ITCS-6150 – Intelligent Systems

Project 2

Solving N-queens problem using hill-climbing search and its variants

Project Report

submitted by
Alaa Alharthi
Anisha Kakwani
Deep Prajapati
Heena Jain

At
The University Of North Carolina at Charlotte

Problem Formulation:

N-Queens:

The problem consists of placing N queens on an N x N chessboard so that no two queens can capture each other. No two queens are allowed to be placed on the same row or the same column or the same diagonal.

Hill Climbing Search:

The hill-climbing search algorithm (steepest-ascent version) is simply a loop that continually moves in the direction of increasing value (uphill). The algorithm terminates when no neighbor has a value greater than the current node. Also, it doesn't allow for evaluation beyond the immediate neighbors of the current state

Hill Climbing Search with sideways moves:

The hill-climbing search with sideways moves works similar to hill-climbing search, However, it is allowed sideways move when there is a plateau that might be a shoulder so keep going sideways when there is no uphill move.

Random Restart Hill-Climbing Search:

The random restart hill-climbing search runs hill-climbing search from a random initial state and restarts till the time it finds a solution. The algorithm eventually generates the goal state as the initial state.

General Working Process of Hill-Climbing Search:

- Evaluate the initial state. If it is the final state, stop, and return success. If it is not, make it the current state.
- Loop until a solution is found or there is no new neighbor to evaluate.
- Apply an evaluation function to the current state.
- Check new state:
    - If it is the goal state, then return success and exit.
    - Else if it is better than the current state then set the new state as a current state.
    - Else if not better than the current state, then return to step 2.

Problem Elements:

Problem statement:
8-Queen is an 8 x 8 board with 8 queens distributed randomly.

Goal:
Arranging the board such that no two queens are allowed to be placed on the same row, the same column, or the same diagonal.

Heuristic function:
Compute the number of conflicts between all the queens. In this problem, we are trying to minimize heuristic value to reduce the number of conflicts.

Actions:
Moving the queen vertically on the board either upward and downward.

Method:
Hill climbing search that evaluates the current states and its successors, The successor is selected for further expansion based on the evaluation function. Neither the parent nor the neighbors are retained.

Program Structure:

1. **Procedure of heuristic evaluation:**

   def calc_heuristic(board,n):
   This function is used to calculate the heuristic value of a particular configuration. It will find the position of the queen, and check each pair of queens are in the same column or row or diagonal; If yes it will increase the heuristic value by one.

   def find_all_heuristics(board,n):
   This function is used to calculate the heuristic value of all n*(n-1) possible configurations reachable from the current configuration. It calls *calc_heuristic()* for all n*(n-1) possible configurations and it returns a matrix of possible heuristic value.

   def find_minimum_heuristics_move(board,n):
   This function calculates the heuristic value of each state by calling the find_all_heuristics function, evaluates the heuristic value for each state, and returns the one with the minimum value

2. **Functions of Hill Climbing and its variants:**

   def hill_climbing(heuristic_current,min_heuristic):
   This is the first approach towards the Hill-Climbing search. It takes the heuristic value of the current state and compares it with the minimum heuristic value. If it is better than the minimum heuristic value it will return true which means it will switch the queen to a new position where it reduces the number of conflicts otherwise it will return false and not switch the queen position.

   def hill_climbing_sideways(heuristic_current,min_heuristic):
   This is the second approach towards the Hill-Climbing search. It takes the heuristic value of the current state and compares it with the minimum heuristic value. If it is equal to the minimum heuristic value that means a sideways move is taken and it will increment the value of the counter, when the counter value exceeds the number of sideways moves allowed the counter is set to 0. If the value of the current heuristic is better than the minimum heuristic and the counter has not exceeded the number of sideways moves allowed it will return true and the queen position will be switched. The number of sideways moves allowed is 100 for *Hill-Climbing with Sideways Moves* and 10 for *Random Restart Hill-Climbing search*.

3. **Main method:**

In the main method, the program asks the user for the size of the N-Queen board, and the hill-climbing search variants.

- The program generates N x N boards based on the user input and distributes the queens randomly.
- It computes the heuristic for all N*(N-1) possible configurations reachable from the current configuration by calling *find_all_heuristics().*
- The program runs no_of_instances times which is set at 100 (by default) to find the average success/failure rates, and an average number of steps for success/failure/restart depending on the approach selected.
- In each iteration, the program calls the variant function of the hill-climbing algorithm that the user has chosen.
- The success rate increases when the heuristic value of the current state equals zero which means no queens in attacking position. The program checks this condition each iteration hill-climbing search runs.
- The failure rate equals 100 - success rate. When the program failed to find the solution and reached the limit of running the algorithm (n times limit).
- If the user chooses to run the **hill-climbing classic**, the program will call *hill_climbing()* and maintain the same process above meaning it stops either when it finds the solution or when it reaches the limit of running the algorithm (n times limit).
- If the user chooses to run the **hill climbing with sideways moves**, the program will call *hill_climbing_sideways()* and maintain the same process above meaning it stops either when it finds the solution or when it reaches the limit of running the algorithm (n times limit).
- If the user chooses to run the **Random restart without sideways moves**, the program will *call hill_climbing()*, and it keeps running until it a success which means the algorithm will not stop until it succeeds.
- If the user chooses to run the **Random restart with sideways moves**, the program will be *hill_climbing_sideways()*, and it keeps running until it a success which means the algorithm will not stop until it succeeds.

4. Other Functions:

```
def print_board(msg,board):
```
This function used to print board configuration.

```
def get_Queen_Position(board,n,i):
```
This function is used to get the queen position.

```
def generate_board(n):
```
This function is used to generate the board with a random position of queens

5. **Global Variables:**

   no_of_instances:
   This variable stores the value of the number of instances, i.e., the number of times the code will run to generate the average success and failure rates.

   steps_success:
   This variable stores the number of successful steps taken by each instance

   steps_failure:
   This variable stores the number of failure steps taken by each instance

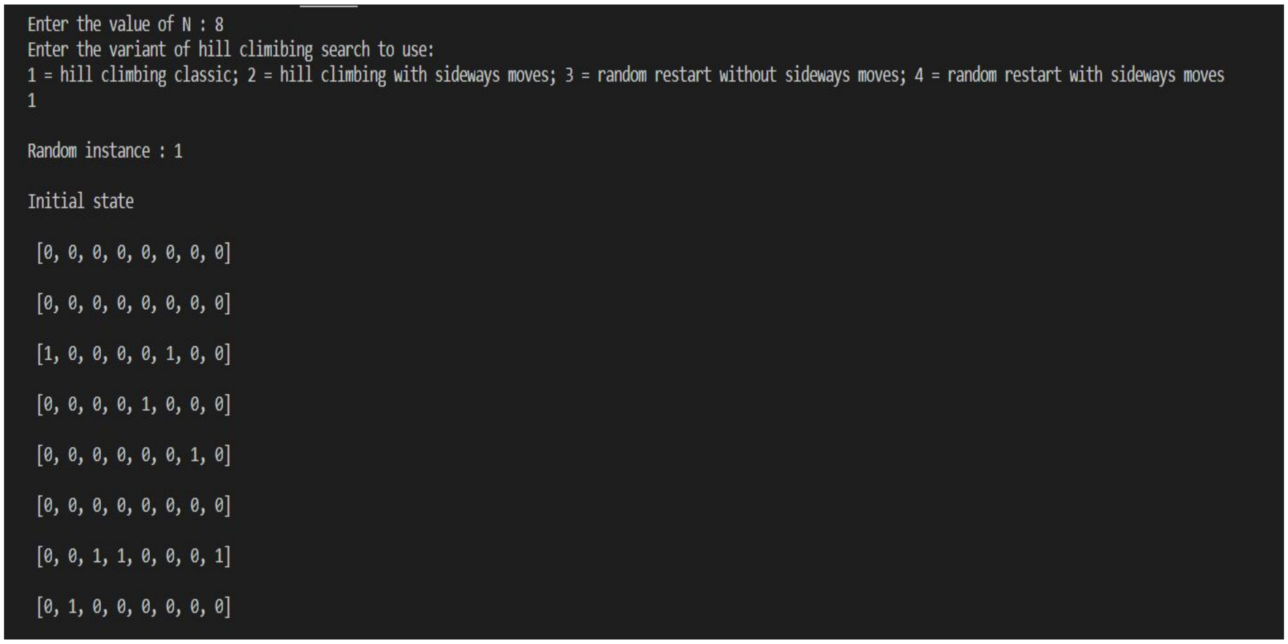
   no_of_random_restart:
   This variable calculates the number of random restarts taken by a random restart hill-climbing approach, both with and without sideways moves.

Project Analysis/ Conclusion:

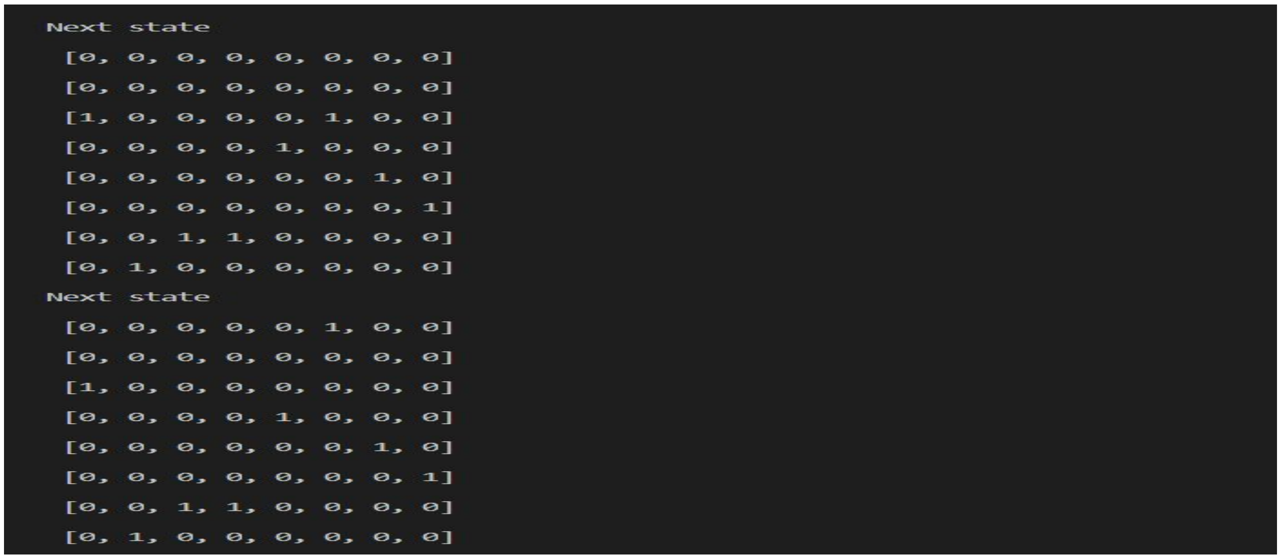
An implementation of the variants of hill-climbing search was performed using Python programming language to solve the N-queens problem. A heuristic function was used to evaluate each board configuration. The value of the heuristic function for any board configuration is equal to the number of pairs of queens attacking each other in a particular state. In total, 4 variants of the steepest descent algorithm were considered: hill-climbing without sideways moves, hill climbing with sideways moves, random restart hill climbing without sideways moves, and random restart hill climbing with sideways moves. The results show that hill climbing with sideways moves has an approximate efficiency of 95% whereas, without sideways moves, the efficiency is only 11%. Random restart is a technique that gives a 100% success rate because the algorithm does not stop until it has reached/generated a goal state. Random restart with sideways moves has an average number of restarts for reaching a goal state as 2 and the average number of steps is 21, whereas for random restart without sideways moves has an average number of restarts for reaching a goal state as 8 and the average number of steps is 27. Allowing sideways moves in hill climbing gives better results.

Program Output (4 approaches):

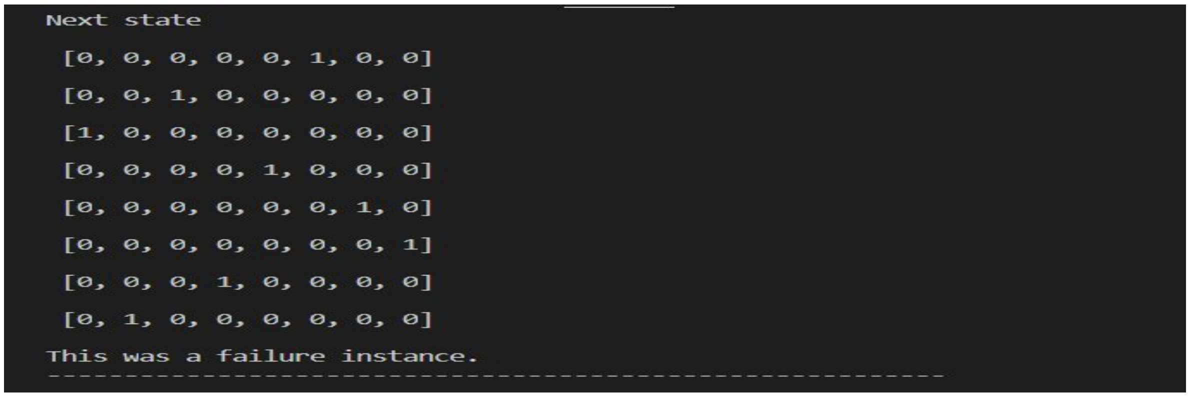
1. **Classic Hill-Climbing Search**



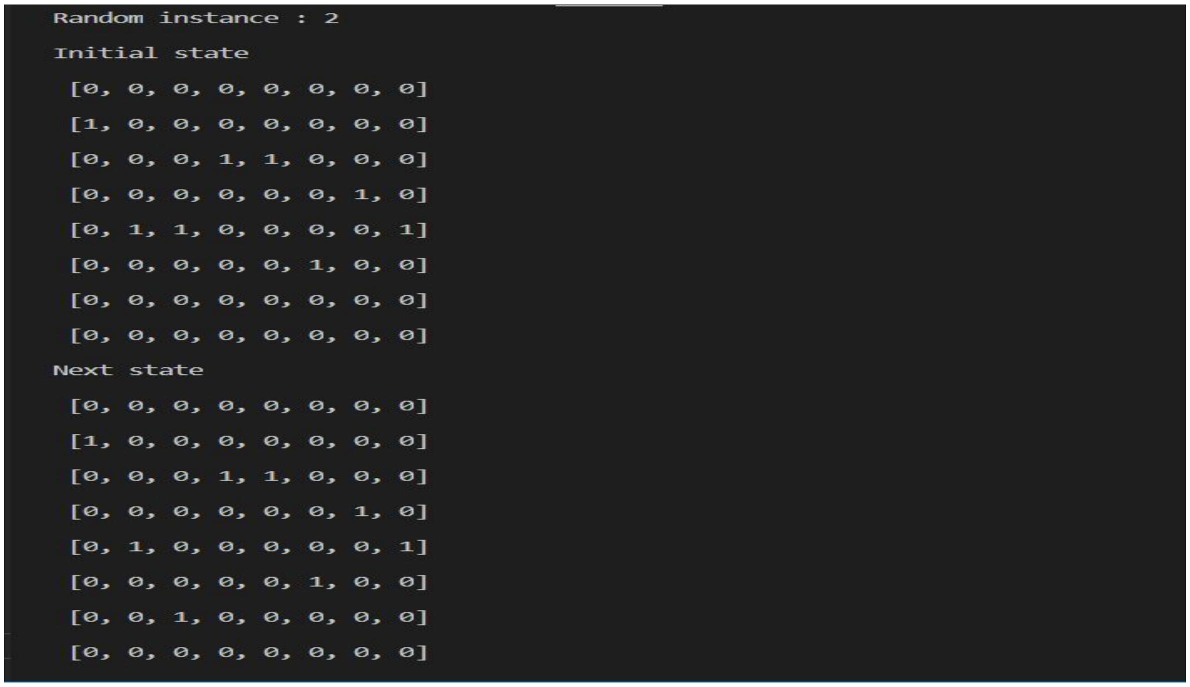
```
Enter the value of N : 8
Enter the variant of hill climibing search to use:
1 = hill climbing classic; 2 = hill climbing with sideways moves; 3 = random restart without sideways moves; 4 = random restart with sideways moves
1

Random instance : 1

Initial state

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 1, 0, 0, 0, 1]

 [0, 1, 0, 0, 0, 0, 0, 0]
```



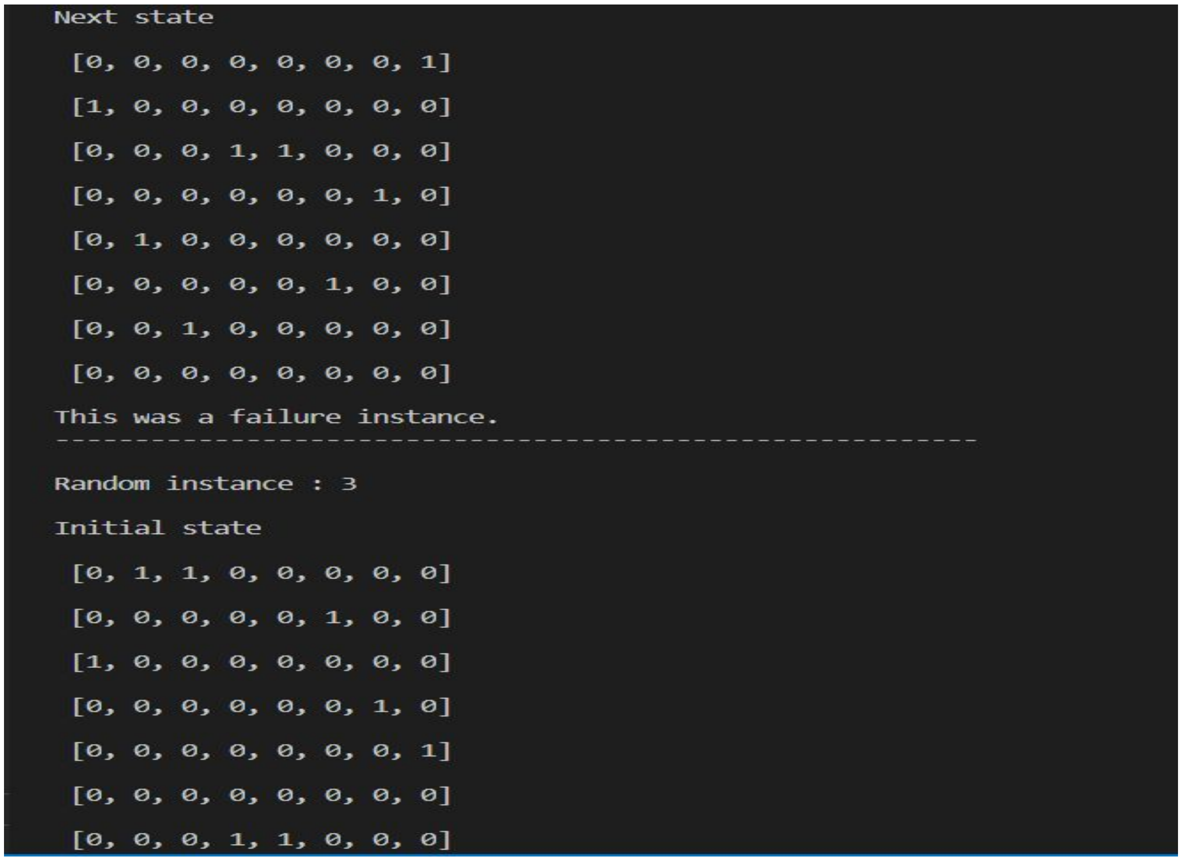
```
 Next state
  [0, 0, 0, 0, 0, 0, 0, 0]
  [0, 0, 0, 0, 0, 0, 0, 0]
  [1, 0, 0, 0, 0, 1, 0, 0]
  [0, 0, 0, 0, 1, 0, 0, 0]
  [0, 0, 0, 0, 0, 0, 1, 0]
  [0, 0, 0, 0, 0, 0, 0, 1]
  [0, 0, 1, 1, 0, 0, 0, 0]
  [0, 1, 0, 0, 0, 0, 0, 0]
 Next state
  [0, 0, 0, 0, 0, 1, 0, 0]
  [0, 0, 0, 0, 0, 0, 0, 0]
  [1, 0, 0, 0, 0, 0, 0, 0]
  [0, 0, 0, 0, 1, 0, 0, 0]
  [0, 0, 0, 0, 0, 0, 1, 0]
  [0, 0, 0, 0, 0, 0, 0, 1]
  [0, 0, 1, 1, 0, 0, 0, 0]
  [0, 1, 0, 0, 0, 0, 0, 0]
```



