T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

CSE4088 ARTIFICIAL INTELLIGENCE

ASSIGNMENT #1 Report

Student name surname : Hilal EKİNCİ

Student Number : 150114057

- 26.10.2018 –

Peg Solitaire

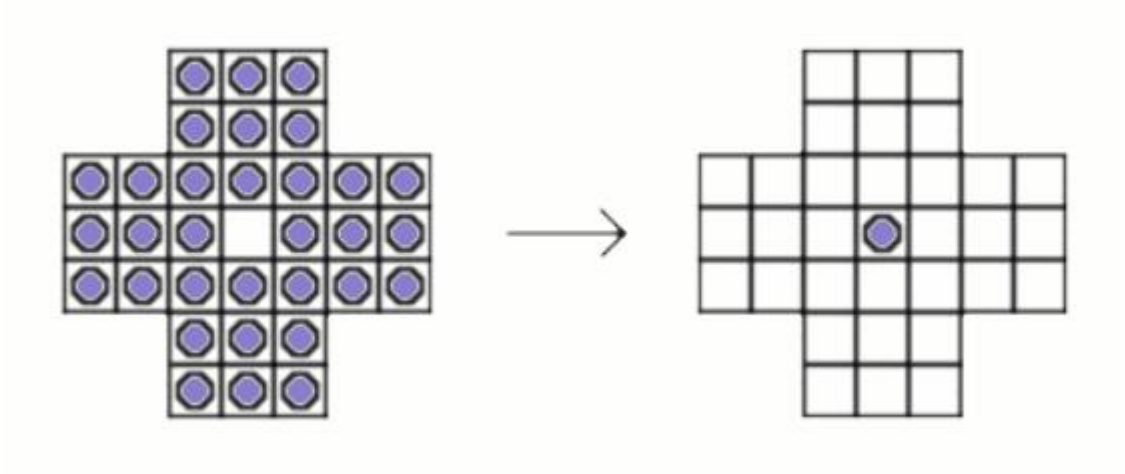

Figure 1:Peg Solitaire Game, initial state shown on the left and goal state on the right

This project was written in Python 3. I used class named as Node to hold board positions as lists, depth numbers and parents as Nodes. Firstly, I needed a skeleton for Peg Solitaire game by implemeting methods that find the movements, do movements on boards and check whether there are any movements on board etc. I calculate the total number of the movement according to restrictions in assignment PDF. After implementing a skeleton for this game, I started to implement tree search algorithms which are described below.

a) Breadth First Search

In **breadth-first search** the frontier is implemented as a **FIFO** (first-in, first-out) **queue**. Thus, the path that is selected from the frontier is the one that was added earliest [1].

In Breadth First Search (BFS), I need a queue structure to create frontier list. I used a module which is named as queue, it has a method named as put() to insert a node to the frontier list and it has a method named as get() to pull a node from the frontier list.

BFS is worst search algorithm for finding solution of peg solitaire game. The solutions of peg solitaire game will be in the leaves of the tree but it takes much time to reach the leaves so BFS could not find the solution in 1 hour. Here is my BFS program's output of peg solitaire according to restrictions given in the assignment PDF. The number of pegs left is 25. The steps of peg solitaire game are printed in the reverse order (O represents there is a peg in that hole, X is for empty hole).

```
(venv) C:\Users\hilal\PycharmProjects\AI_assignment_1>python bfs.py

BFS started!
Timeout!!!
# of nodes generated:  999209
# of nodes expanded :  92379
Sub-optimum Solution Found with 25 remaining pegs
DFS took 1 hour

--------------
    o x o
    x x x
 x x o x x o o
 o o o o o o o
 o o o o o o o
    o o o
    o o o
--------------
--------------
    o x o
    x x x
 o o x x x o o
 o o o o o o o
 o o o o o o o
    o o o
    o o o
--------------
--------------
    o x o
    x x x
 o x o o x o o
 o o o o o o o
 o o o o o o o
    o o o
    o o o
--------------
--------------
    o x x
    x x o
 o x o o o o o
 o o o o o o o
 o o o o o o o
    o o o
    o o o
--------------
--------------
    x o o
    x x o
 o x o o o o o
 o o o o o o o
 o o o o o o o
    o o o
    o o o
--------------
--------------
    o o o
    o x o
 o x x o o o o
 o o o o o o o
 o o o o o o o
    o o o
    o o o
--------------
```

```
--------------
    O O O
    O X O
 O O O X O O O
 O O O O O O O
 O O O O O O O
    O O O
    O O O
--------------
--------------
    O O O
    O O O
 O O O O O O O
 O O O X O O O
 O O O O O O O
    O O O
    O O O
--------------
```

b) Depth First Search

In **depth-first search**, the frontier acts like a **LIFO** (last-in, first-out) **stack** of paths. In a stack, elements are added and removed from the top of the stack. Using a stack means that the path selected and removed from the frontier at any time is the last path that was added [2].

