ARTIFICAL INTELLIGENCE

PROJECT 1

KNIGHT' S TOUR PROBLEM

AHMET AŞIK – 150113062

UFUK GÜRBÜZ – 150113058

PROCEDURES

Introduction to Knight's Tour Problem

A knight's tour is a sequence of moves of a knight on a chessboard such that the knight visits every square only once. If the knight ends on a square that is one knight's move from the beginning square, the tour is closed, otherwise it is open.

The knight's tour problem is the mathematical problem of finding a knight's tour. Variations of the knight's tour problem involve chessboards that has N × N sizes. The knight's tour problem is an instance of the more general Hamiltonian path problem in **graph theory**.

General Problem Formulation

The knight's tour problem has no associated path cost to minimize. Any sequence of NxN moves that reaches a goal state is a perfectly valid solution. In other words, if there is a final state, it must be at the last level of the search tree. The problem can simply be formulated as follows:

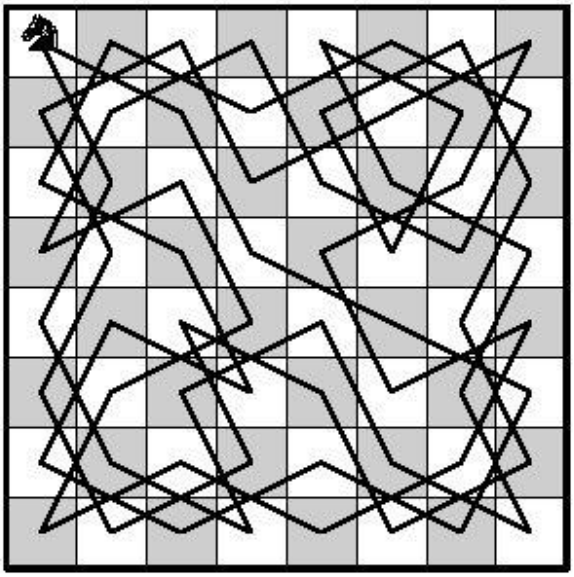
- **Goal test:** any sequence of exactly NxN different positions, obtained after performing valid knight moves.
- **Path cost:** zero
- **Operators:** perform a knight move from a previously visited square to a new unvisited square
- **States:** any sequence of 0 to NxN visited squares.

The formulation strategy is incremental rather than complete-state, because new valid squares are added one by one to the sequence. If the sequence has exactly NxN different squares, then a new solution is found.

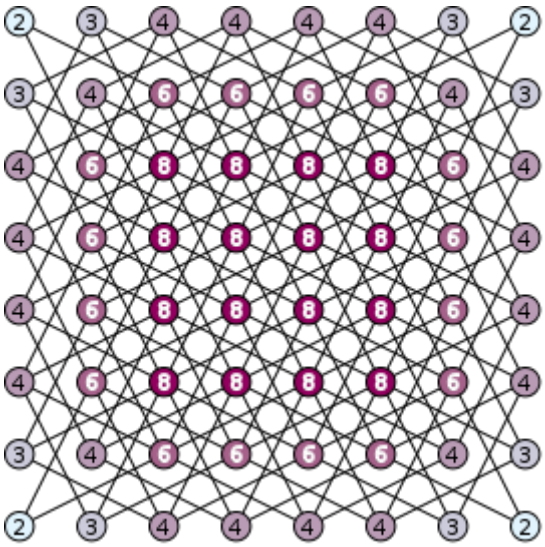
A knight can move in any of 8 ways. In each of these ways, one coordinate of the knight's position changes by 2 units (positively or negatively), and the other coordinate changes by 1 unit (positively or negatively). We can show these coordinate changes as follows:

- ✓ Up two steps, right one step
- ✓ Up two steps, left one step
- ✓ Down two steps, left one step
- ✓ Down two steps, right one step
- ✓ Right two steps, up one step
- ✓ Right two steps, down one step
- ✓ Left two steps, up one step
- ✓ Left two steps, down one step.

