Fr. Conceicao Rodrigues College of Engineering

Department of Computer Engineering

EXPERIMENT NO. 3

Practical No:	3
Title:	8-Puzzle Problem
Date of Performance:	12/03/2025
Date of Submission:	27/04/2025
Roll No:	9913
Name of the Student:	Mark Lopes

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Theory understanding(02)	02(Correct)	NA	01 (Tried)	
3	Coding Standards (03): Comments/indention/Naming conventions Output/Test Cases	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Academic Year	2024-25	Estimated Time	Experiment No. 3 – 02 Hours
Course & Semester	T.E. (CE) – Sem. VI	Subject Name	CSC604: Artificial Intelligence
Chapter No.	03	Chapter Title	Solving Problems by Searching
Experiment Type	Software/Finding Solution	Software	Prolog/Python

AIM: Write a program to find a solution to 8-Puzzle problem in Prolog/Python.

I. OBJECTIVES

• To gain insights of informed search algorithms.

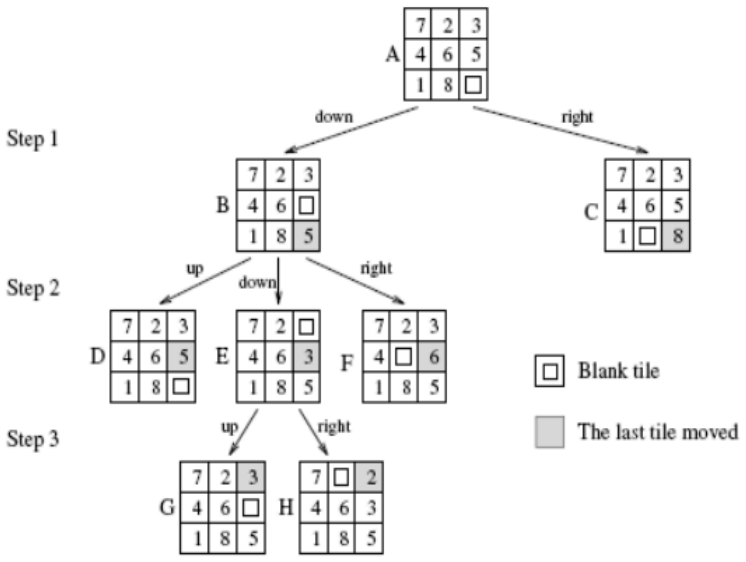
• To be able to implement search space improvement techniques for 8-Puzzle Problem.

2. Finding Optimal puzzle Strategies

The title of this section refers to a familiar and popular sliding tile puzzle that has been around for at least forty years. The most frequent older versions of this puzzle have numbers or letters and the sliding tiles, and the player is supposed to slide tiles into new positions in order to realign a scrambled puzzle back into a goal alignment. For illustration, we use the 3 x 3 8-tile version, which is depicted here in goal configuration.

7	2	3
4	6	5
1	8	

To represent these puzzle "states" we will use a Prolog term representation employing '/' as a separator. The positions of the tiles are listed (separated by '/') from top to bottom, and from left to right. Use "0" to represent the empty tile (space). For example, the goal is ... goal(1/2/3/8/0/4/7/6/5).