In Depth First Search, we need a stack structure to create frontier list. I used Python list for stack, it has a method named as append() to insert a node to the frontier list and it has a method named as pop() to pull a node from the frontier list. DFS is most suitable for finding solution of peg solitaire game because the goal state will be in leaves of the tree. However, although DFS can find the solution of peg solitaire, DFS could not find the solution according to restrictions in 1 hour. Here is my DFS program's output of peg solitaire according to restrictions given in the assignment PDF. The number of pegs left is 2. The steps of peg solitaire game are printed in the reverse order (O represents there is a peg in that hole, X is for empty hole).

```
(venv) C:\Users\hilal\PycharmProjects\AI_assignment_1>python dfs.py

DFS started!
Timeout!!!
# of nodes generated:  1119959
# of nodes expanded :  1119795
Sub-optimum Solution Found with 2 remaining pegs
DFS took 1 hour
--------------
    X X X
    X X X
 X X X X X X X
 X X X X X X X
 X X X X X X X
    X X X
    O X O
--------------
```

X X X
X X X
X X X X X X X
X X X X X X X
X X X X O X X
X X O
O X X

X X X
X X X
X X X X X X X
X X X X X X X
X X X X X O O
X X O
O X X

X X X
X X X
X X X X X X X
X X X X O X X
X X X X O O O
X X X
O X X

X X X
X X X
X X X X X X X
X X X X O X X
X X X X X O O
X X O
O X O

X X X
X X X
X X X X X X X
X X X X O X X
X X X O O X O
X X O
O X O

X X X
X X X
X X X X X X X
X X X X O X X
X X X X O X O
X O O
O O O

X X X
X X X
X X X X X X X
X X X X O X X
X X O O X X O
X O O
O O O

```
---------------
     X X X
     X X X
 X X X X X X X
 X X X X O X X
 X X O X O O O
     X O O
     O O O
---------------
---------------
     X X X
     X X X
 X X O X X X X
 X X O X O X X
 X X X X O O O
     X O O
     O O O
---------------
---------------
     X X X
     X X X
 X X O X X X X
 X X X X O X X
 X X O X O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 X X O X X X X
 X X X X X O O
 X X O X O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 O O X X X X X
 X X X X X O O
 X X O X O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 O X O O X X X
 X X X X X O O
 X X O X O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 O X O X X X X
 X X X O X O O
 X X O O O O O
     O O O
     O O O
---------------
```

```
---------------
     X X X
     X X X
 O X X O O X X
 X X X O X O O
 X X O O O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 O O O X O X X
 X X X O X O O
 X X O O O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 O X O X O X X
 X O X O X O O
 X O O O O O O
     O O O
     O O O
---------------
---------------
     X X X
     X X X
 X X O X O X X
 O O X O X O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     X X O
     X X O
 X X O X X X X
 O O X O X O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     X X O
     X X X
 X X O X O X X
 O O X O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     X X O
     X X X
 X X O X X O O
 O O X O O O O
 O O O O O O O
     O O O
     O O O
---------------
```

```
---------------
     O X O
     O X X
 X X X X X O O
 O O X O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X O
     X X X
 X X O X X O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X O
     X X X
 O O X X X O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X O
     X X X
 O X O O X O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X X
     X X O
 O X O O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     X O O
     X X O
 O X O O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X O
 O X X O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
```

```
---------------
     O O O
     O X O
 O O O X O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O X O O O
 O O O O O O O
     O O O
     O O O
---------------
```