The knight's move can be symmetric. Such that, if a knight can move from a square A to a square B. Also, it can move from B to A. You can see solution of knight's tour problem for a chess board that has 8x8 sizes.



We can show these movements in the Knight's tour problem using the graph structure. A graph comprises a set of vertices or nodes and a set of edges or arcs or lines, where each edge joins two nodes.



In our problem, the vertices of the graphs are the squares of the chessboard, and there is an edge joining two neighbor vertices. This graph completely captures the behaviour of the chessboard in the context of the knight's move. Thus, any problem pertaining to the knight's move should be expressible as a problem in terms of properties of this graph.

Problem – Agent Type Classification

As we have formulated our problem above, reaching the target here is important. Therefore, the "artificial intelligence agent" we develop must be targeted to solve the problem.

Our agent type must be a "Goal-based agent". There is an FSM (Finite State Machine) for this agent type. The transition between states is known for any action performed on this agent type. In addition, other states which may be passed in any state are selected which will bring us closer to our target.

Problem – Environment Type Classification

The environments in which our agents are located differ in our problems. These environments may have different characteristics. Classifying the environments according to these properties facilitates the process of problem identification and solution generation. In general, we can examine media types under the following 7 headings:

- **Full observable vs Partially observable:** If it is possible to determine the complete state of the environment at each time point from the percepts it is observable; otherwise it is only partially observable. So, our environment is "partially observable".

- **Deterministic vs Stochastic:** If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is stochastic(non-deterministic). Our environment is "stochastic".

- **Episodic vs Sequential:** In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions in the previous episodes. Episodic environments are much simpler because the agent does not need to think ahead. So, our environment is "sequential (non-deterministic)".

- **Static vs Dynamic:** If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic. So, our environment is "dynamic".

- **Discrete vs Continuous:** If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete (For example, chess); otherwise it is continuous (For example, driving). So, our environment is "discrete".

- **Accessible vs Inaccessible:** If the agent's sensory apparatus can have access to the complete state of the environment, then the environment is accessible to that agent. So, our environment is "inaccessible".

- **Single agent vs Multi-agent:** The environment may contain other agents which may be of the same or different kind as that of the agent. So, our environment is "single agent".

The environment type of Knight's tour problem is "Partially observable", "Stochastic", "Sequential(non-episodic)", "Dynamic", "Discrete", "Inaccessible" and "Single agent".

The Results of Our Program For Different Board Sizes

In our program, we run all the algorithms for different board sizes. We have made various measurements for these algorithms and have seen the following results.

	6	8	10	12	16	20
BFS	Out of Memory	Out of Memory	Out of Memory	Out of Memory	Out of Memory	Out of Memory
DFS	Node:2511583 Time:2 s	Timeout	Timeout	Timeout	Timeout	Timeout
DFS HEURISTIC	Node:36 Time:0 s	Node:64 Time:0 s	Node:100 Time:0 s	Node:144 Time:0 s	Node:256 Time:0 s	Node:400 Time:0 s

- BFS method use too much memory because try to expands all tree but we know that goal is in certain depth so that we only need to expand part of tree. We found a solution for **maximum** of **5x5** board sizes in this algorithm.
- DFS method can be find dead end so that using to much time for big board size. We found a solution for **maximum** of **7x7** board sizes in this algorithm.
- DFS Heuristic escapes dead end and choose node such that it has less child . if number of childs equals it choose node which closer to the edge. So with this heuristic no dead and no backward(algortihm way) this is why it's faster. We found a solution for **maximum** of **170x170** board sizes in this algorithm within 10 minutes.