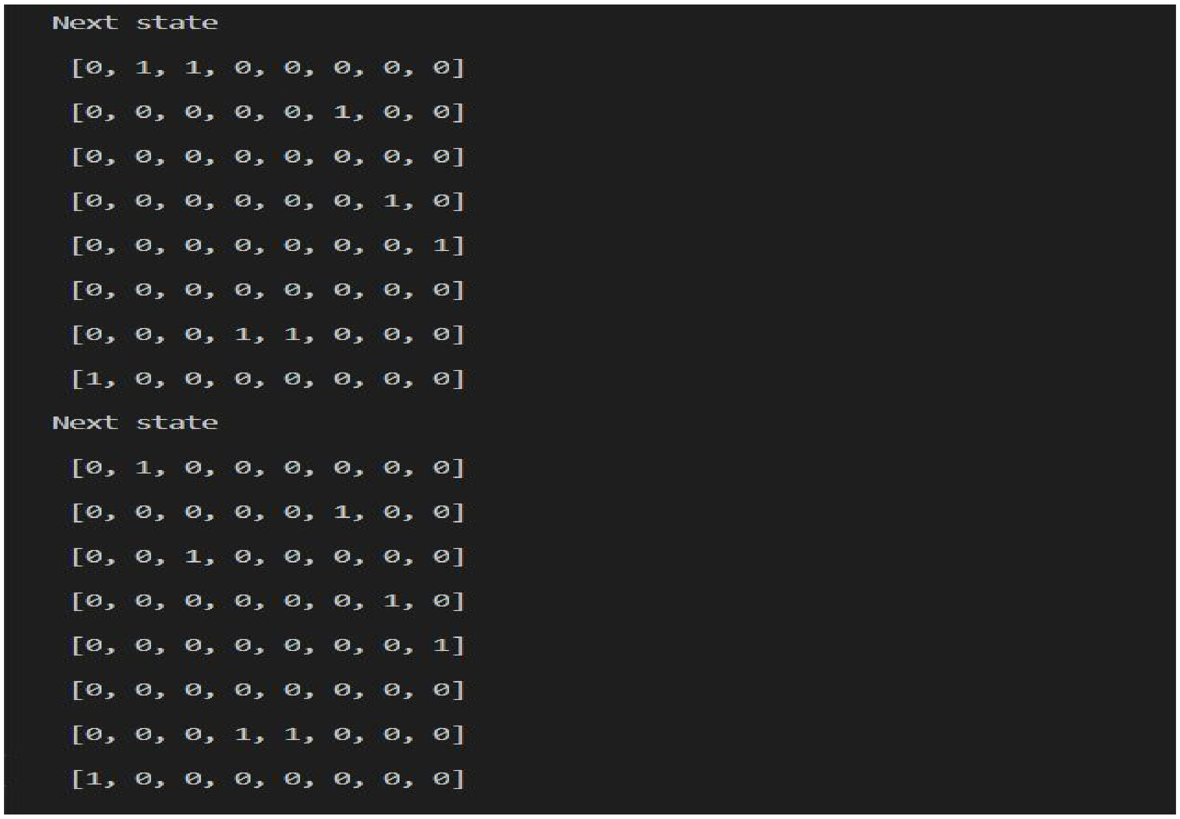
```
Next state
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
This was a failure instance.
------------------------------------------------------------
```



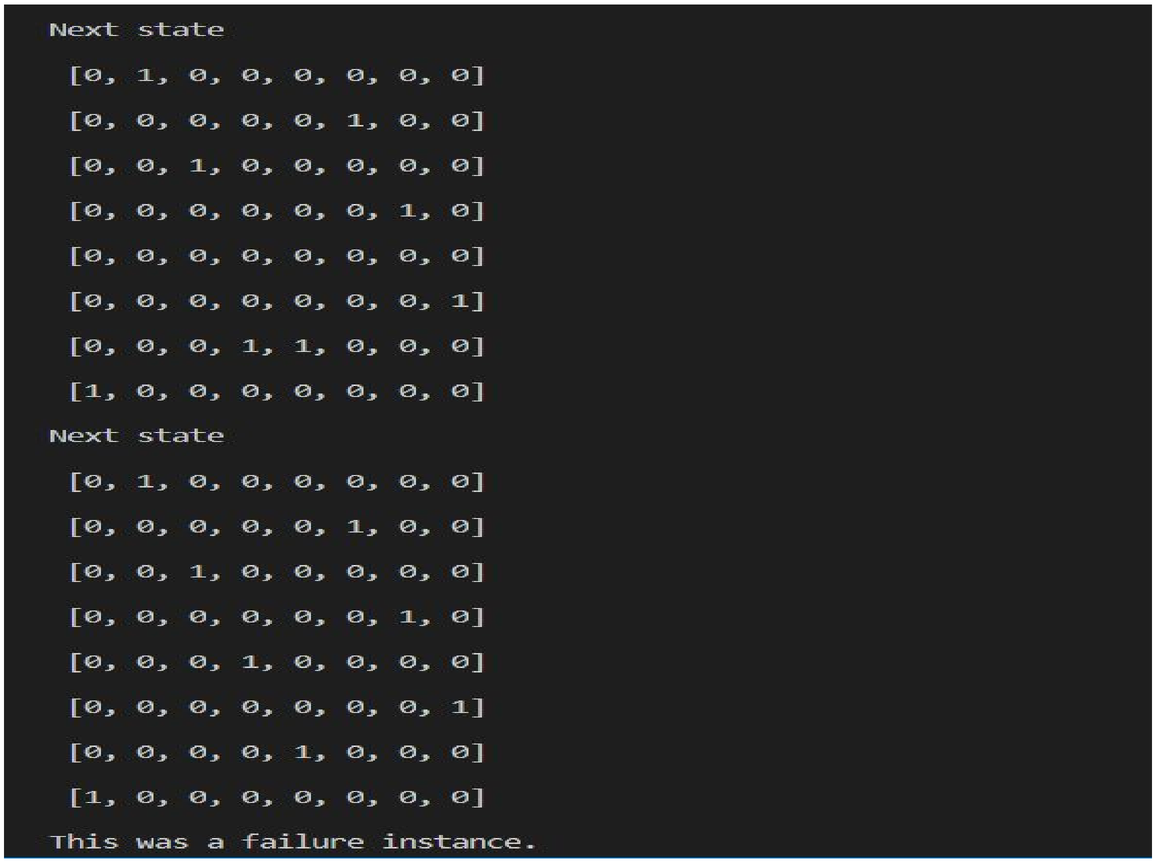
```
Random instance : 2
Initial state
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 1, 1, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 1, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
```



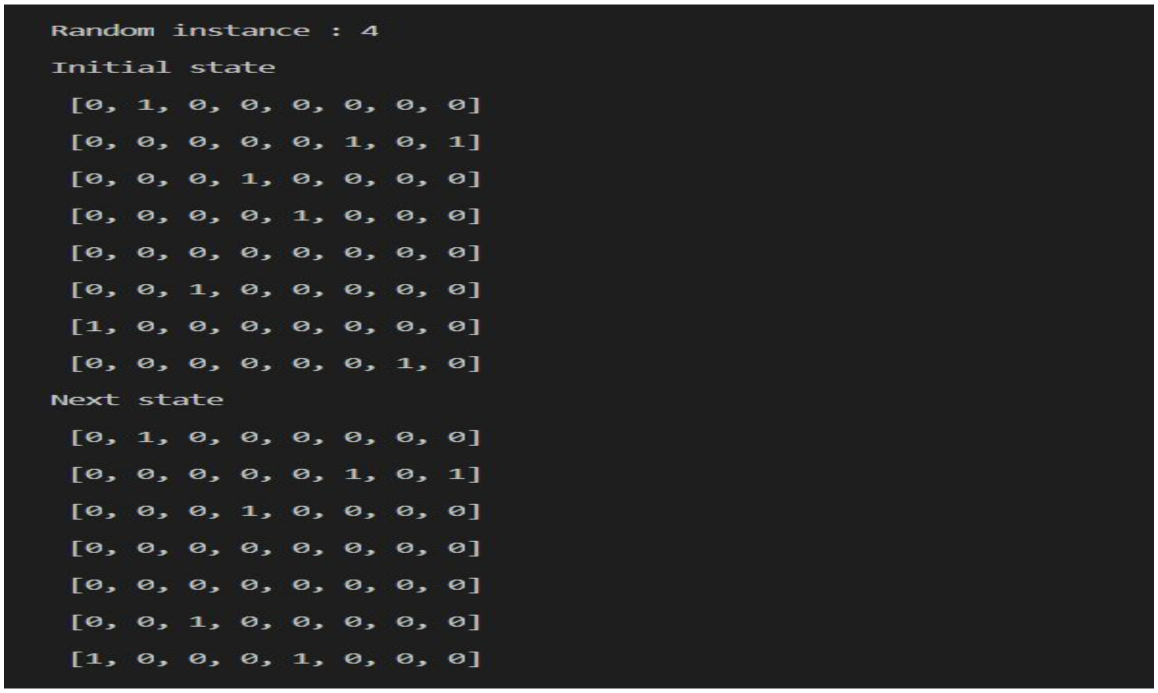
```
Next state
 [0, 0, 0, 0, 0, 0, 0, 1]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
This was a failure instance.
----------------------------------------------------------

Random instance : 3
Initial state
 [0, 1, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 1, 0, 0, 0]
```



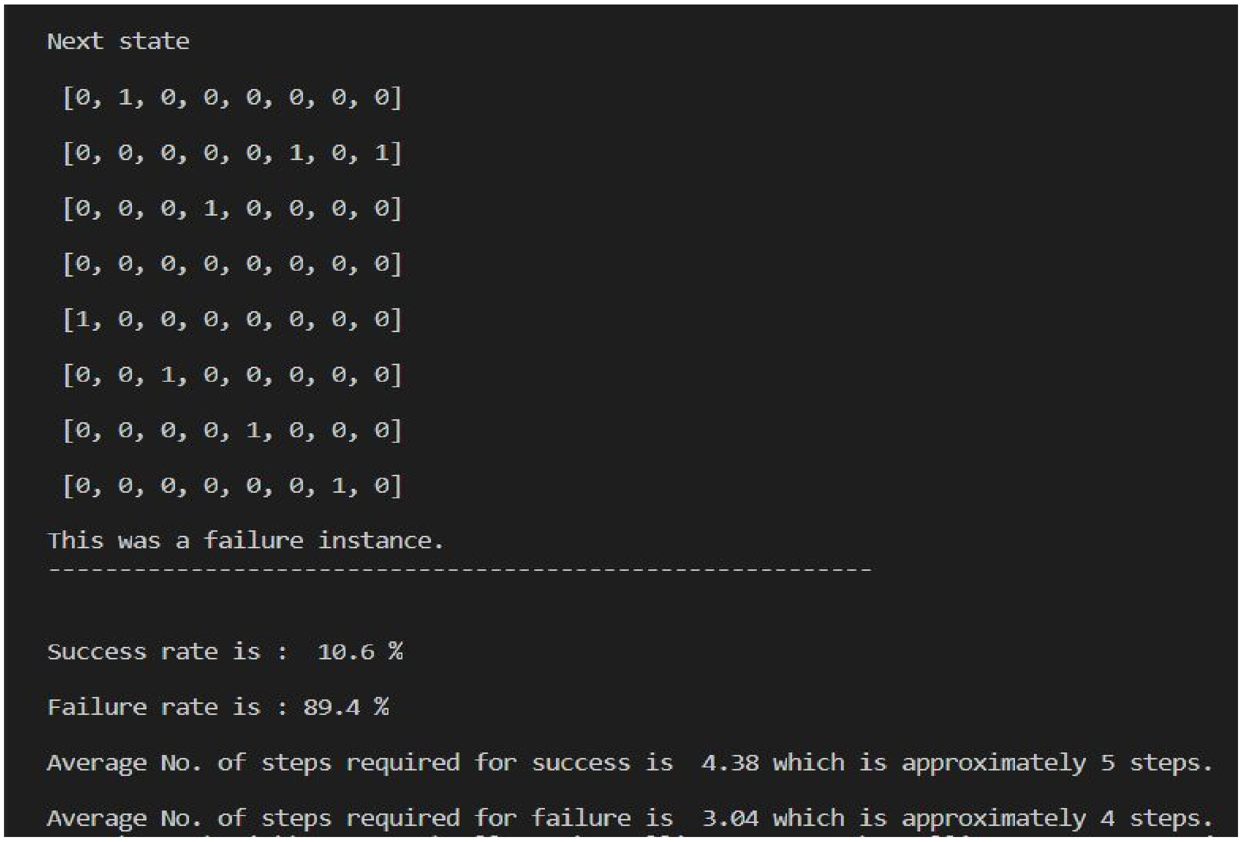
```
Next state
 [0, 1, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
```



```
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 1, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
This was a failure instance.
```

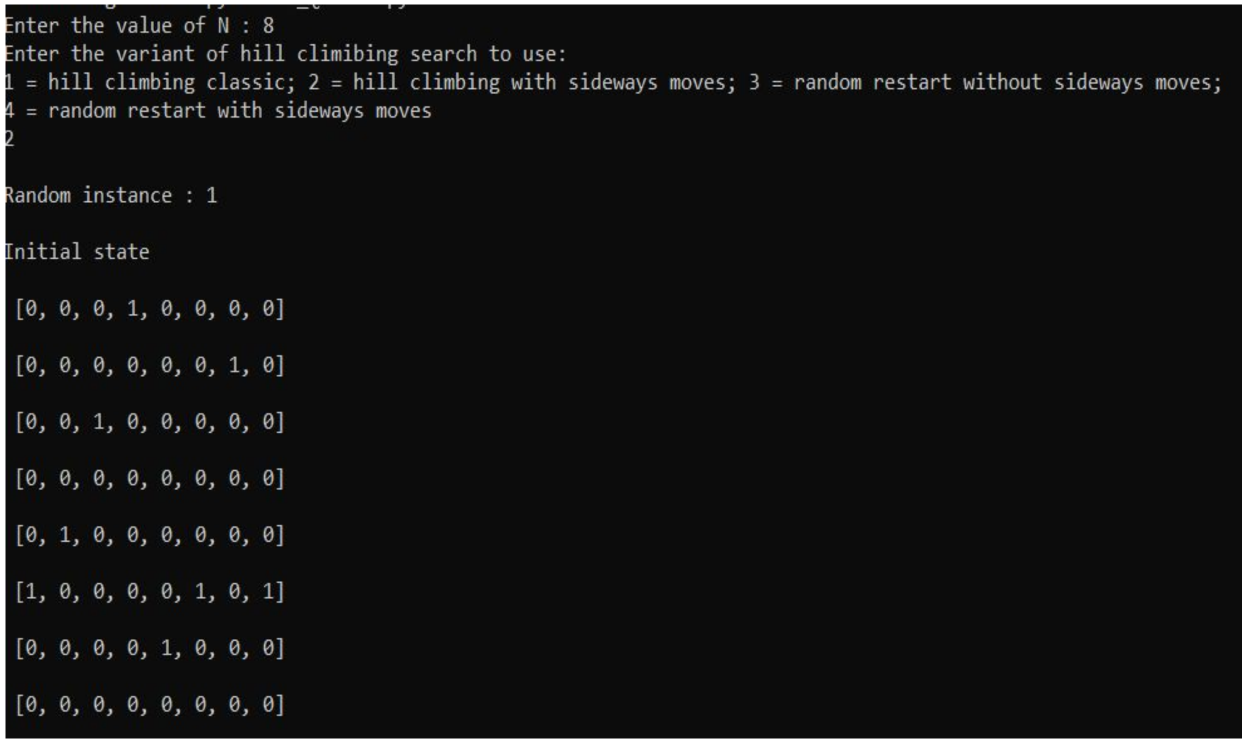


```
Random instance : 4
Initial state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 1]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 1]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 1, 0, 0, 0]
```

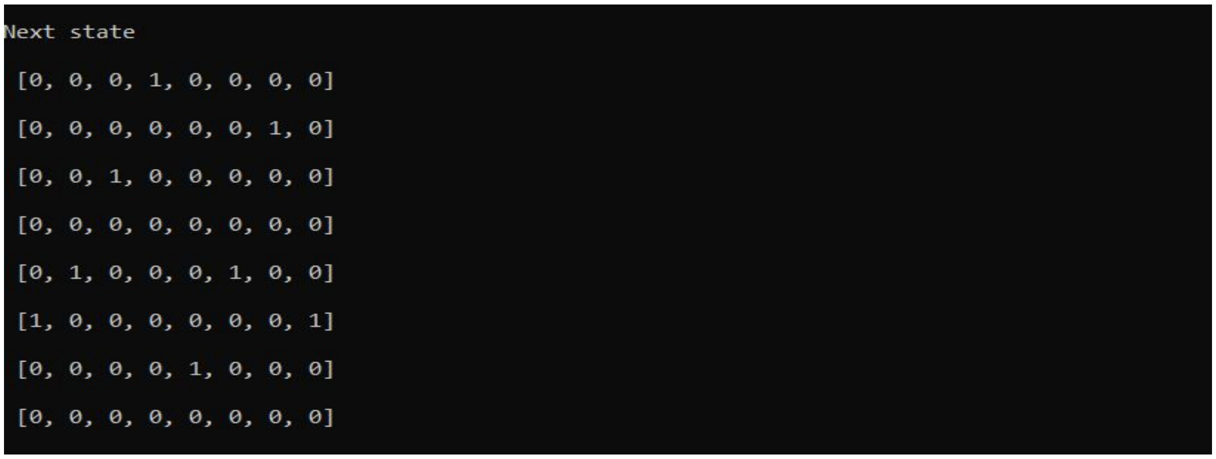


```
Next state

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 1]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

This was a failure instance.
----------------------------------------------------------


Success rate is :  10.6 %

Failure rate is : 89.4 %

Average No. of steps required for success is  4.38 which is approximately 5 steps.

Average No. of steps required for failure is  3.04 which is approximately 4 steps.
```