Here is an example only to show my DFS implementation correctly works with no restriction. It can find an optimal solution before a minute by moving firstly the last pegs of the board. Here is my DFS program's output of peg solitaire according with no restrictions. The steps of peg solitaire game are printed in the reverse order (O represents there is a peg in that hole, X is for empty hole).

```
C:/Users/hilal/PycharmProjects/AI_assignment_1/myDFS.py

DFS started!
Optimal solution found!
# of nodes generated:  20380
# of nodes expanded :  20274
DFS took 38 seconds

---------------
     X X X
     X X X
 X X X X X X X
 X X X O X X X
 X X X X X X X
     X X X
     X X X
---------------
---------------
     X X X
     X O X
 X X X O X X X
 X X X X X X X
 X X X X X X X
     X X X
     X X X
---------------
```

```
---------------
     X X X
     X O X
 X O O X X X X
 X X X X X X X
 X X X X X X X
     X X X
     X X X
---------------
---------------
     X X X
     X O X
 X O X X X X X
 X X O X X X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     X X X
     X O X
 X X O O X X X
 X X O X X X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     X X X
     X O X
 X X O X O O X
 X X O X X X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     X X O
     X O O
 X X O X X O X
 X X O X X X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O X
     X O O
 X X O X X O X
 X X O X X X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O X
     X O X
 X X O X O O X
 X X O X O X X
 X X O X X X X
     X X X
     X X X
---------------
```

```
---------------
     O O O
     X O O
 X X O X X O X
 X X O X O X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O O
     X O O
 O O X X X O X
 X X O X O X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O O
     O O O
 O O O X X O X
 X X X X O X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O O
     O O O
 O O O O O X X
 X X X X O X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O O
     O O O
 O O O O X O O
 X X X X O X X
 X X O X X X X
     X X X
     X X X
---------------
---------------
     O O O
     O O O
 O O O O X O O
 X X X X X X X
 X X O X O X X
     X X O
     X X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 X X X X O X X
 X X O X X X X
     X X O
     X X X
---------------
```

```
---------------
     O O O
     O O O
 O O O O O O O
 X X X X O X X
 X X X X X X X
     O X O
     O X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 X X O X O X X
 X X O X X X X
     X X O
     O X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O X X O X X
 X X O X X X X
     X X O
     O X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O X X X
 X X O X X X X
     X X O
     O X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O X X
 X X O X O X X
     X X X
     O X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O X O O
 X X O X O X X
     X X X
     O X X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O X O O
 X X O X X X X
     X X O
     O X O
---------------
```

```
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 X X O X O X X
     X X X
     O X O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 O O X X O X X
     X X X
     O X O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 O O O X O X X
     O X X
     X X O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 O O O X X O O
     O X X
     X X O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 O O O O O X O
     O X X
     X X O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 O O O O O X O
     O X X
     O O X
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O O O O O
 O O O O X X O
     O X O
     O O O
---------------
```

c) Iterative Deepening Search

Iterative deepening repeatedly calls a **depth-bounded searcher**, a depth-first searcher that takes in an integer **depth bound** and never explores paths with more arcs than this depth bound. Iterative deepening first does a depth-first search to depth 1 by building paths of length 1 in a depth-first manner. If that does not find a solution, it can build paths to depth 2, then depth 3, and so on until a solution is found. When a search with depth-bound nn fails to find a solution, it can throw away all of the previous computation and start again with a depth-bound of n+1n+1. Eventually, it will find a solution if one exists, and, as it is enumerating paths in order of the number of arcs, a path with the fewest arcs will always be found first [3].

In Iterative Deepening Search (IDS), like Depth First Search I used stack, I hold also a depth attribute in class Node and I solved peg solitaire on IDS by using the depths of the nodes. IDS is not suitable for finding the solution of peg solitaire game because of its iteratively searching the same nodes again so it could not find the solution in 1 hour. Here is my DFS program's output of peg solitaire below according to restrictions given in the assignment PDF. When I traced by looking movements of the peg solitaire game in real up to level 3, I see that my program's generated number of nodes are correct with the real numbers. The number of pegs left is 24. The steps of peg solitaire game are printed in the reverse order (O represents there is a peg in that hole, X is for empty hole).

```
IDS started.

At depth 0: # of nodes generated is 1,  total # of nodes generated 1
At depth 1: # of nodes generated is 5,  total # of nodes generated 6
At depth 2: # of nodes generated is 17,  total # of nodes generated 23
At depth 3: # of nodes generated is 85,  total # of nodes generated 108
At depth 4: # of nodes generated is 617,  total # of nodes generated 725
At depth 5: # of nodes generated is 5095,  total # of nodes generated 5820
At depth 6: # of nodes generated is 48559,  total # of nodes generated 54379
At depth 7: # of nodes generated is 508381,  total # of nodes generated 562760

Timeout!!!
IDS with random selection took 1 hour
Sub-optimum Solution Found with 24 remaining pegs
---------------
     O X O
     O X X
 X X X X X O O
 O O X O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X O
     X X X
 X X O X X O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X O
     X X X
 O O X X X O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X O
     X X X
 O X O O X O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O X X
     X X O
 O X O O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
```

```
---------------
     X O O
     X X O
 O X O O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X O
 O X X O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X O
 O O O X O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O O O
     O O O
 O O O O O O O
 O O O X O O O
 O O O O O O O
     O O O
     O O O
---------------
```