BOARD SIZE : 6

BFS

```
Enter BoardSize: 6
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):a
Memory status: 'OutOfMemoryError!'
Number of Expanded Node: 11231056

 to exit enter 'q' to continue just press enter
```

DFS

```
Enter BoardSize: 6
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):b

Result Path : [-a1-b3-c5-d3-e5-f3-e1-c2-e3-f1-d2-b1-a3-b5-d6-f5-d4-e6-f4-e2-c1-a2-c3-d5-f6-e4-f2-d1-b2-a4-b6-c4-a5-c6-b4-a6-]
Result Path Matrix:
        35      |       30      |       33      |       14      |       17      |       24      |
        32      |       13      |       2       |       23      |       4       |       15      |
        29      |       34      |       31      |       16      |       25      |       18      |
        12      |       1       |       22      |       3       |       8       |       5       |
        21      |       28      |       7       |       10      |       19      |       26      |
        0       |       11      |       20      |       27      |       6       |       9       |
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 2
Number of Expanded Node: 2511583

 to exit enter 'q' to continue just press enter
```

DFS Heuristic

```
Enter BoardSize: 6
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):c

Result Path : [-a1-b3-a5-c6-e5-f3-e1-c2-a3-b1-d2-f1-e3-f5-d6-b5-d4-e6-f4-e2-c1-a2-b4-a6-c5-d3-f2-e4-f6-d5-b6-c4-b2-a4-c3-d1-]
Result Path Matrix:
        23      |       30      |       3       |       14      |       17      |       28      |
        2       |       15      |       24      |       29      |       4       |       13      |
        33      |       22      |       31      |       16      |       27      |       18      |
        8       |       1       |       34      |       25      |       12      |       5       |
        21      |       32      |       7       |       10      |       19      |       26      |
        0       |       9       |       20      |       35      |       6       |       11      |
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 0
Number of Expanded Node: 36

 to exit enter 'q' to continue just press enter
```

BOARD SIZE : 8

BFS -> “Out Of Memory”

DFS -> “Timeout”

DFS Heuristic

```
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):c

Result Path : [-a1-b3-a5-b7-d8-f7-h8-g6-f8-h7-g5-h3-g1-e2-c1-a2-b4-a6-b8-d7-c5-a4-b2-d1-c3-b1-a3-c2-e1-d3-f2-h1-g3-h5-g7-e8-c7-
Result Path Matrix:
        37      |       18      |       39      |       4       |       35      |       8       |       45      |       6
        40      |       3       |       36      |       19      |       54      |       5       |       34      |       9
        17      |       38      |       53      |       42      |       59      |       44      |       7       |       46
        2       |       41      |       20      |       55      |       52      |       57      |       10      |       33
        21      |       16      |       51      |       58      |       43      |       60      |       47      |       62
        26      |       1       |       24      |       29      |       56      |       63      |       32      |       11
        15      |       22      |       27      |       50      |       13      |       30      |       61      |       48
        0       |       25      |       14      |       23      |       28      |       49      |       12      |       31
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 0
Number of Expanded Node: 64

 to exit enter 'q' to continue just press enter
```

BOARD SIZE : 10

BFS -> “Out Of Memory”

DFS -> “Timeout”

DFS Heuristic

```
Enter BoardSize: 10
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):c

Result Path : [-a1-b3-a5-b7-a9-c10-b8-a10-c9-e10-g9-i10-j8-h9-j10-i8-h10-j9-i7-j5-i3-j1-h2-f1-d2-b1-a3-c2-e1-g2-i1-j3-i5-j7-i9-
Result Path Matrix:
        7       |       40      |       5       |       58      |       9       |       38      |       35      |       16
        4       |       59      |       8       |       39      |       36      |       57      |       10      |       13
        41      |       6       |       61      |       64      |       79      |       70      |       37      |       56
        60      |       3       |       80      |       69      |       62      |       65      |       78      |       85
        81      |       42      |       63      |       76      |       91      |       84      |       71      |       66
        2       |       75      |       90      |       83      |       68      |       77      |       92      |       87
        43      |       82      |       45      |       74      |       93      |       88      |       67      |       72
        26      |       1       |       48      |       89      |       46      |       73      |       98      |       95
        49      |       44      |       27      |       24      |       51      |       94      |       29      |       22
        0       |       25      |       50      |       47      |       28      |       23      |       52      |       99
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 0
Number of Expanded Node: 100

 to exit enter 'q' to continue just press enter
```