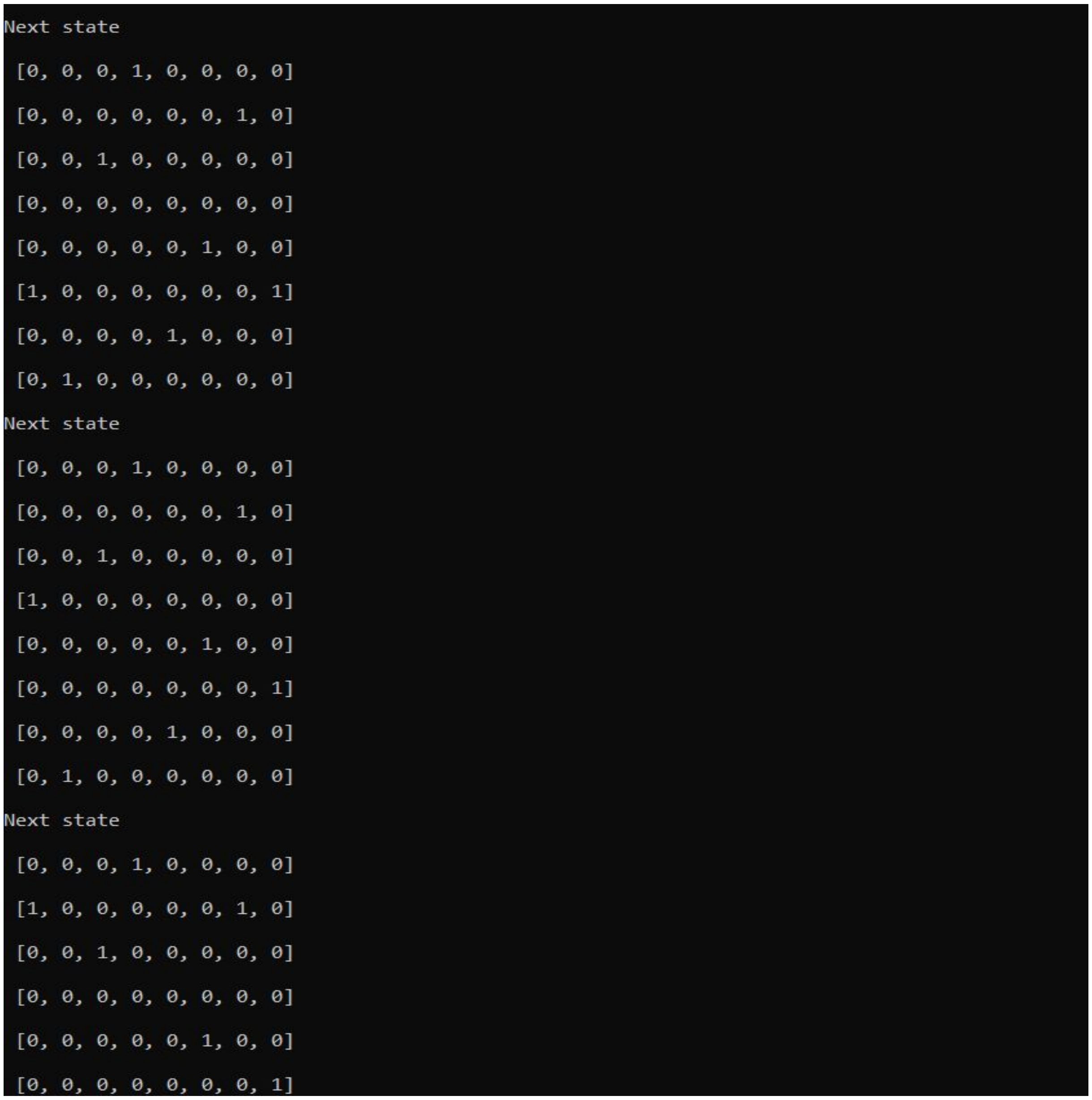
2. **Hill Climbing with sideways moves**



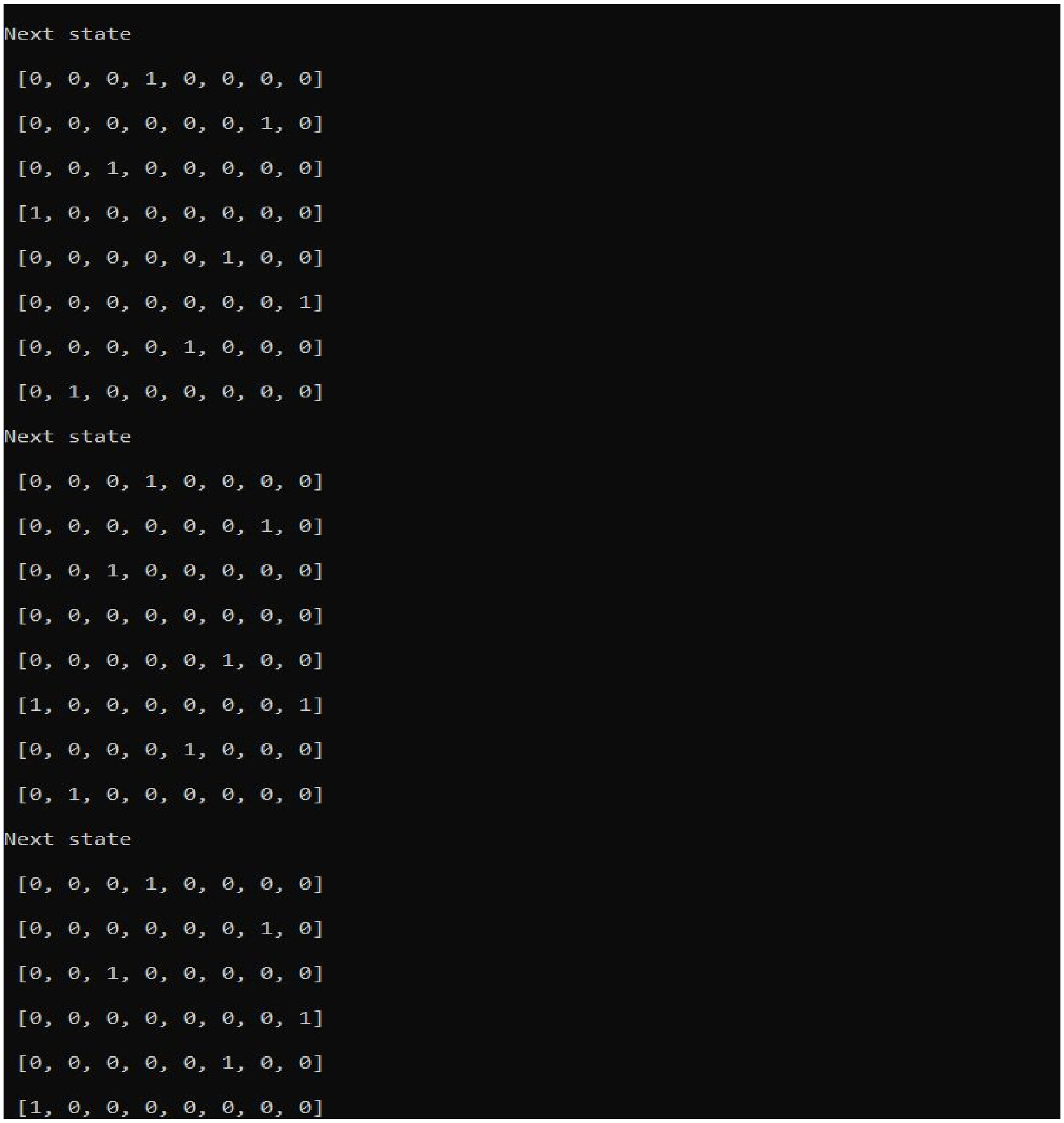
```
Enter the value of N : 8
Enter the variant of hill climibing search to use:
1 = hill climbing classic; 2 = hill climbing with sideways moves; 3 = random restart without sideways moves;
4 = random restart with sideways moves
2

Random instance : 1

Initial state

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 1, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]
```

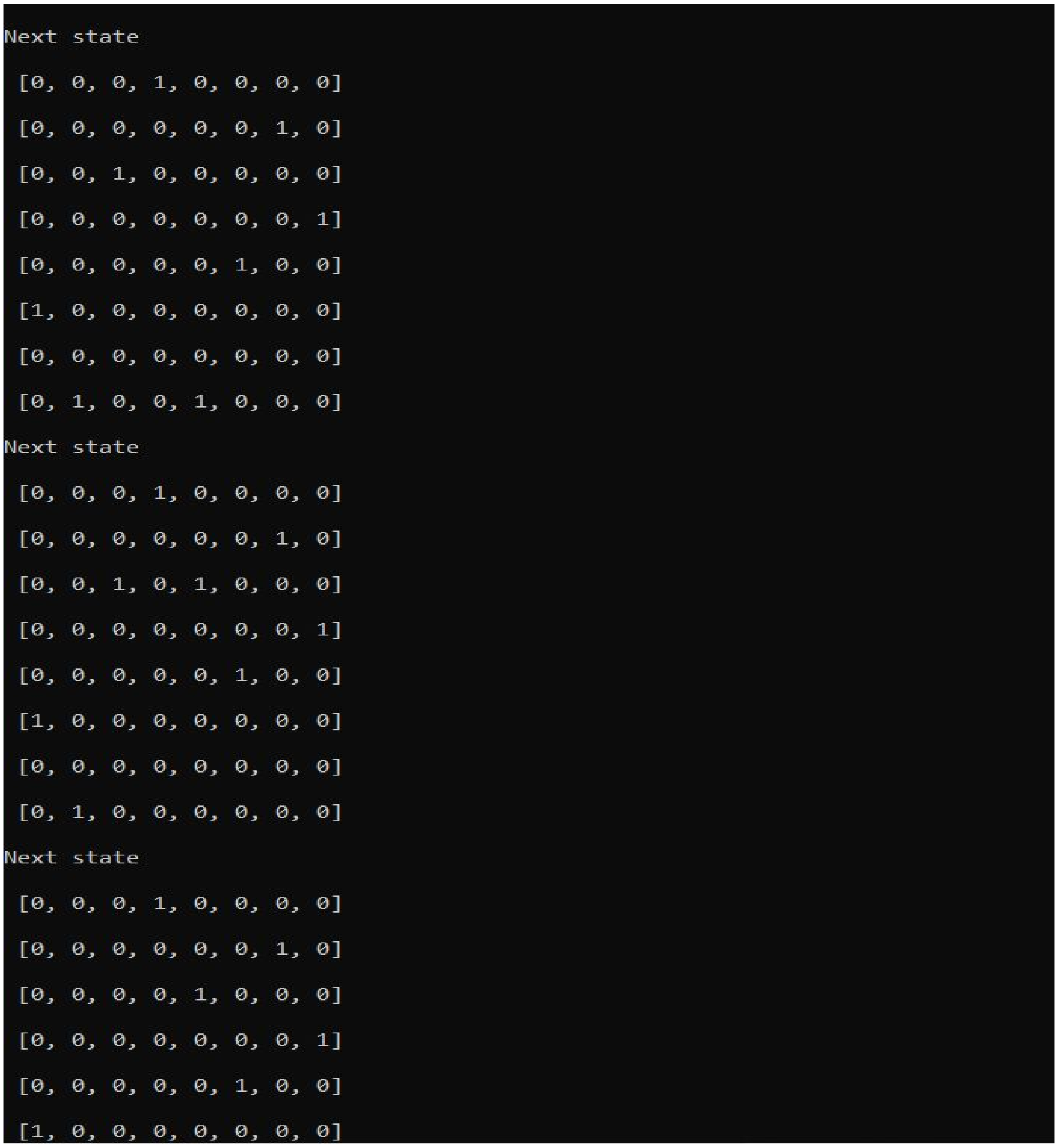


```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
```

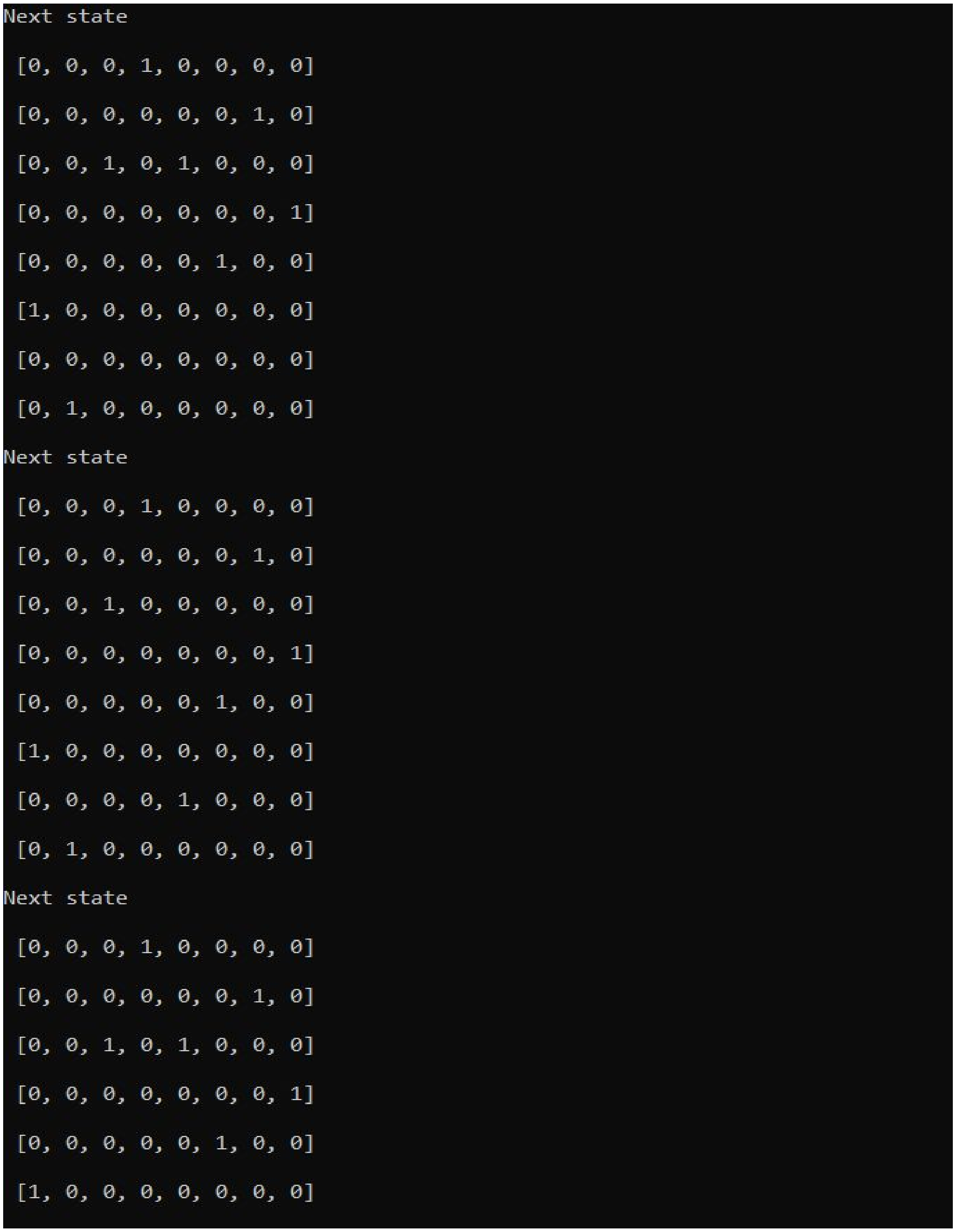


```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
```

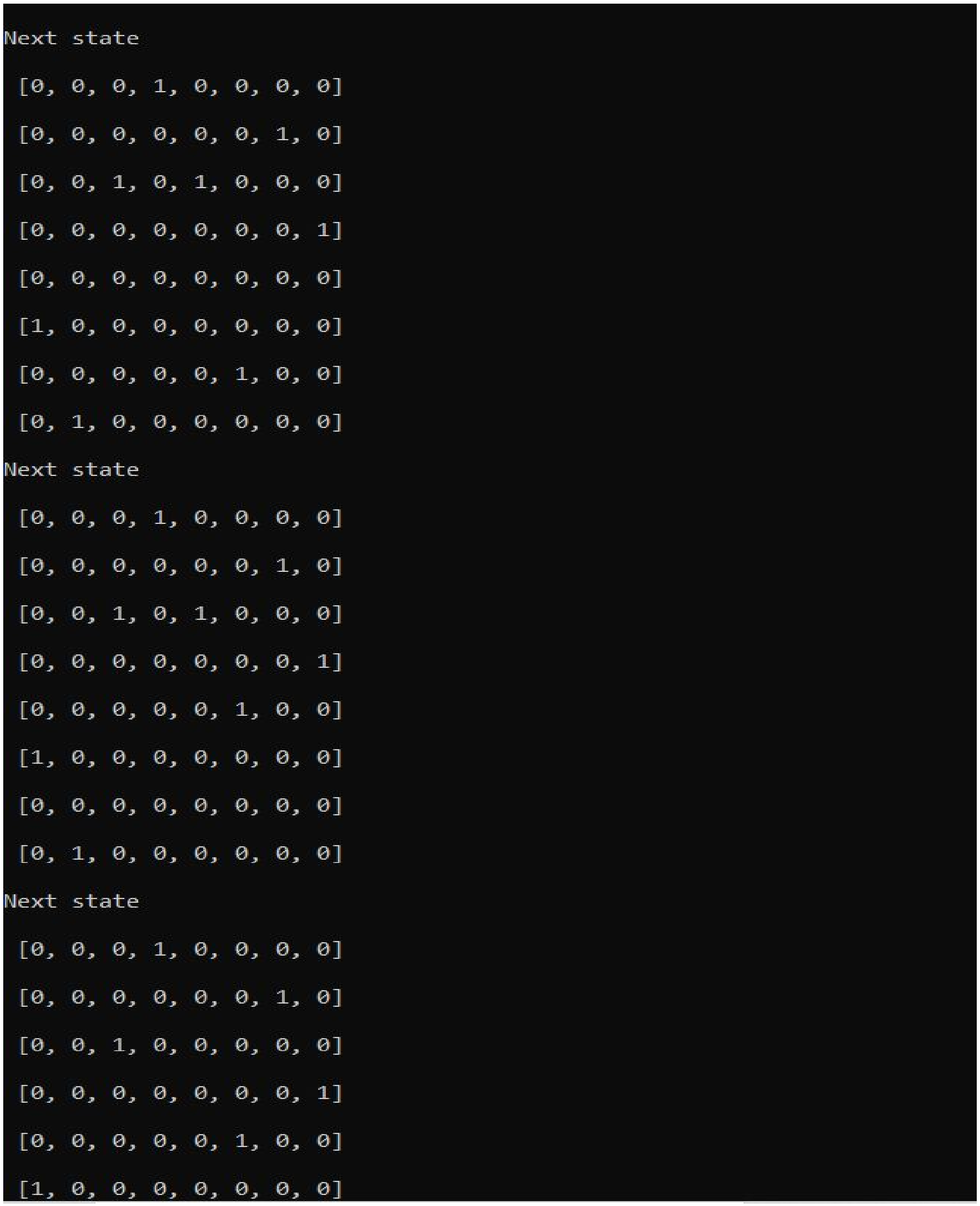

```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
```