d) Depth First Search with Random Selection

In Depth First Search with Random Selection, the program finds the movements in the board, puts the movements to the stack list randomly and applies the movements. DFS with random selection could not find the solution in 1 hour because of random way. Here is my DFS with random selection program's output of peg solitaire. For each time I run the DFS with random selection, I obtained different number of pegs remaining as expected. The minimum number of pegs left is 4. The steps of peg solitaire game are printed in the reverse order (O represents there is a peg in that hole, X is for empty hole).

```
C:/Users/hilal/PycharmProjects/AI_assignment_1/random.py

DFS with random selection started!
Timeout!!!

# of nodes generated:  1232650
# of nodes expanded :  1232536
DFS with random selection took 1 hour
Sub-optimum Solution Found with 4 remaining pegs
--------------
     O X X
     X X X
 O X X X X X O
 X X X X X X X
 X X X X X X X
     X X X
     X X O
--------------
--------------
     X O O
     X X X
 O X X X X X O
 X X X X X X X
 X X X X X X X
     X X X
     X X O
--------------
--------------
     X O O
     X X X
 X O O X X X O
 X X X X X X X
 X X X X X X X
     X X X
     X X O
--------------
--------------
     O O O
     O X X
 X O X X X X O
 X X X X X X X
 X X X X X X X
     X X X
     X X O
--------------
--------------
     O O O
     O X X
 X X O O X X O
 X X X X X X X
 X X X X X X X
     X X X
     X X O
--------------
--------------
     O O O
     O X X
 X X O X X X O
 X X X O X X X
 X X X O X X X
     X X X
     X X O
--------------
```

```
---------------
     O O O
     O X X
 O O X X X X O
 X X X O X X X
 X X X O X X X
     X X X
     X X O
---------------
---------------
     O O O
     O X X
 O X O O X X O
 X X X O X X X
 X X X O X X X
     X X X
     X X O
---------------
---------------
     O O O
     O X X
 O X O O X X O
 X X X O X X X
 X X X O X X X
     X X X
     O O X
---------------
---------------
     O O O
     O X X
 O X O O X X O
 X X X O X X X
 X X X X O O X
     X X X
     O O X
---------------
---------------
     O O O
     O X X
 O X O O X X O
 X X X O X X X
 X X X X X O X
     X X O
     O O O
---------------

---------------
     O O O
     O X X
 O X O O X X O
 X X X O X X X
 X X X O O X X
     X X O
     O O O
---------------
---------------
     O O O
     O X X
 O X O X O O O
 X X X O X X X
 X X X O O X X
     X X O
     O O O
---------------
```

```
---------------
     O O O
     O X X
 X X O X O O O
 O X X O X X X
 O X X O O X X
     X X O
     O O O
---------------
---------------
     O O O
     O X X
 X X O X O O O
 O X X O X X X
 O O O X O X X
     X X O
     O O O
---------------
---------------
     O O O
     O X X
 X X O X O O O
 O X X O X X X
 O O O X X O O
     X X O
     O O O
---------------
---------------
     O O O
     O X X
 O O X X O O O
 O X X O X X X
 O O O X X O O
     X X O
     O O O
---------------
---------------
     O O O
     O X X
 O O X X O O O
 O X X X X X X
 O O O O X O O
     X O O
     O O O
---------------

---------------
     O O O
     O X X
 O O O X O O O
 O X O X X X X
 O O X O X O O
     X O O
     O O O
---------------
---------------
     O O O
     O X X
 O O O X X O O
 O X O X O X X
 O O X O O O O
     X O O
     O O O
---------------
```

```
---------------
     O O O
     O X X
 O O O O X O O
 O X O O O X X
 O O X X O O O
     X O O
     O O O
---------------
---------------
     O O O
     O X X
 O O O O X O O
 O X X O O X X
 O O O X O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X X
 O O O O X O O
 O O O X O X X
 O O O X O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X X
 O O O O X O O
 O O O X X O O
 O O O X O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X X
 O O O O X O O
 O O O O O X O
 O O O X O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X O
 O O O O O O O
 O O O O X X O
 O O O X O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X O
 O O O O O O O
 O O O X O O O
 O O O X O O O
     O O O
     O O O
---------------
```

```
---------------
      O O O
      O X O
  O O O X O O O
  O O O O O O O
  O O O O O O O
      O O O
      O O O
---------------
---------------
      O O O
      O O O
  O O O O O O O
  O O O X O O O
  O O O O O O O
      O O O
      O O O
---------------
```