BOARD SIZE : 12

BFS -> “Out Of Memory”

DFS -> “Timeout”

DFS Heuristic

```
Enter BoardSize: 12
Enter Time Limit(minutes)  :10
Enter Search Method (a,b,c):c

Result Path : [-a1-b3-a5-b7-a9-b11-d12-f11-h12-j11-l12-k10-j12-l11-k9-l7-k5-l3-k1-i2-g1-e2-c1-a2-b4-a6-b8-a10-b12-c10-a11-c12-
Result Path Matrix:
        55      |       28      |       31      |       6       |       47      |       60      |       33      |       8
        30      |       5       |       54      |       59      |       32      |       7       |       46      |       61
        27      |       56      |       29      |       48      |       53      |       80      |       77      |       44
        4       |       49      |       58      |       81      |       76      |       45      |       102     |       95
        57      |       26      |       75      |       52      |       103     |       94      |       79      |       98
        50      |       3       |       84      |       93      |       82      |       121     |       100     |       109
        25      |       74      |       51      |       104     |       91      |       108     |       131     |       120
        2       |       85      |       92      |       83      |       126     |       117     |       122     |       133
        73      |       24      |       105     |       90      |       107     |       132     |       127     |       138
        88      |       1       |       86      |       125     |       116     |       123     |       118     |       135
        23      |       72      |       89      |       106     |       21      |       70      |       137     |       114
        0       |       87      |       22      |       71      |       124     |       115     |       20      |       69
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 0
Number of Expanded Node: 144
```

BOARD SIZE : 16

BFS -> “Out Of Memory”

DFS -> “Timeout”

DFS Heuristic

```
Enter BoardSize: 16
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):c

Result Path : [-a1-b3-a5-b7-a9-b11-a13-b15-d16-f15-h16-j15-l16-n15-p16-o14-n16-p15-o13-p11-o9-p7-o5-p3-o1-m2-k1-i2-g1-e2-c1-a2-
Result Path Matrix:
        71      |       38      |       41      |       8       |       63      |       76      |       43      |       10
        40      |       7       |       70      |       75      |       42      |       9       |       62      |       125
        37      |       72      |       39      |       64      |       69      |       94      |       77      |       60
        6       |       65      |       74      |       85      |       78      |       61      |       124     |       93
        73      |       36      |       79      |       68      |       95      |       84      |       109     |       128
        66      |       5       |       86      |       83      |       108     |       123     |       92      |       139
        35      |       80      |       67      |       96      |       91      |       110     |       129     |       208
        4       |       87      |       82      |       107     |       122     |       207     |       170     |       131
        81      |       34      |       97      |       90      |       111     |       130     |       209     |       216
        88      |       3       |       106     |       121     |       206     |       169     |       212     |       229
        33      |       98      |       89      |       112     |       119     |       204     |       245     |       210
        2       |       105     |       120     |       205     |       168     |       211     |       228     |       249
        99      |       32      |       113     |       118     |       227     |       250     |       203     |       242
        104     |       1       |       102     |       165     |       116     |       167     |       226     |       251
        31      |       100     |       117     |       114     |       29      |       164     |       253     |       202
        0       |       103     |       30      |       101     |       166     |       115     |       28      |       163
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 0
Number of Expanded Node: 256
```

BOARD SIZE : 20

BFS -> “Out Of Memory”

DFS -> “Timeout”