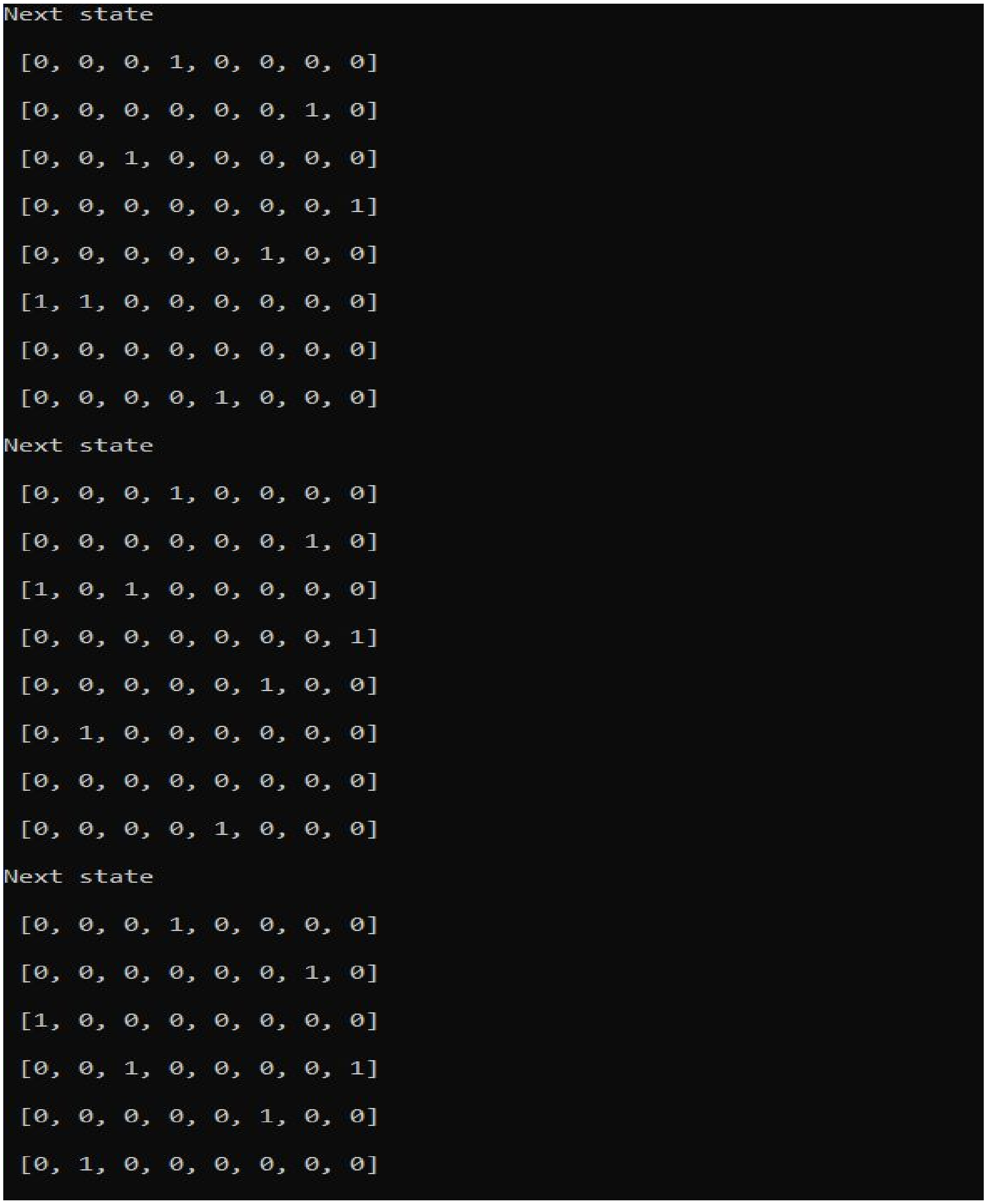
```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 1, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
```

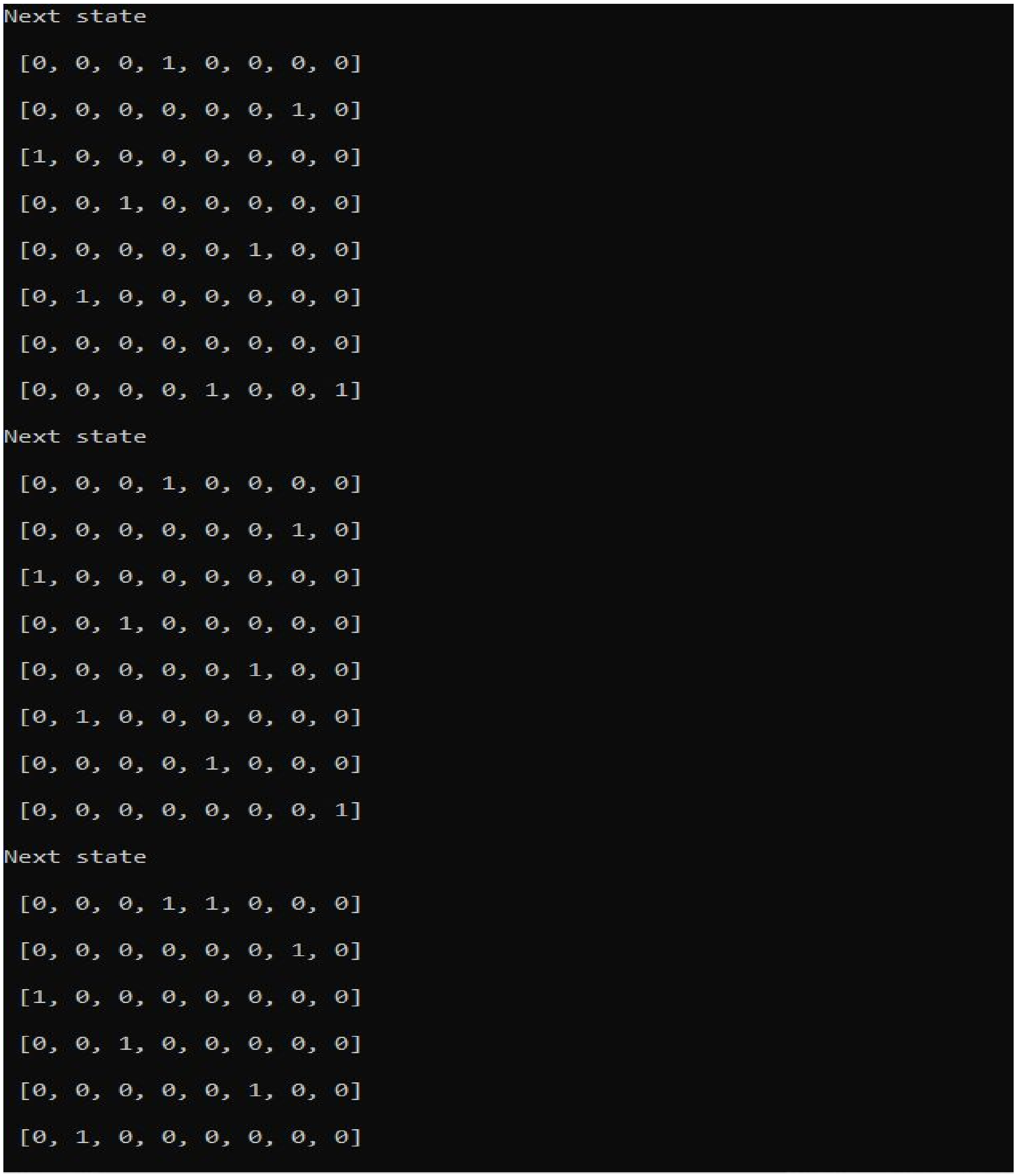


```
Next state

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]
Next state

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]
Next state

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]
```

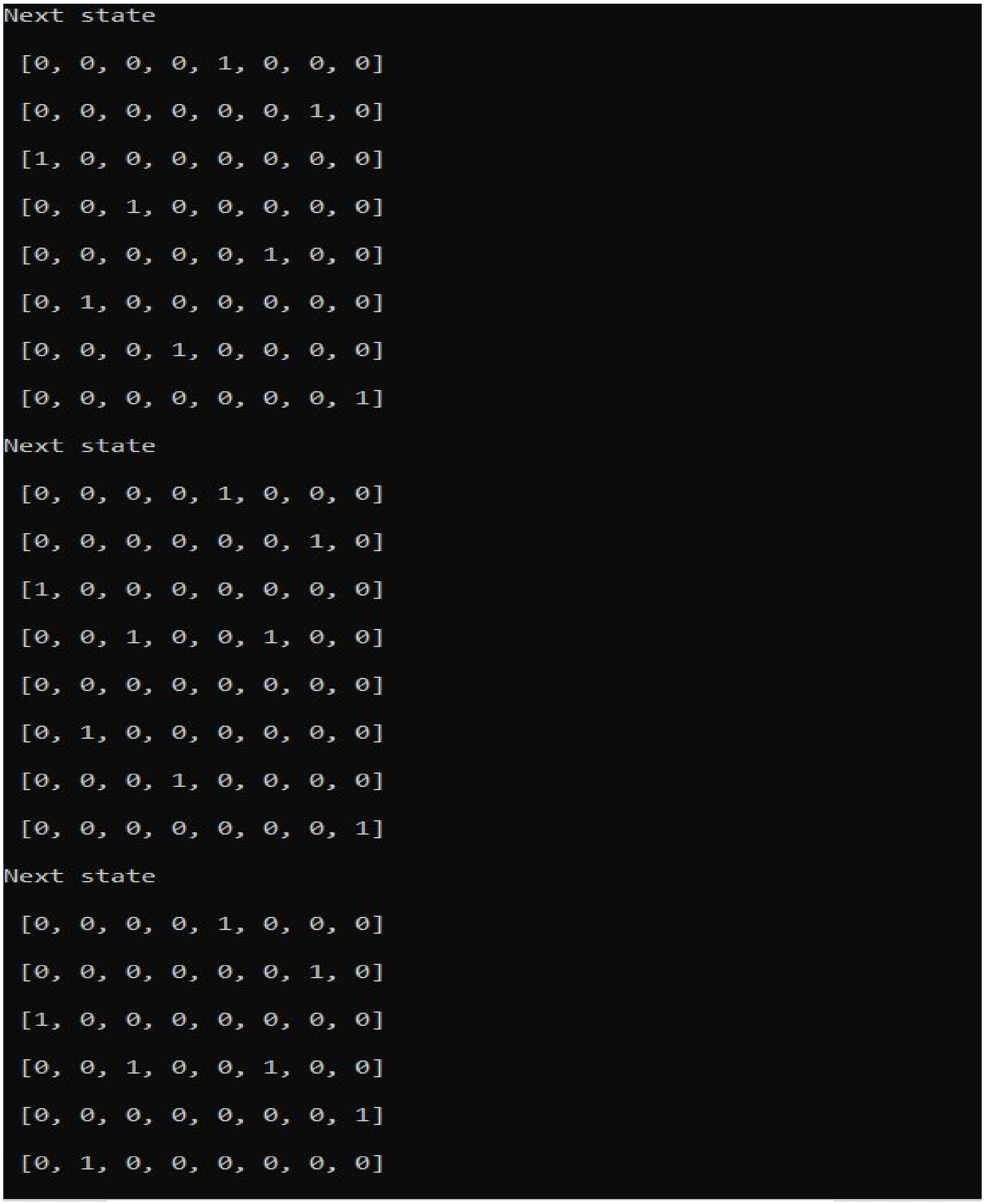


```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
```

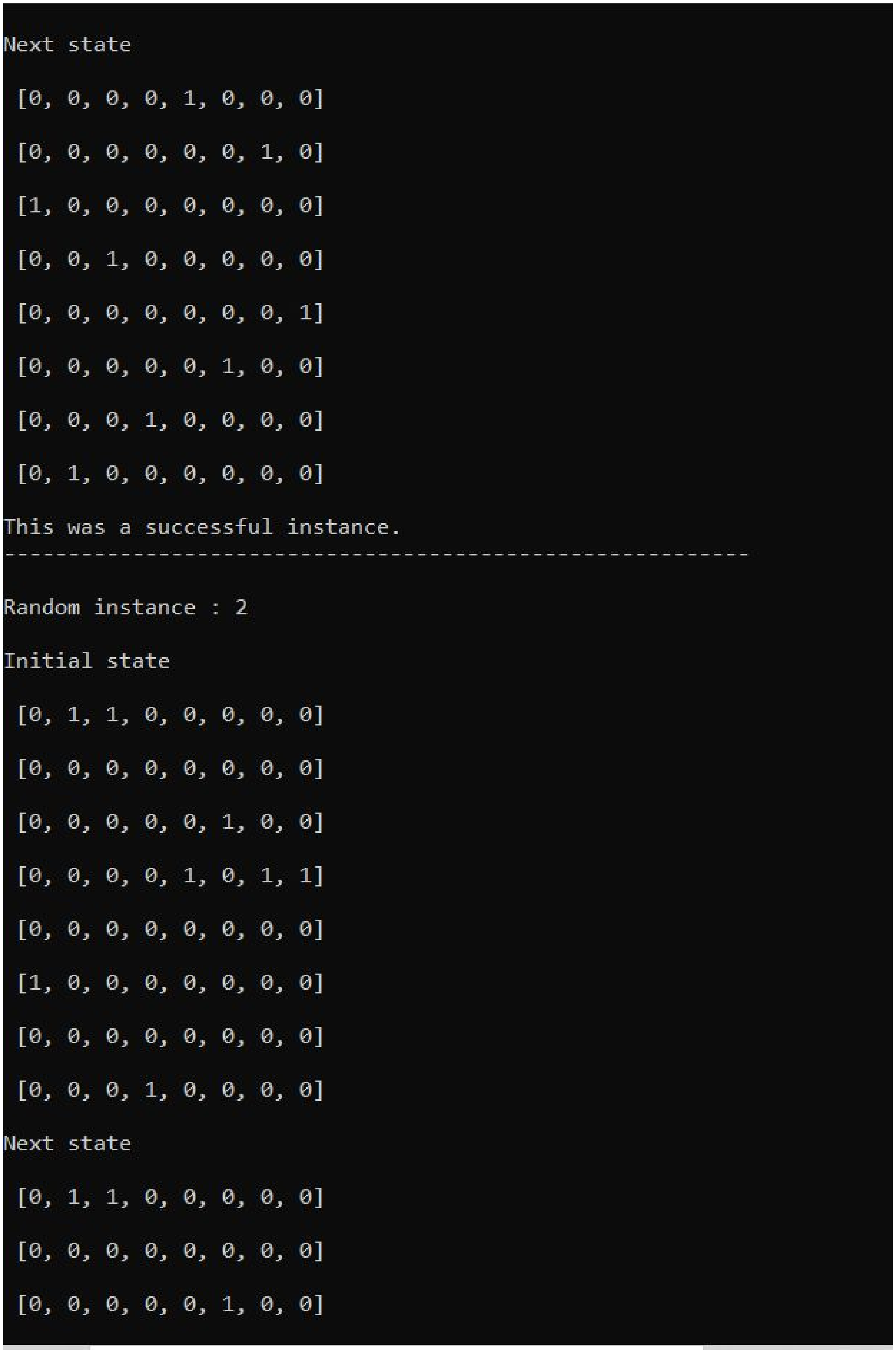

```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [1, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
```



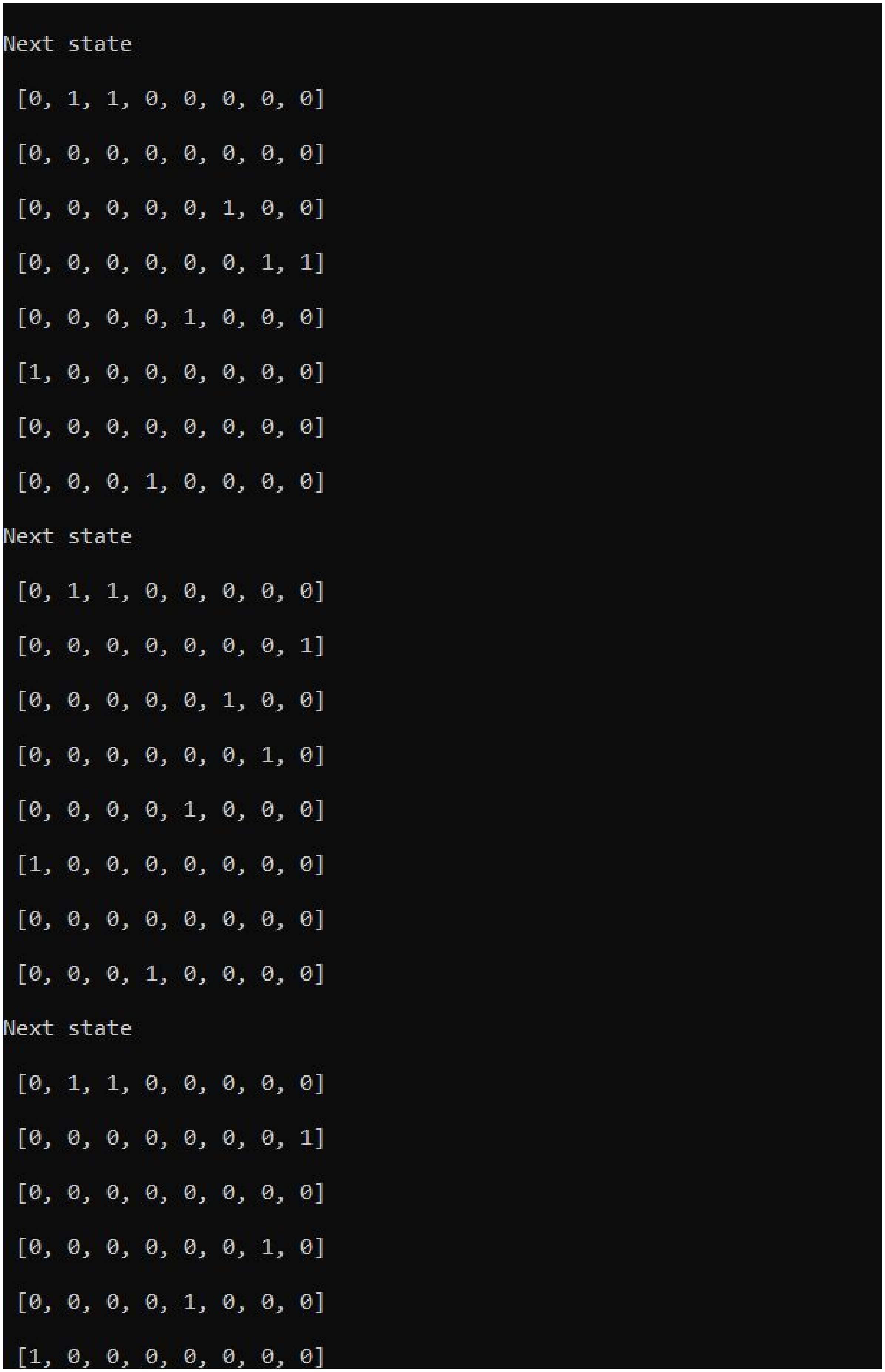
```
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 1]
Next state
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
Next state
 [0, 0, 0, 1, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
```