e) Depth First Search with a Node Selection Heuristic

My node selection heuristic is based on gathering the pegs in the middle area of the board. By doing this, I prevent remaining one peg in the corners of the board. I have found the optimal solution in 30 minutes by using DFS with my node selection heuristic. The pegs which will be moved firstly are shown in the figure 1 below.

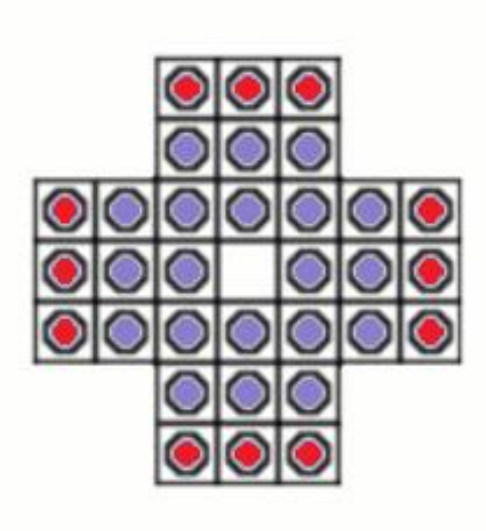

Figure 2: The pegs with most priority in DFS to move

The pegs which will be moved after the pegs shown in the figure 1 are shown in the figure 2 below.

