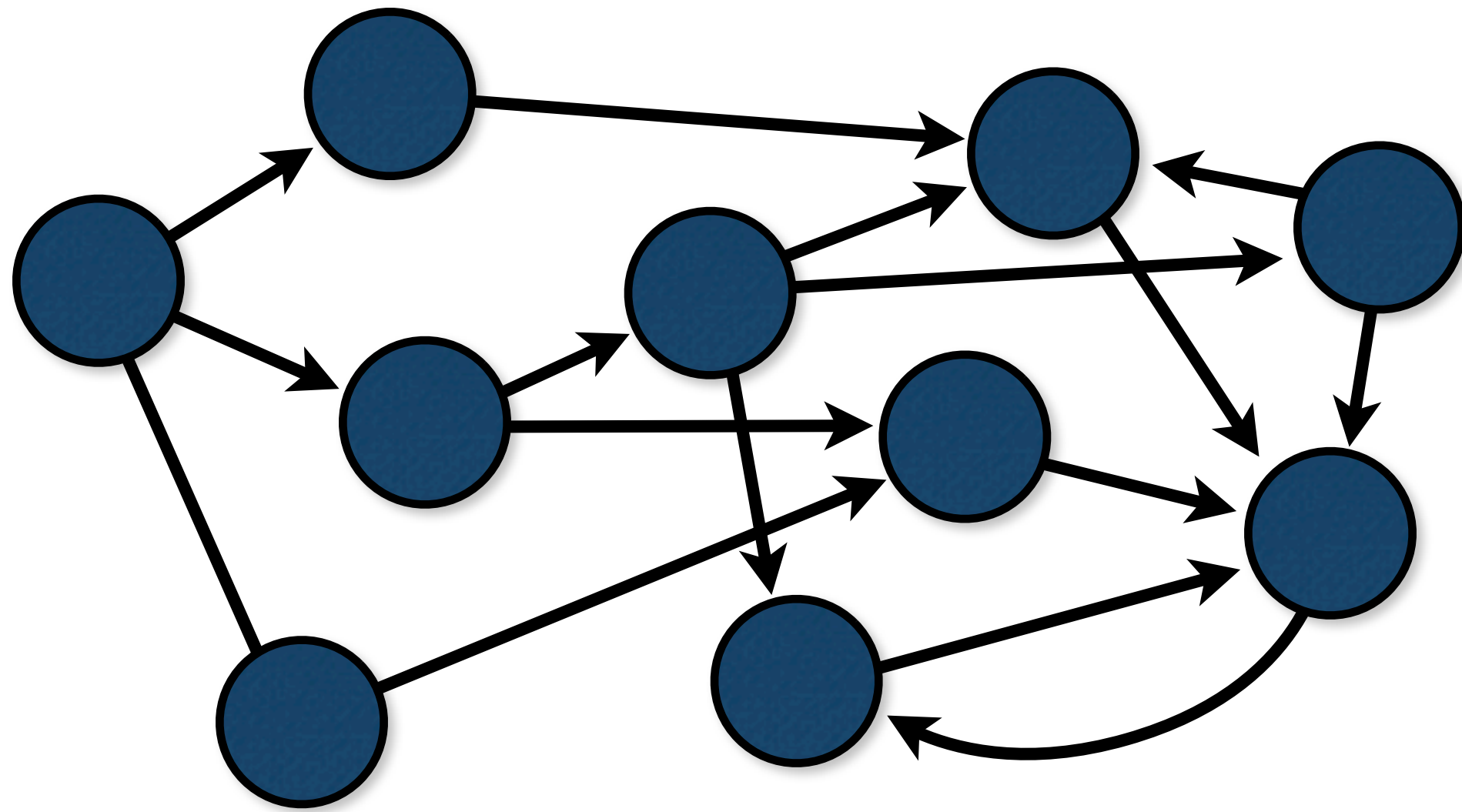
- Optimizing a function  $f$

# Neighborhood

- ▶ Optimizing a function  $f$
- ▶ Local moves define a neighborhood
  - configurations that are close
  - $N: C \rightarrow 2^C$

# Neighborhood

- ▶ Optimizing a function  $f$
- ▶ Local moves define a neighborhood
  - configurations that are close
  - $N: C \rightarrow 2^C$
- ▶ Local search is a graph exploration



# Local Minima and Local Search

## ► Local minima

- a configuration  $c$  is a local minima with respect to neighborhood  $N$  if

$$\forall n \in N(c) : f(n) \geq f(c)$$

# Local Minima and Local Search

## ► Local minima

- a configuration  $c$  is a local minima with respect to neighborhood  $N$  if

$$\forall n \in N(c) : f(n) \geq f(c)$$

## ► Local search

```
select a configuration c;  
I = { n in N(c) | f(n) < f(c) };  
while (|I| > 0) {  
    c = select a configuration from I;  
    I = { n in N(c) | f(n) < f(c) };  
}
```



# If you want guarantees, buy a toaster (C. Eastwood)

- ▶ Local minima

- a configuration  $c$  is a local minima with respect to neighborhood  $N$  if

$$\forall n \in N(c) : f(n) \geq f(c)$$

- ▶ No guarantees for global optimality

- escaping local optima is a critical issue in local search

# Local Search for Satisfaction

- ▶ The function  $f$  minimizes the constraint violations

# Local Search for Satisfaction

- ▶ The function  $f$  minimizes the constraint violations
- ▶ When  $f(c) = 0$ , the configuration  $c$  is feasible

# Local Search for Satisfaction

- ▶ The function  $f$  minimizes the constraint violations
- ▶ When  $f(c) = 0$ , the configuration  $c$  is feasible
- ▶ Constraint violation
  - 0/1: whether the constraint is violated
  - degree of violation

# Local Search for Satisfaction

- ▶ The function  $f$  minimizes the constraint violations
- ▶ When  $f(c) = 0$ , the configuration  $c$  is feasible
- ▶ Constraint violation
  - 0/1: whether the constraint is violated
  - degree of violation
- ▶ Variable violations
  - e.g., the number of violated constraints the variables appears in



Until Next Time