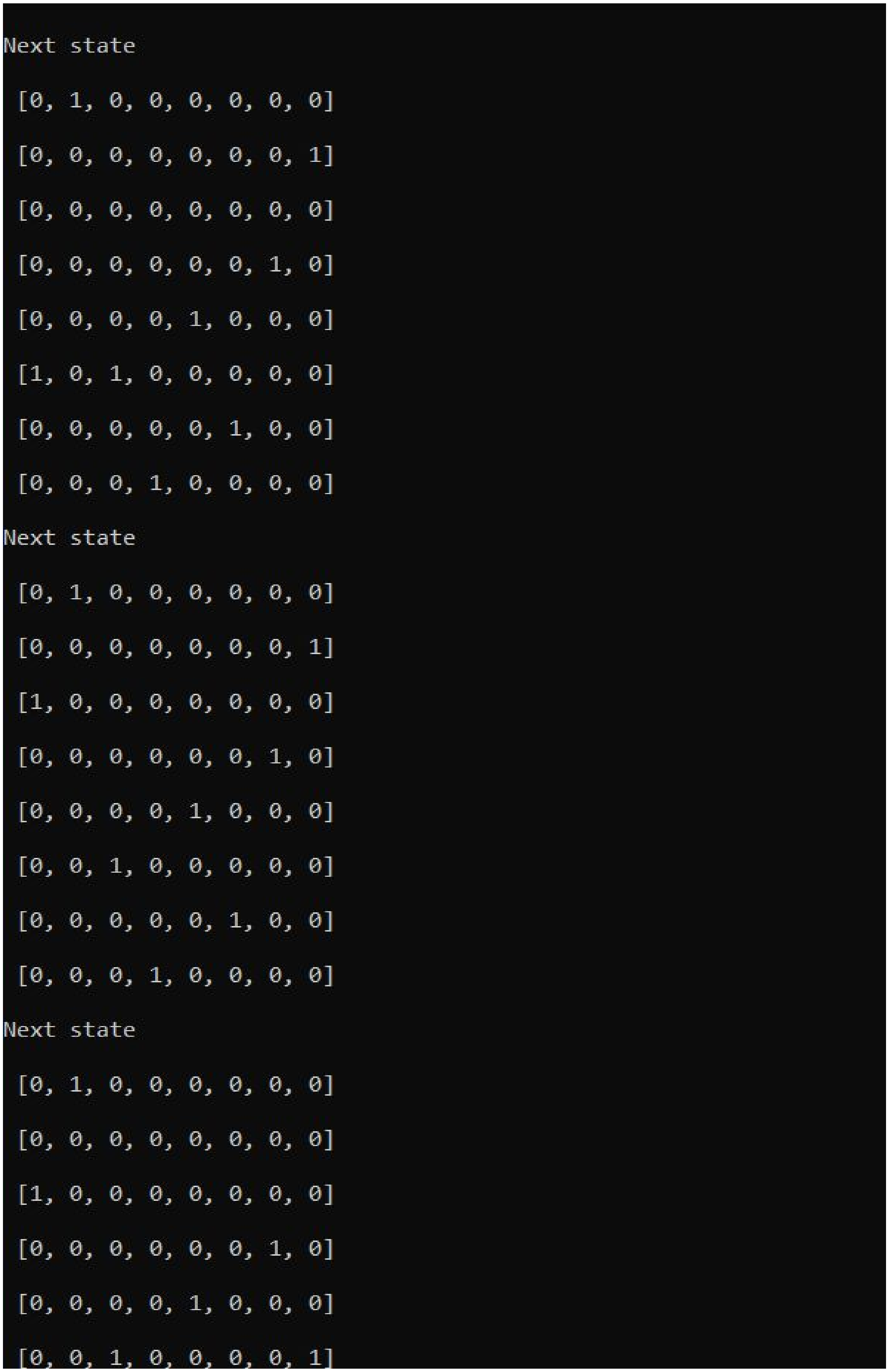
```
Next state
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
Next state
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
Next state
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 1, 0, 0, 0, 0, 0, 0]
```



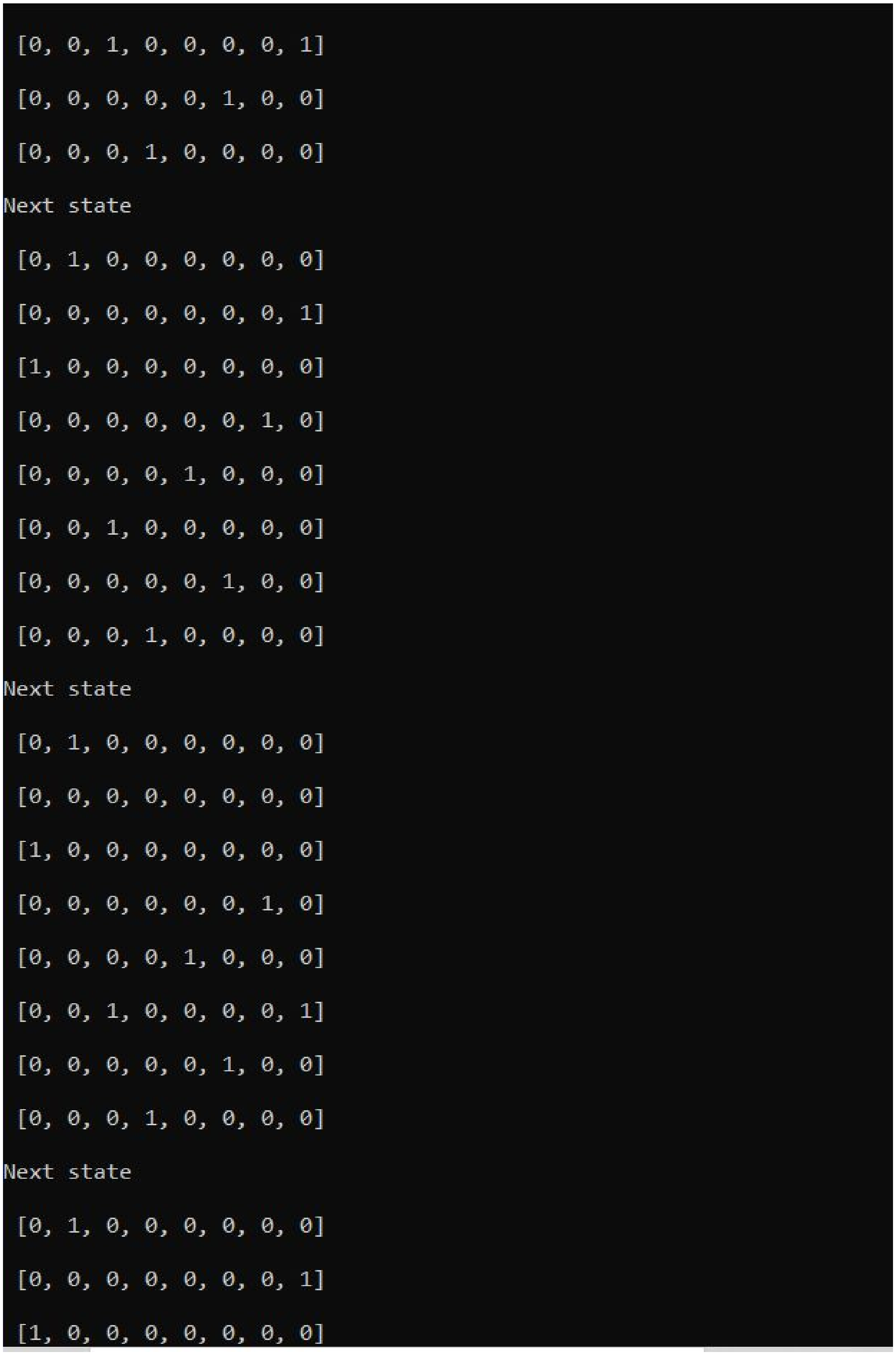
```
Next state

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

This was a successful instance.
-----------------------------------------------------------

Random instance : 2

Initial state

 [0, 1, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 1, 0, 1, 1]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [0, 1, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]
```



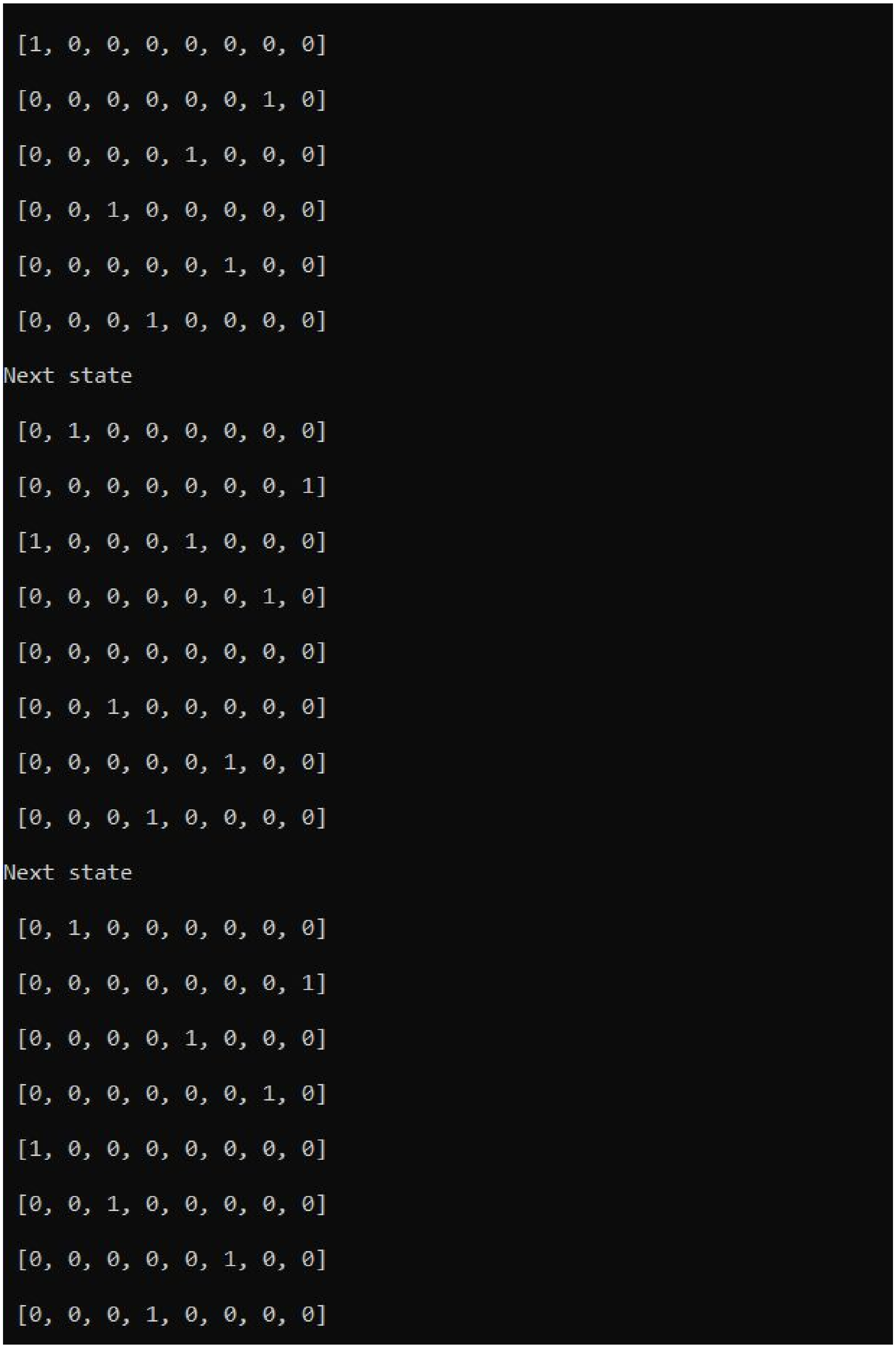
```
Next state
 [0, 1, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
```

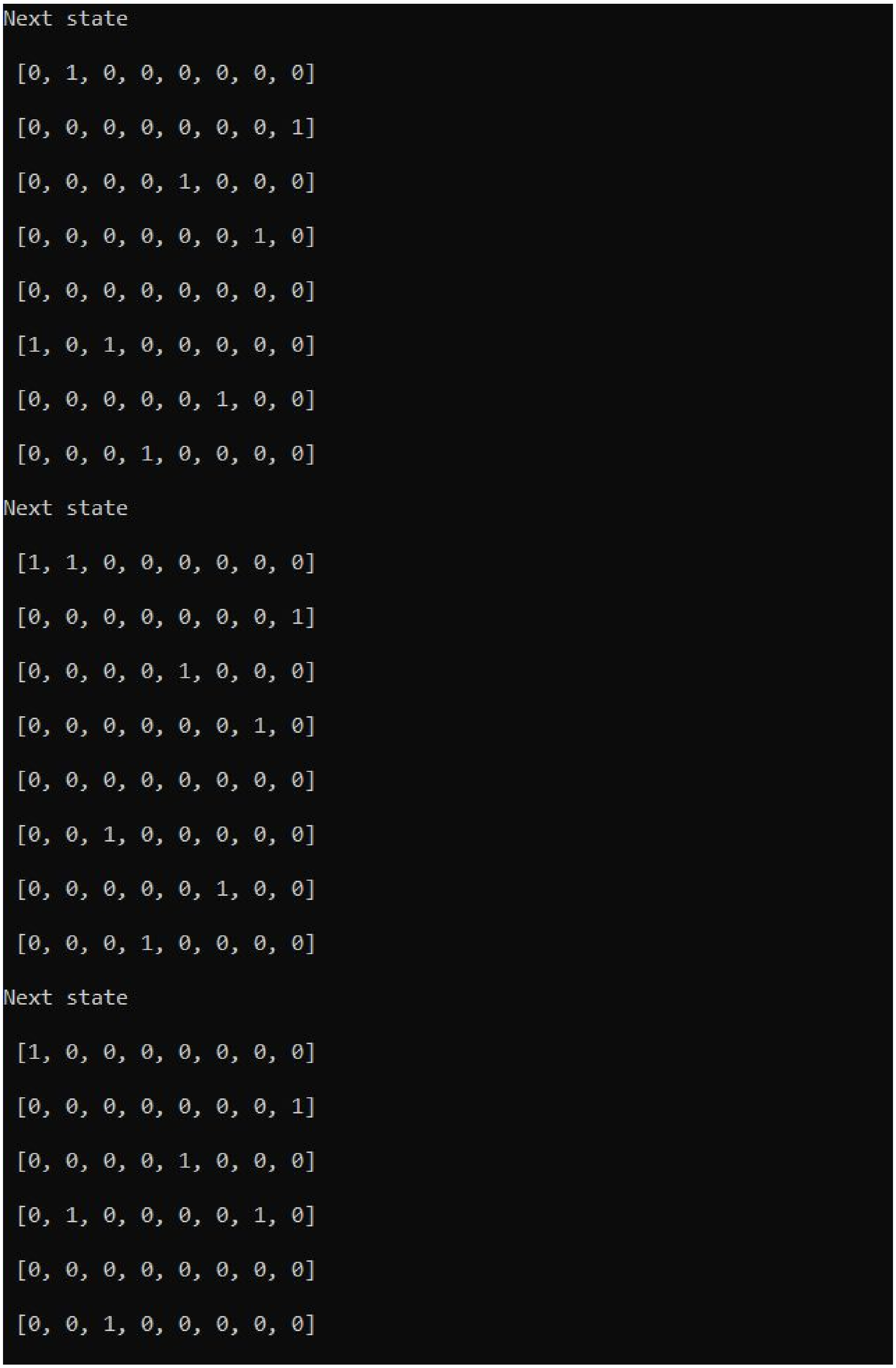


```
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [1, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 1]
```

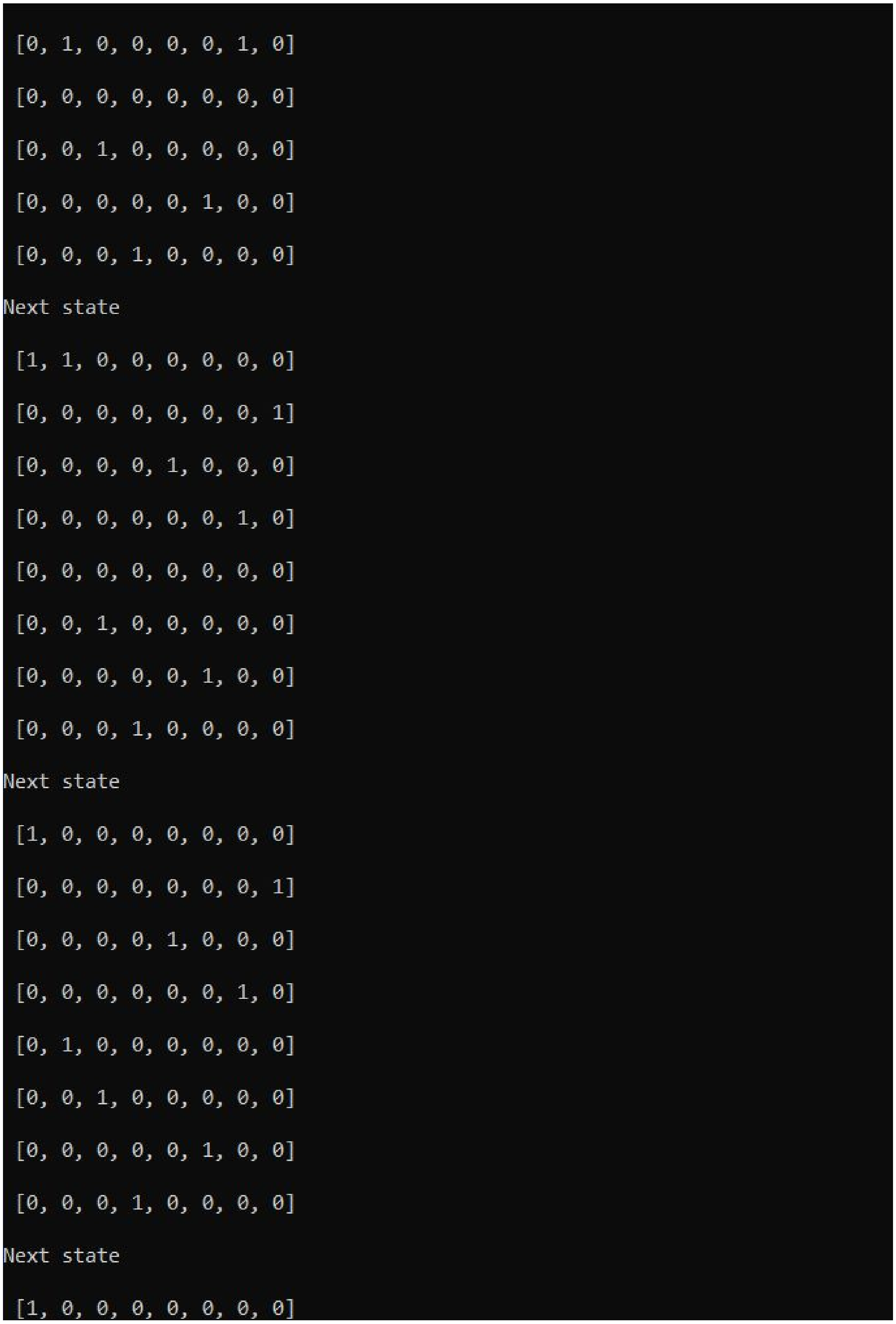


```
 [0, 0, 1, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [1, 0, 0, 0, 0, 0, 0, 0]
```