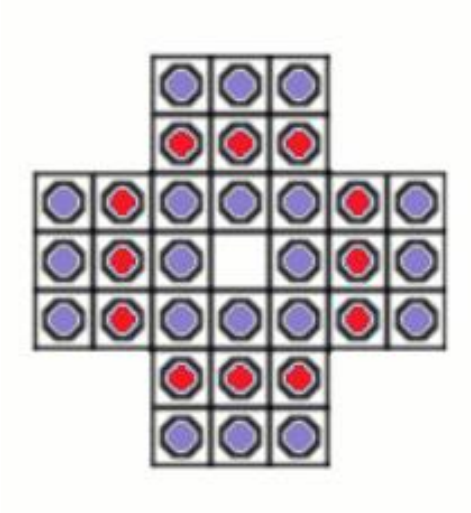

Figure 3: The pegs with the secondly most priority in DFS to move

I also check board if it has a movement available before I put it into frontier list. It decreases the number of the nodes will be expanded and it makes my program fast. Here is my DFS with node selection heuristic program's output of peg solitaire. The steps of peg solitaire game are printed in the reverse order (O represents there is a peg in that hole, X is for empty hole).

```
DFS with node selection heurist started.
Optimum solution found!
# of nodes generated:  809988
# of nodes expanded :  809829
DFS took 1842.50 seconds, 30.7 minutes

---------------
     X X X
     X X X
 X X X X X X X
 X X X O X X X
 X X X X X X X
     X X X
     X X X
---------------
---------------
     X X X
     X X X
 X X X X X X X
 X X X X X X X
 X X X O X X X
     X O X
     X X X
---------------
```

```
---------------
     X X X
     X X X
 X X X X X X X
 X X X X X X X
 X O O X X X X
     X O X
     X X X
---------------
---------------
     X X X
     X X X
 X X X X X X X
 X X X O X X X
 X O O O X X X
     X X X
     X X X
---------------
---------------
     X X X
     X X X
 X X X X X X X
 X X X X O O X
 X O O O X X X
     X X X
     X X X
---------------
---------------
     X X X
     X X O
 X X X X O X X
 X X X X X O X
 X O O O X X X
     X X X
     X X X
---------------
---------------
     X X X
     X X O
 X X X X X O O
 X X X X X O X
 X O O O X X X
     X X X
     X X X
---------------
---------------
     X X X
     X X O
 X X X O O X O
 X X X X X O X
 X O O O X X X
     X X X
     X X X
---------------
---------------
     X X X
     X X O
 X O O X O X O
 X X X X X O X
 X O O O X X X
     X X X
     X X X
---------------
```

```
---------------
     X X X
     X X O
 X O O X O X X
 X X X X X O O
 X O O O X X O
     X X X
     X X X
---------------
---------------
     X X X
     X X O
 X O O X X O O
 X X X X X O O
 X O O O X X O
     X X X
     X X X
---------------
---------------
     X X X
     X X O
 X O O O O X O
 X X X X X O O
 X O O O X X O
     X X X
     X X X
---------------
---------------
     O X X
     O X O
 X O X O O X O
 X X X X X O O
 X O O O X X O
     X X X
     X X X
---------------

---------------
     O X X
     X X O
 X O O O O X O
 X X O X X O O
 X O O O X X O
     X X X
     X X X
---------------
---------------
     O X X
     X X O
 X O O O O X O
 X X O O O X O
 X O O O X X O
     X X X
     X X X
---------------
---------------
     O X X
     X X O
 X O O O O X O
 X X O O O X O
 X O O X O O O
     X X X
     X X X
---------------
```

```
---------------
     O X X
     X X O
 X O O O O X O
 X X O O O X O
 X O X X O O O
     O X X
     O X X
---------------
---------------
     O X X
     X X O
 X O O O O X O
 X X O O O X O
 X X O O O O O
     O X X
     O X X
---------------
---------------
     O X X
     X X O
 X O O O O O O
 X X O O O O O
 X X O O O X O
     O X X
     O X X
---------------
---------------
     O X X
     X X O
 X O O O O O O
 X X O O O O O
 X X O O X X O
     O X O
     O X O
---------------
---------------
     O X X
     X X O
 X O O O O O O
 X X O O O O O
 X X O X O O O
     O X O
     O X O
---------------
---------------
     O X X
     X X O
 X O O O O O O
 X X O O O O O
 O O X X O O O
     O X O
     O X O
---------------
---------------
     O X X
     X X O
 X O O O O O O
 X X O O O O O
 O X O O O O O
     O X O
     O X O
---------------
```

```
---------------
     O X X
     X X O
 X O O O O O O
 X X O O O O O
 O X O X O O O
     O O O
     O O O
---------------
---------------
     O X X
     X X O
 O O O O O O O
 O X O O O O O
 X X O X O O O
     O O O
     O O O
---------------
---------------
     O X X
     X X O
 O O O O O O O
 O X O O O O O
 O O X X O O O
     O O O
     O O O
---------------
---------------
     O X X
     X X O
 O O O O O O O
 O X O O O O O
 O X O O O O O
     O O O
     O O O
---------------
---------------
     O X X
     X X O
 O X O O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     X O O
     X X O
 O X O O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
---------------
     O O O
     O X O
 O X X O O O O
 O O O O O O O
 O O O O O O O
     O O O
     O O O
---------------
```

Refrences

[1] BFS, ARTIFICIAL INTELLIGENCE 2E FOUNDATIONS OF COMPUTATIONAL AGENTS, [ONLINE] AVAILABLE AT: **https://artint.info/2e/html/ArtInt2e.Ch3.S5.SS1.html** (Date of Access: 10.12.2018)

[2] DFS, ARTIFICIAL INTELLIGENCE 2E FOUNDATIONS OF COMPUTATIONAL AGENTS, [ONLINE] AVAILABLE AT: **https://artint.info/2e/html/ArtInt2e.Ch3.S5.SS2.html** (Date of Access: 10.12.2018)

[3] IDS, ARTIFICIAL INTELLIGENCE 2E FOUNDATIONS OF COMPUTATIONAL AGENTS, [ONLINE] AVAILABLE AT: **https://artint.info/2e/html/ArtInt2e.Ch3.S5.SS3.html** (Date of Access: 10.12.2018)