DFS Heuristic

```
Enter BoardSize: 20
Enter Time Limit(minutes)  :30
Enter Search Method (a,b,c):c

Result Path : [-a1-b3-a5-b7-a9-b11-a13-b15-a17-b19-d20-f19-h20-j19-l20-n19-p20-r19-t20-s18-r20-t19-s17-t15-s13-t11-s9-t7-s5-
Result Path Matrix:
        87      |       48      |       51      |       10      |       79      |       92      |       53      |       12
        50      |       9       |       86      |       91      |       52      |       11      |       78      |       121
        47      |       88      |       49      |       80      |       85      |       110     |       93      |       76
        8       |       81      |       90      |       101     |       94      |       77      |       120     |       109
        89      |       46      |       95      |       84      |       111     |       100     |       137     |       124
        82      |       7       |       102     |       99      |       126     |       119     |       108     |       167
        45      |       96      |       83      |       112     |       107     |       138     |       125     |       208
        6       |       103     |       98      |       127     |       118     |       153     |       166     |       135
        97      |       44      |       113     |       106     |       139     |       134     |       207     |       172
        104     |       5       |       128     |       117     |       152     |       165     |       154     |       233
        43      |       114     |       105     |       140     |       133     |       206     |       173     |       278
        4       |       129     |       116     |       151     |       164     |       155     |       204     |       269
        115     |       42      |       141     |       132     |       205     |       174     |       277     |       202
        130     |       3       |       150     |       163     |       156     |       201     |       270     |       273
        41      |       142     |       131     |       196     |       161     |       274     |       175     |       276
        2       |       149     |       162     |       157     |       192     |       195     |       200     |       293
        143     |       40      |       191     |       160     |       197     |       176     |       275     |       194
        148     |       1       |       146     |       177     |       158     |       193     |       198     |       181
        39      |       144     |       159     |       190     |       37      |       182     |       179     |       188
        0       |       147     |       38      |       145     |       178     |       189     |       36      |       183
Result is : true
Solution status: 'A solution found!'
Execution Time(seconds): 0
Number of Expanded Node: 400
```

BOARD SIZE : 170

BFS -> “Out Of Memory”

DFS -> “Timeout”

DFS Heuristic (Just for max 10 minutes)

We can not show the solution as a picture because the size of the board is too large. If you want to see the result for this size, you can run our program. You may need to make several settings for the IDE you are using before running the program. Your use should give the IDE access to the whole memory of your computer.

Your settings should be as follows:

- -Xss1000m
- -Xms4000m
- -Xmx5000m

```
Enter BoardSize: 170
Enter Time Limit(minutes)  :10
Enter Search Method (a,b,c):c
```

14	923	864	949	920	1007	1052	1125
863	366	915	968	937	1048	991	1072
868	13	924	865	950	969	1008	1053
365	862	867	914	967	936	1047	990
12	869	832	925	866	951	970	1009
831	364	861	900	913	966	935	1046
836	11	870	833	926	901	952	971
363	830	835	860	899	912	965	934
10	837	798	871	834	927	902	953
797	362	829	886	859	898	911	964
802	9	838	799	872	887	928	903
361	796	801	828	885	858	897	910
8	803	788	839	800	873	888	929
787	360	795	854	827	884	857	896
772	7	804	789	840	855	874	889
359	786	773	794	853	826	883	856
6	771	762	805	790	841	792	875
6	771	762	805	790	841	792	875
761	358	785	774	793	852	825	882
766	5	770	763	806	791	842	891
357	760	765	784	775	824	851	878
4	767	732	769	764	807	782	843
731	356	759	776	783	778	823	850
724	3	768	733	758	781	808	779
355	730	725	728	777	744	757	822
2	723	734	745	736	727	780	743
721	354	729	726	739	746	737	756
346	1	722	735	718	349	740	747
353	720	347	344	351	738	717	342
0	345	352	719	348	343	350	741

```
Result is : true
Execution Time(seconds): 148
Number of Expanded Node: 28900
```