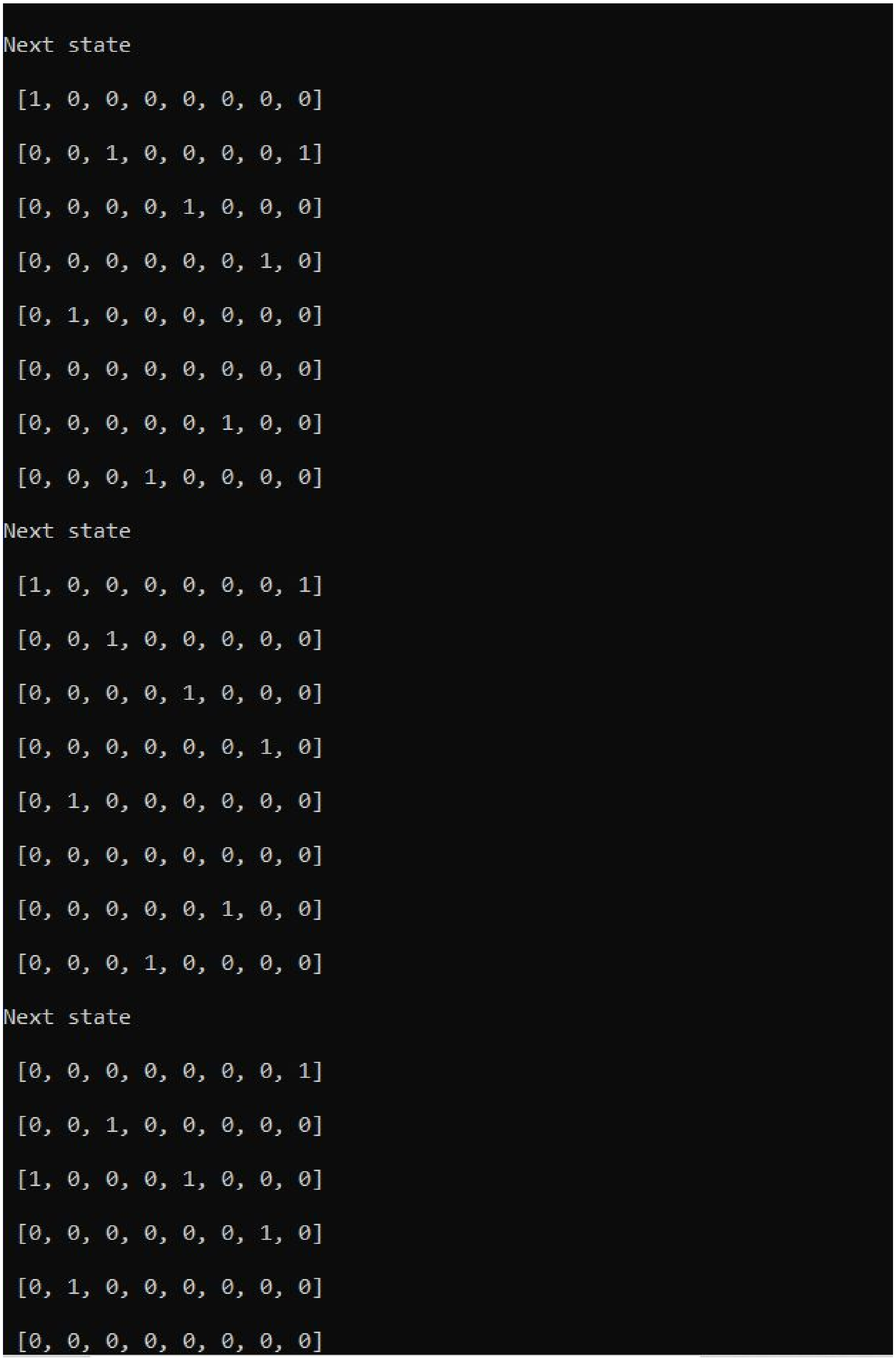

```
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [1, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
```



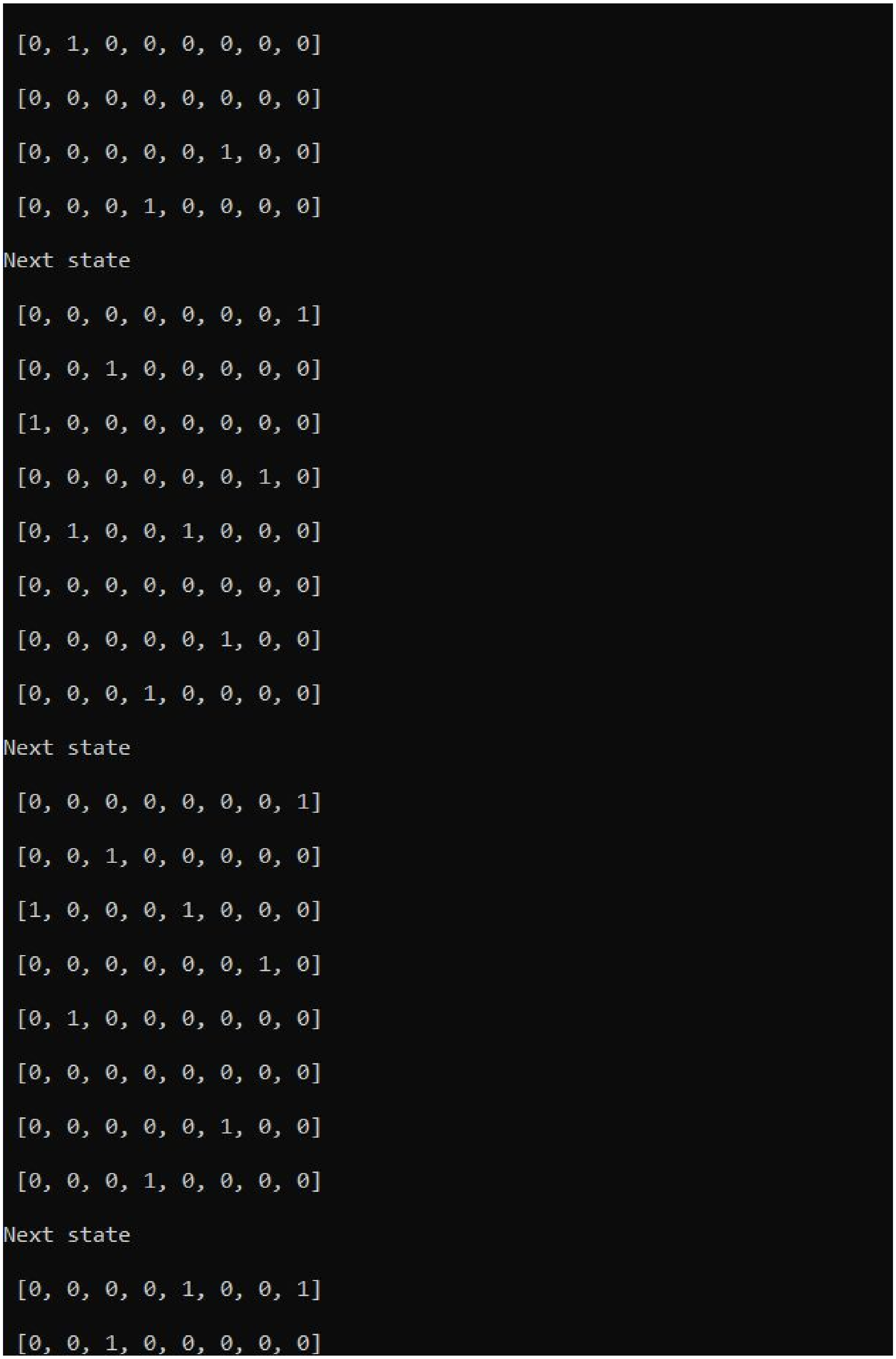
```
Next state

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [1, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [1, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]
```



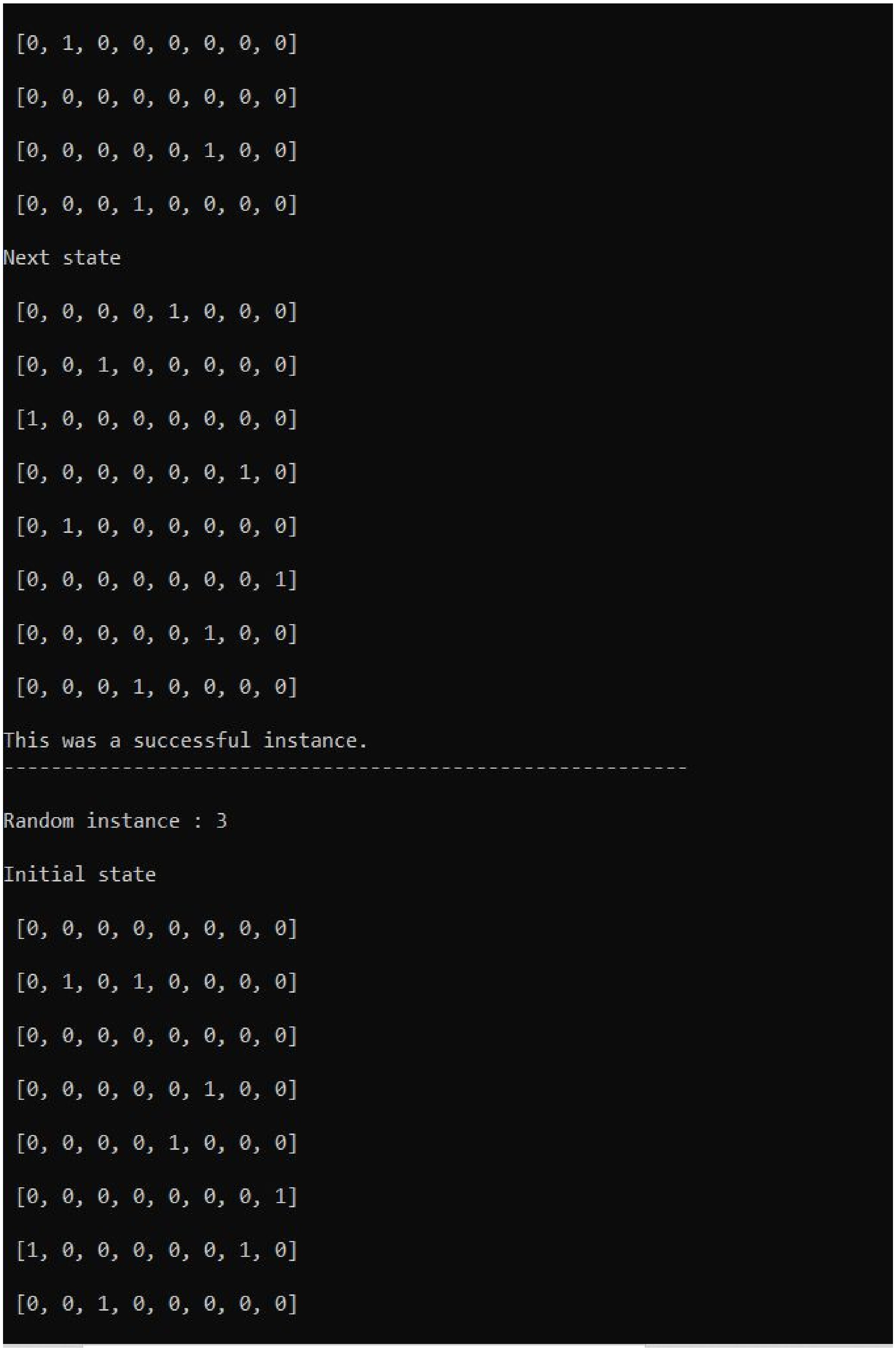
```
 [0, 1, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [1, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [1, 0, 0, 0, 0, 0, 0, 0]
```



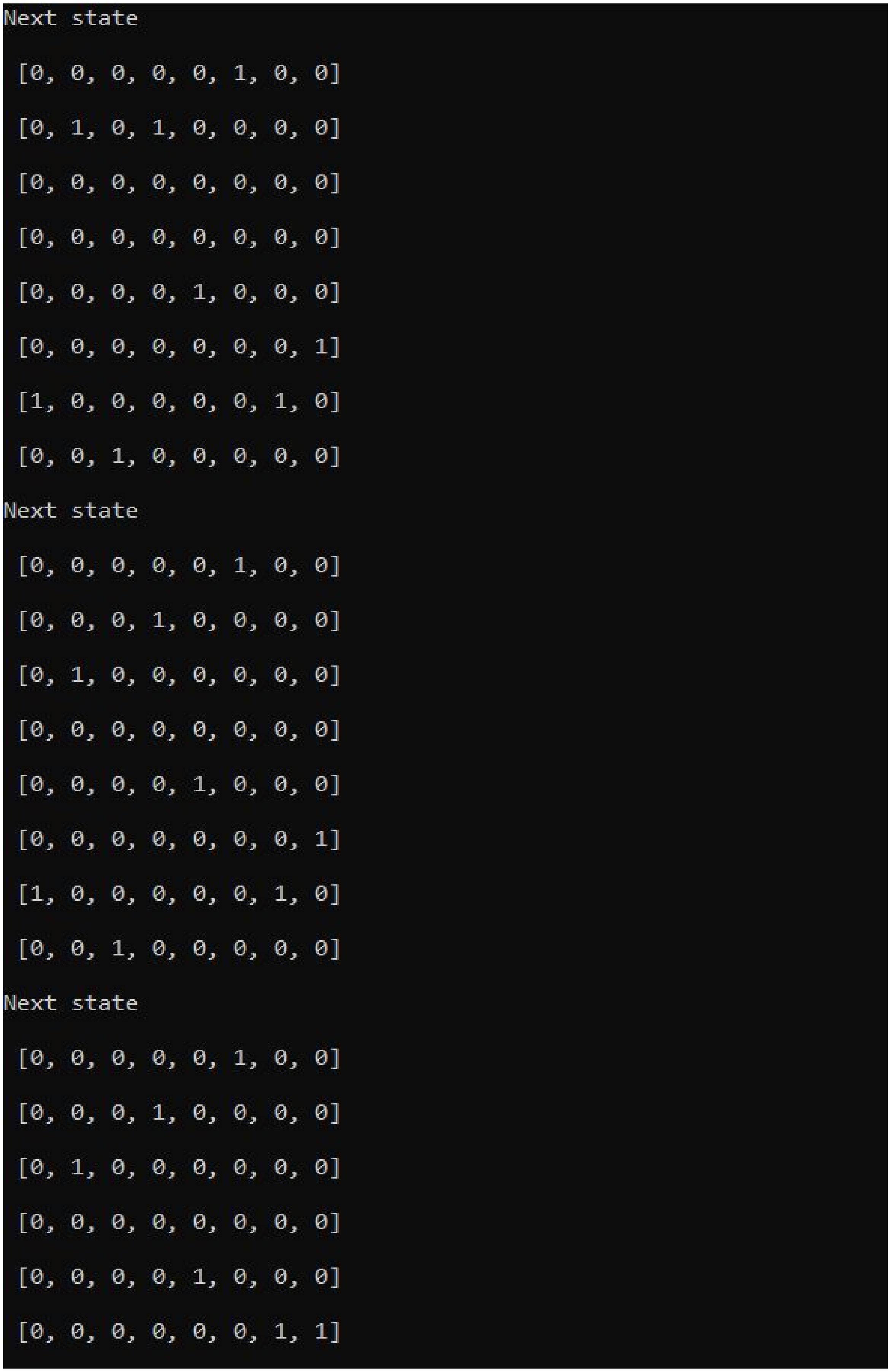
```
Next state

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]
Next state

 [1, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]
Next state

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]
```



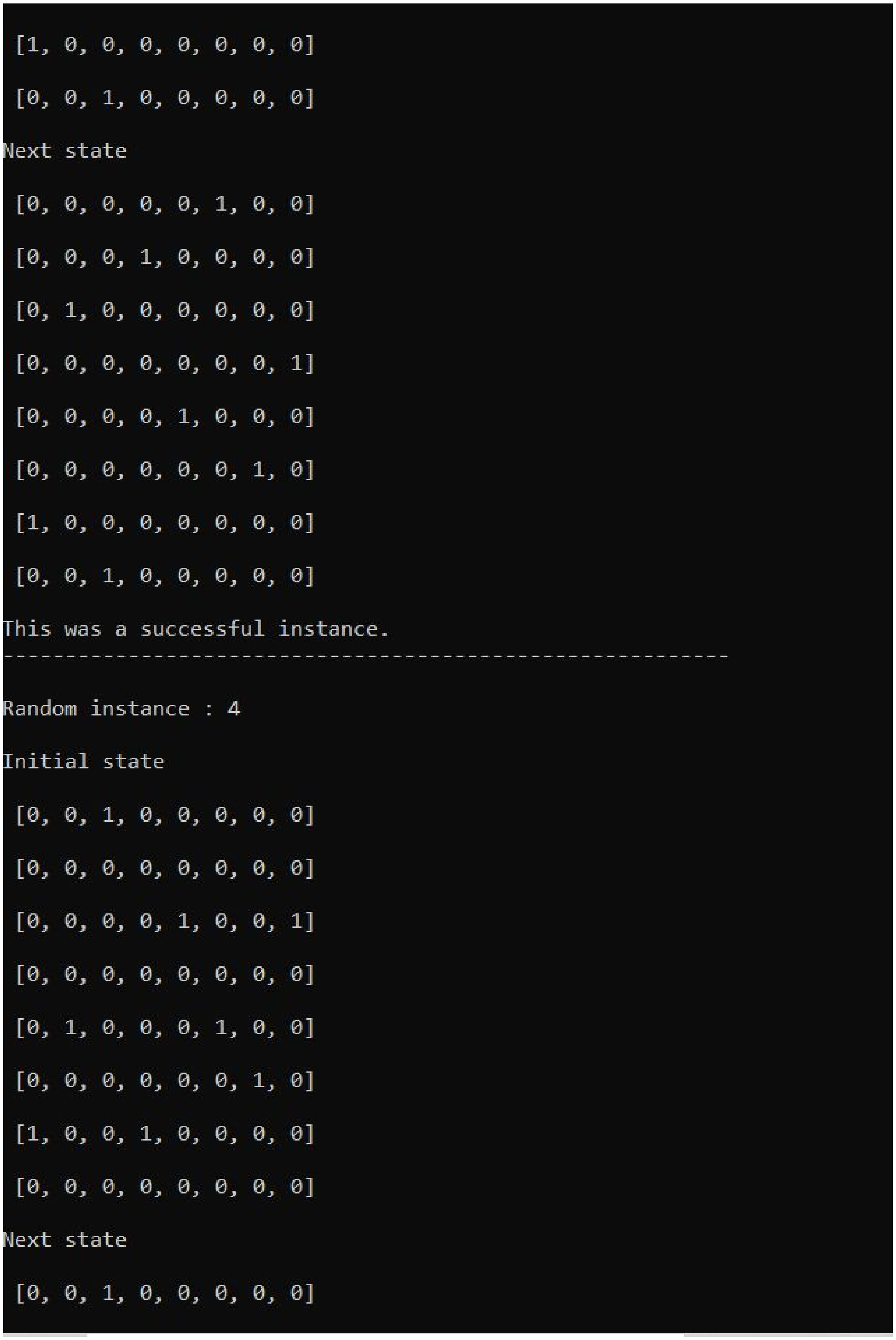
```
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 1, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 0, 0, 1, 0, 0, 1]
 [0, 0, 1, 0, 0, 0, 0, 0]
```



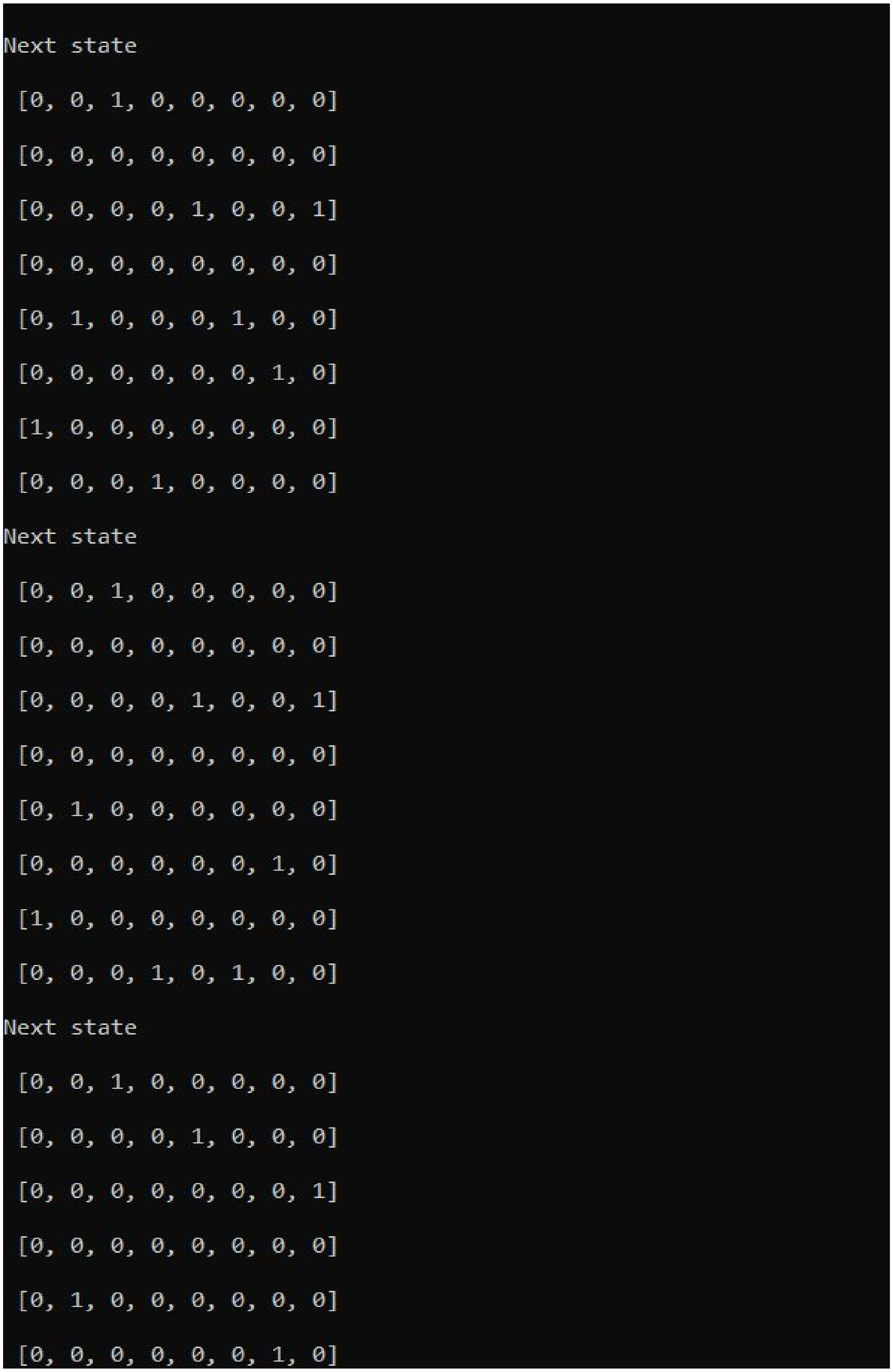
```
 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

This was a successful instance.
----------------------------------------------------------

Random instance : 3

Initial state

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 1, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [1, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]
```



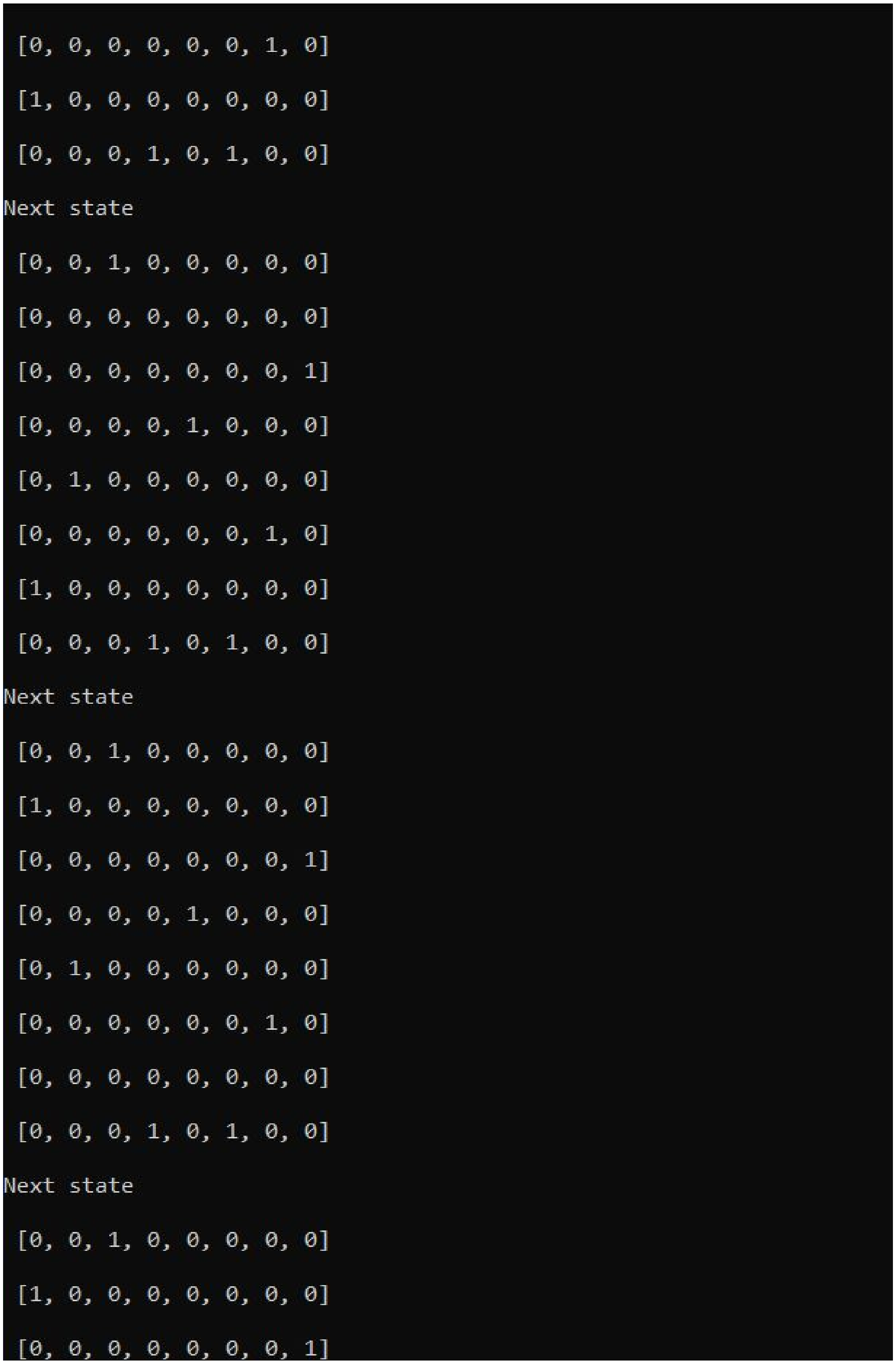
```
Next state

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 1, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [1, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]
Next state

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [1, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]
Next state

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 1]
```



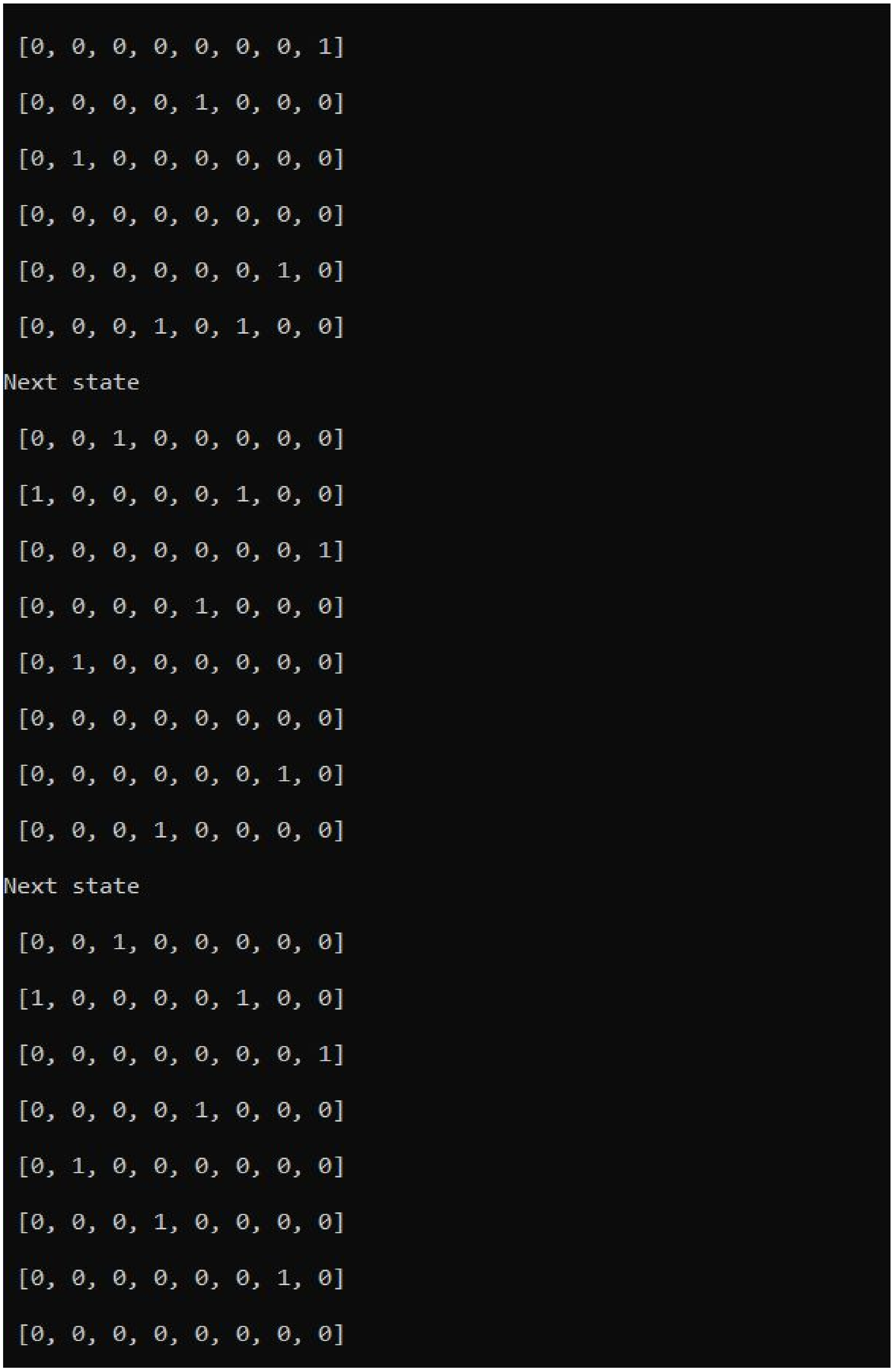
```
 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

Next state

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 1, 0, 0, 0, 0, 0]

This was a successful instance.
-----------------------------------------------------------

Random instance : 4

Initial state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 1]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 1, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [1, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

Next state

 [0, 0, 1, 0, 0, 0, 0, 0]
```



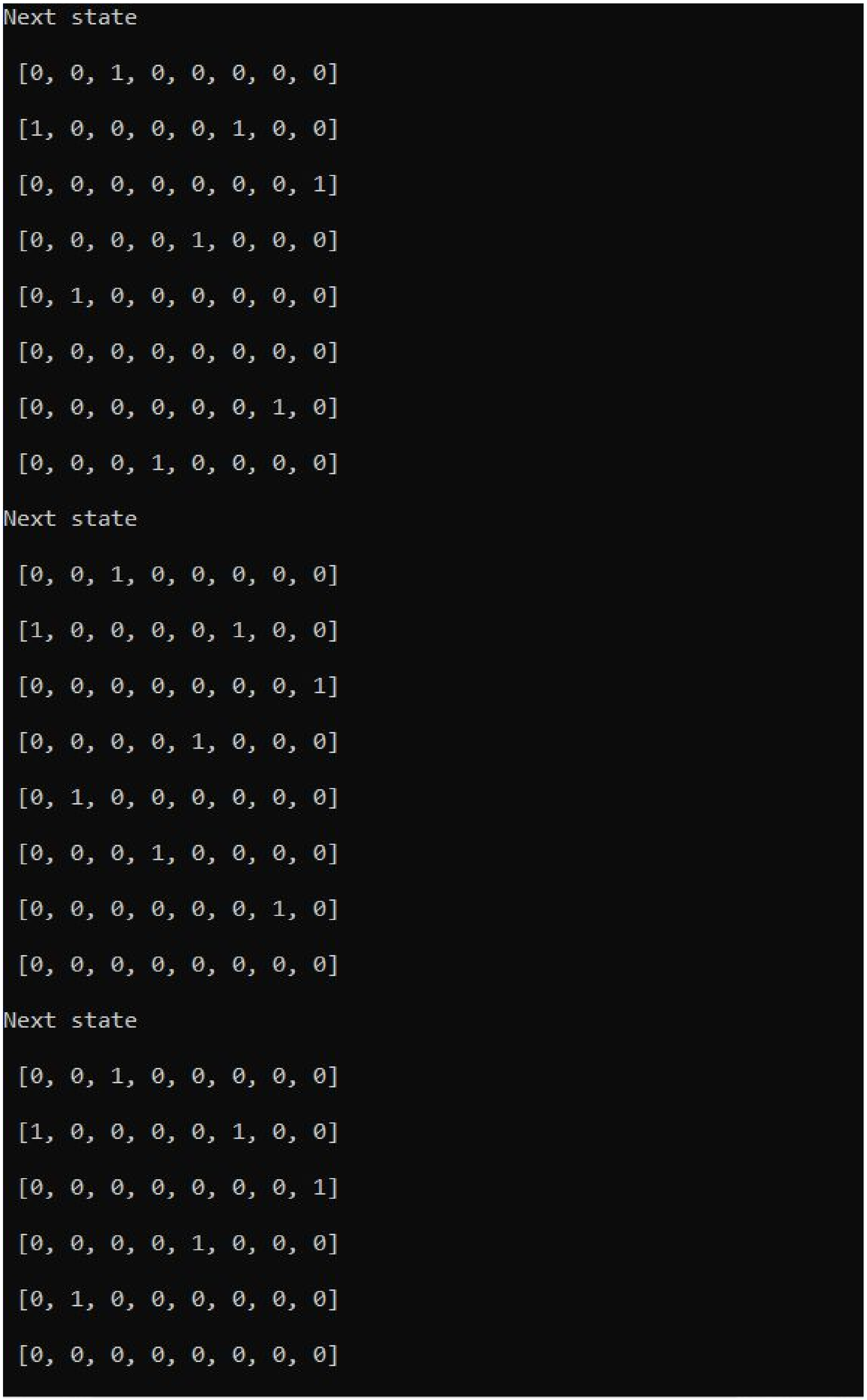
```
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 1, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
```



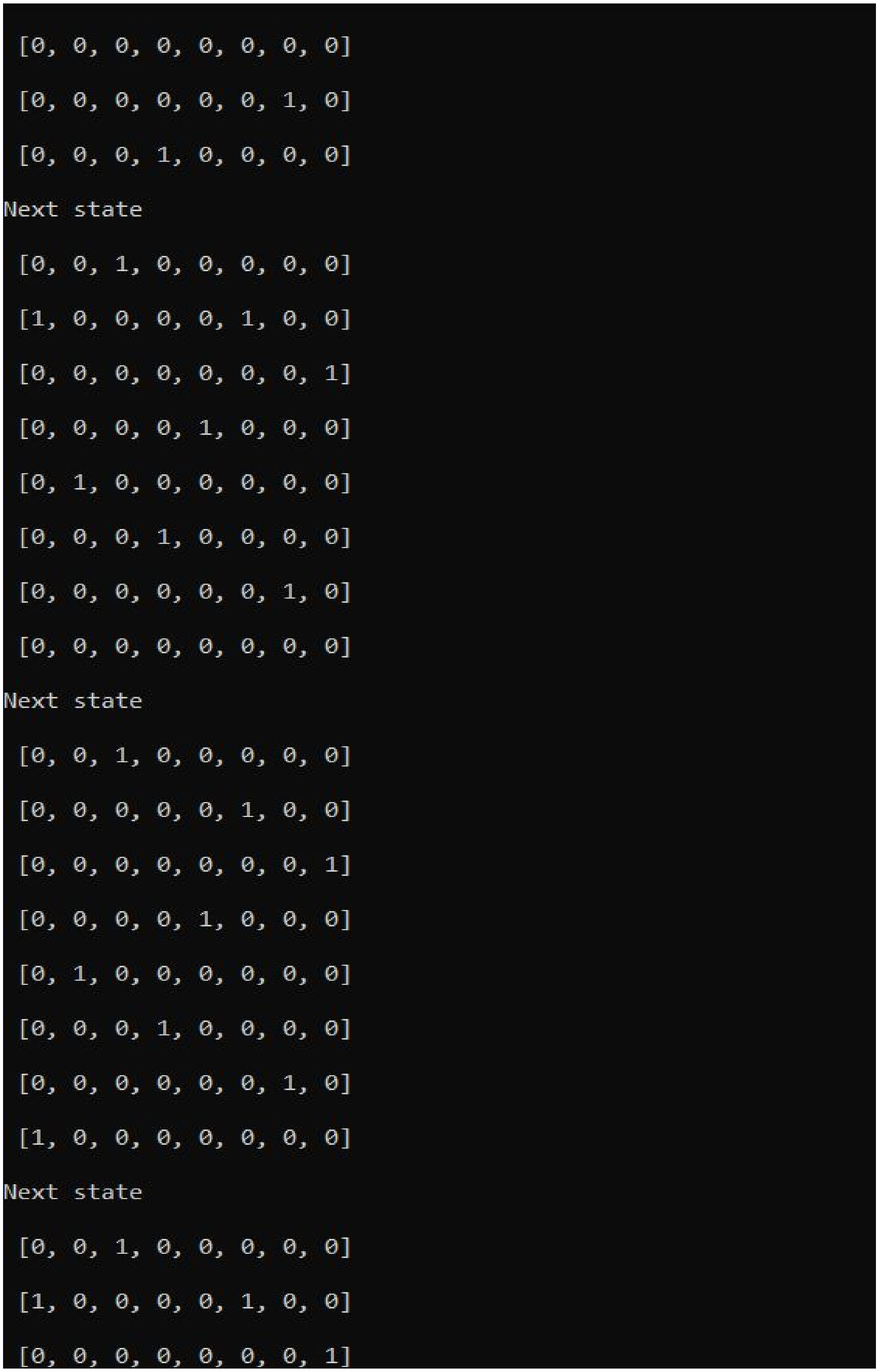
```
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 1, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 1, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 1, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
```

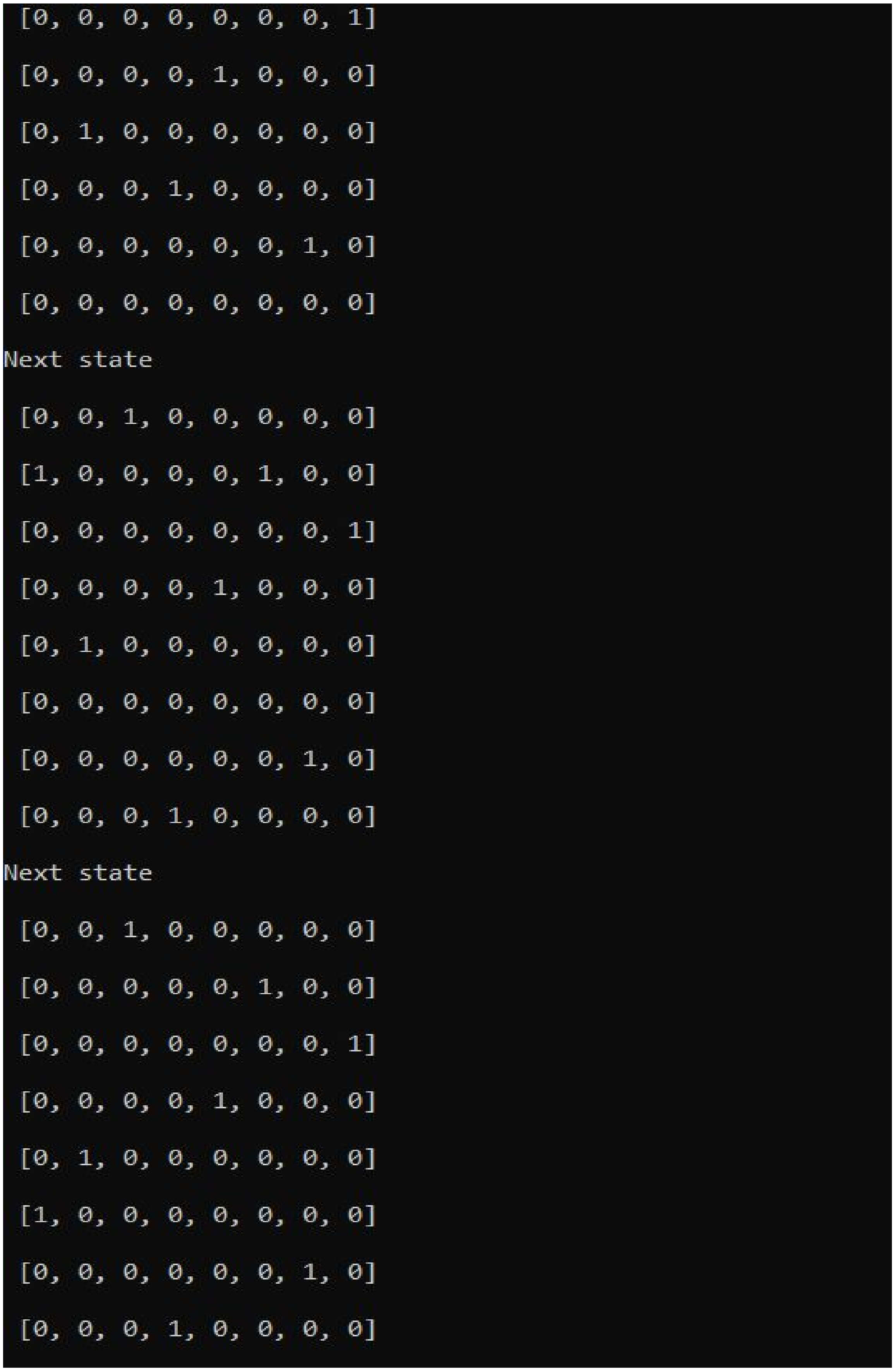


```
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 1, 0, 1, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
```

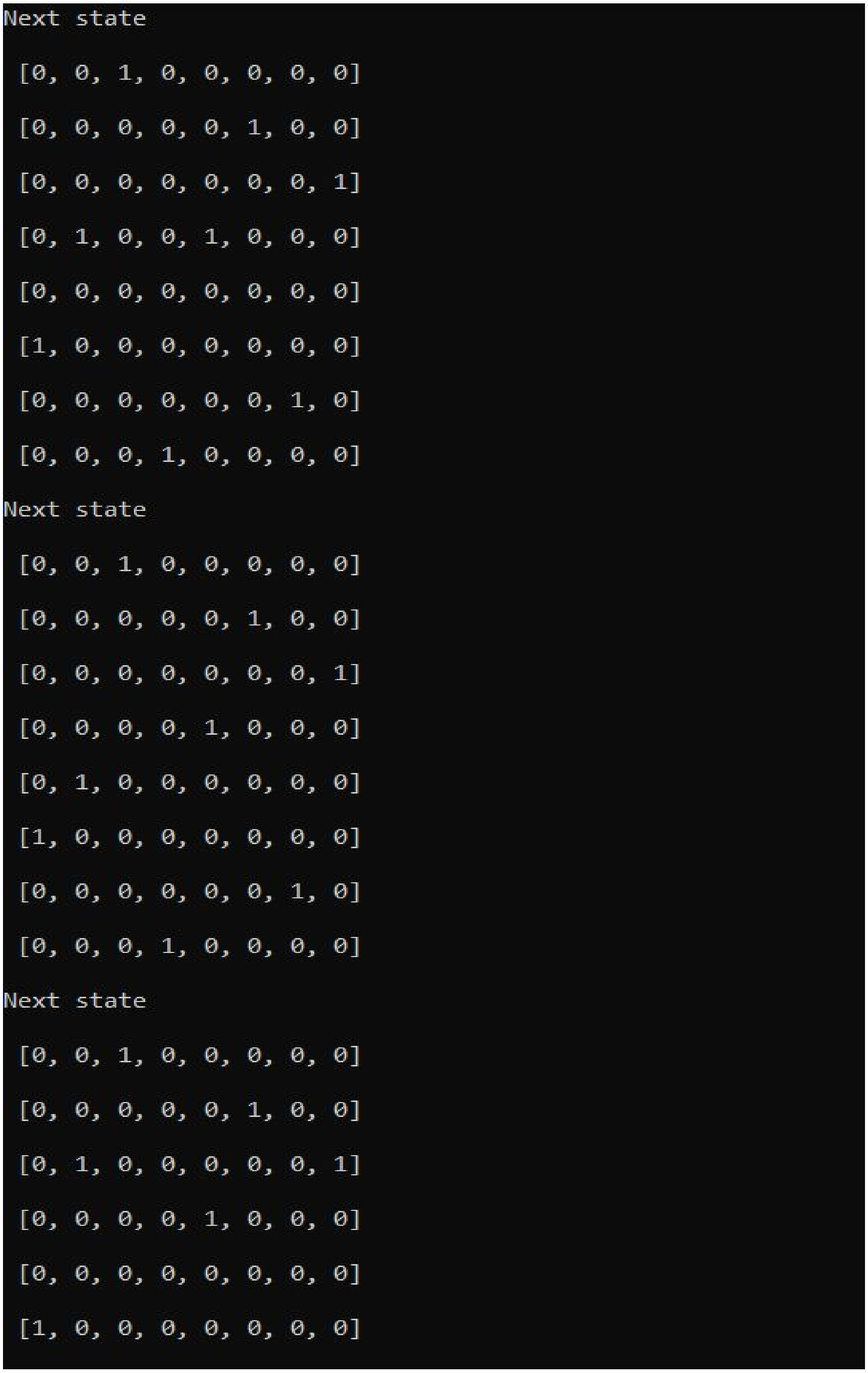


```
Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]
Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]
Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]
```

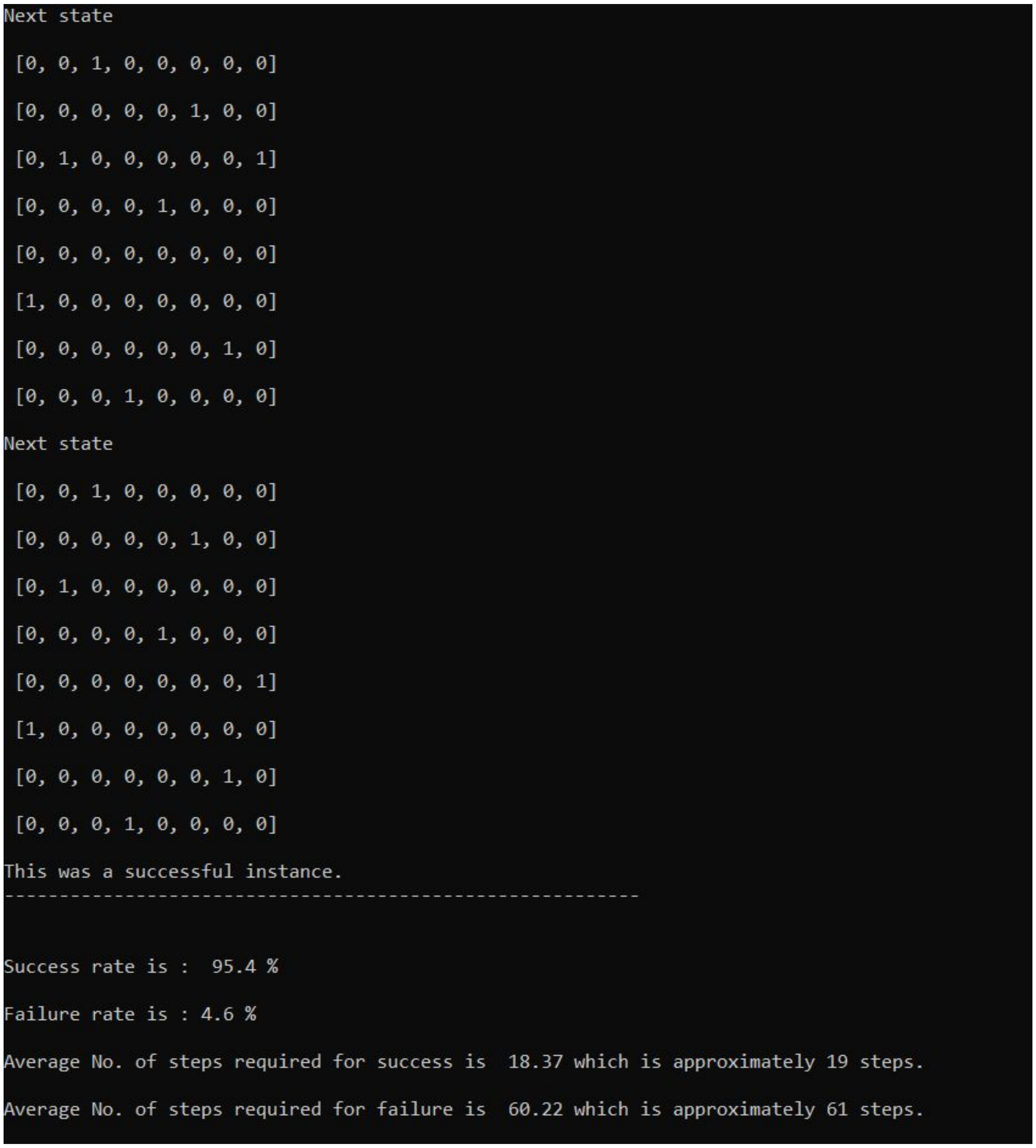

```
 [0, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
```



```
 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]
```

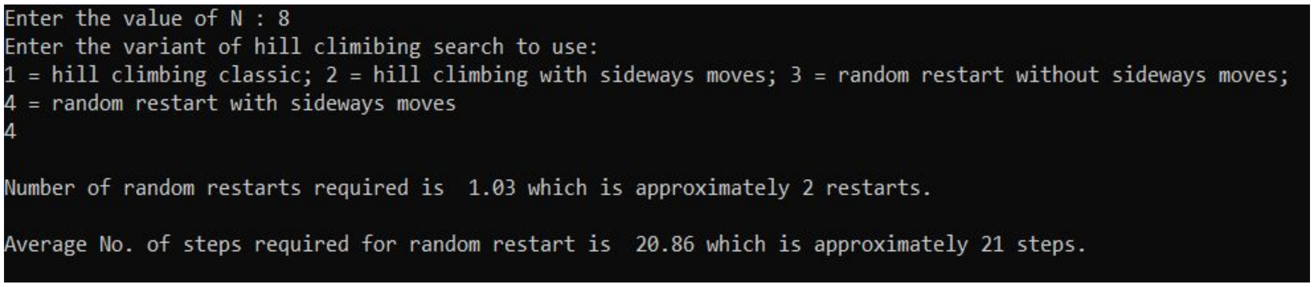


```
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 1, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 1, 0]
 [0, 0, 0, 1, 0, 0, 0, 0]
Next state
 [0, 0, 1, 0, 0, 0, 0, 0]
 [0, 0, 0, 0, 0, 1, 0, 0]
 [0, 1, 0, 0, 0, 0, 0, 1]
 [0, 0, 0, 0, 1, 0, 0, 0]
 [0, 0, 0, 0, 0, 0, 0, 0]
 [1, 0, 0, 0, 0, 0, 0, 0]
```



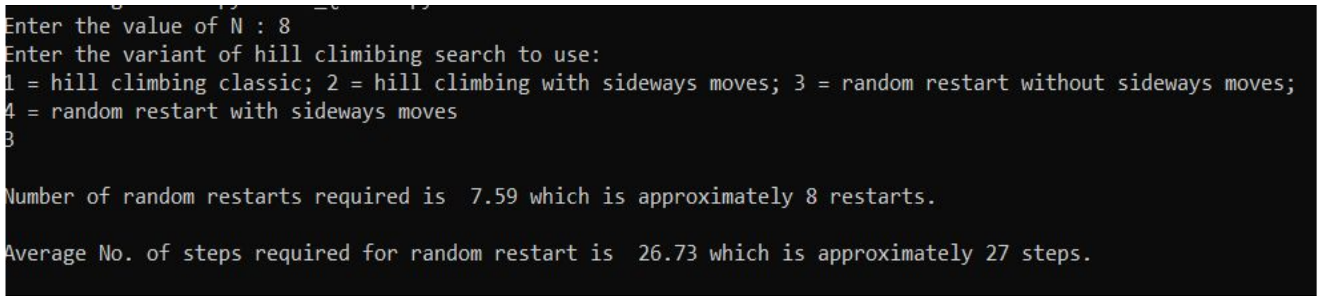
```
Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 1]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 0]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

Next state

 [0, 0, 1, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 1, 0, 0]

 [0, 1, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 1, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 0, 1]

 [1, 0, 0, 0, 0, 0, 0, 0]

 [0, 0, 0, 0, 0, 0, 1, 0]

 [0, 0, 0, 1, 0, 0, 0, 0]

This was a successful instance.
----------------------------------------------------------


Success rate is :  95.4 %

Failure rate is : 4.6 %

Average No. of steps required for success is  18.37 which is approximately 19 steps.

Average No. of steps required for failure is  60.22 which is approximately 61 steps.
```

3. Random restart with sideways moves



```
Enter the value of N : 8
Enter the variant of hill climibing search to use:
1 = hill climbing classic; 2 = hill climbing with sideways moves; 3 = random restart without sideways moves;
4 = random restart with sideways moves
4

Number of random restarts required is  1.03 which is approximately 2 restarts.

Average No. of steps required for random restart is  20.86 which is approximately 21 steps.
```

4. Random restart without sideways moves



```
Enter the value of N : 8
Enter the variant of hill climibing search to use:
1 = hill climbing classic; 2 = hill climbing with sideways moves; 3 = random restart without sideways moves;
4 = random restart with sideways moves
3

Number of random restarts required is  7.59 which is approximately 8 restarts.

Average No. of steps required for random restart is  26.73 which is approximately 27 steps.
```