3. Attach the screenshot of the code.

```
ids :-
 start(State),
 length(Moves, N),
 dfs([State], Moves, Path), !,
 show([start|Moves], Path),
 format('~nmoves = ~w~n', [N]).
dfs([State|States], [], Path) :-
 goal(State), !,
 reverse([State|States], Path).
dfs([State|States], [Move|Moves], Path) :-
 move(State, Next, Move),
 not(memberchk(Next, [State|States])),
 dfs([Next,State|States], Moves, Path).
show([], _).
show([Move|Moves], [State|States]) :-
 State = state(A,B,C,D,E,F,G,H,I),
 format('~n~w~n~n', [Move]),
 format('~w ~w ~w~n',[A,B,C]),
 format('~w ~w ~w~n',[D,E,F]),
 format('~w ~w ~w~n',[G,H,I]),
 show(Moves, States).
start( state(6,1,3,4,*,5,7,2,0) ).
goal( state(*,0,1,2,3,4,5,6,7) ).
move( state(*,B,C,D,E,F,G,H,J), state(B,*,C,D,E,F,G,H,J), right).
move( state(*,B,C,D,E,F,G,H,J), state(D,B,C,*,E,F,G,H,J), down ).
move( state(A,*,C,D,E,F,G,H,J), state(*,A,C,D,E,F,G,H,J), left ).
move( state(A,*,C,D,E,F,G,H,J), state(A,C,*,D,E,F,G,H,J), right).
move( state(A,*,C,D,E,F,G,H,J), state(A,E,C,D,*,F,G,H,J), down ).
move( state(A,B,*,D,E,F,G,H,J), state(A,*,B,D,E,F,G,H,J), left ).
move( state(A,B,*,D,E,F,G,H,J), state(A,B,F,D,E,*,G,H,J), down ).
move( state(A,B,C,*,E,F,G,H,J), state(*,B,C,A,E,F,G,H,J), up ).
move( state(A,B,C,*,E,F,G,H,J), state(A,B,C,E,*,F,G,H,J), right).
move( state(A,B,C,*,E,F,G,H,J), state(A,B,C,G,E,F,*,H,J), down ).
move( state(A,B,C,D,*,F,G,H,J), state(A,*,C,D,B,F,G,H,J), up ).
move( state(A,B,C,D,*,F,G,H,J), state(A,B,C,D,F,*,G,H,J), right).
move( state(A,B,C,D,*,F,G,H,J), state(A,B,C,D,H,F,G,*,J), down ).
move( state(A,B,C,D,*,F,G,H,J), state(A,B,C,*,D,F,G,H,J), left ).
move( state(A,B,C,D,E,*,G,H,J), state(A,B,*,D,E,C,G,H,J), up ).
move( state(A,B,C,D,E,*,G,H,J), state(A,B,C,D,*,E,G,H,J), left ).
move( state(A,B,C,D,E,*,G,H,J), state(A,B,C,D,E,J,G,H,*), down ).
move( state(A,B,C,D,E,F,*,H,J), state(A,B,C,D,E,F,H,*,J), right).
```

```
move( state(A,B,C,D,E,F,*,H,J), state(A,B,C,*,E,F,D,H,J), up ).
move( state(A,B,C,D,E,F,G,*,J), state(A,B,C,D,E,F,*,G,J), left ).
move( state(A,B,C,D,E,F,G,*,J), state(A,B,C,D,*,F,G,E,J), up ).
move( state(A,B,C,D,E,F,G,*,J), state(A,B,C,D,E,F,G,J,*), right).
move( state(A,B,C,D,E,F,G,H,*), state(A,B,C,D,E,*,G,H,F), up ).
move( state(A,B,C,D,E,F,G,H,*), state(A,B,C,D,E,F,G,*,H), left ).
```

4. Attach the screenshot of the output.

```
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\ai> cd .\exp_4
PS C:\Users\Mark Lopes\Desktop\college\Sem_6\ai\exp_4> swipl .\8-puzzle.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- ids.

start

6 1 3
4 * 5
7 2 0

left

6 1 3
* 4 5
7 2 0

up

* 1 3
6 4 5
7 2 0

right

1 * 3
6 4 5
7 2 0
```

```
down

1 4 3
6 * 5
7 2 0

right

1 4 3
6 5 *
7 2 0

down

1 4 3
6 5 0
7 2 *

left

1 4 3
6 5 0
7 * 2

left

1 4 3
6 5 0
* 7 2

up

1 4 3
* 5 0
6 7 2
```

```
right

1 4 3
5 * 0
6 7 2

right

1 4 3
5 0 *
6 7 2

down

1 4 3
5 0 2
6 7 *

left

1 4 3
5 0 2
6 * 7

left

1 4 3
5 0 2
* 6 7
```

```
up

1 4 3
* 0 2
5 6 7

right

1 4 3
0 * 2
5 6 7

right

1 4 3
0 2 *
5 6 7

up

1 4 *
0 2 3
5 6 7

left

1 * 4
0 2 3
5 6 7
```

```
left

* 1 4
0 2 3
5 6 7

down

0 1 4
* 2 3
5 6 7

right

0 1 4
2 * 3
5 6 7

right

0 1 4
2 3 *
5 6 7

up

0 1 *
2 3 4
5 6 7
```

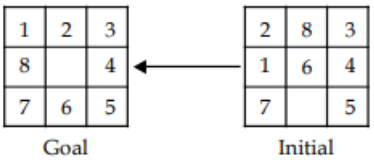
```
left

0 * 1
2 3 4
5 6 7

left

* 0 1
2 3 4
5 6 7

moves = 26
true.

2 ?-
```

5. Conclusion

Therefore i have used depth first search to solve this 8 puzzle problem.

6. Postlab Questions

a. Apply A* and Greedy Best First search on following example.



b. Provide a solution for 8-Puzzle problem without Heuristics and with Heuristics. Comment on Performance.

Initial State

7	2	3
4	6	5
1	8	

Goal State

1	2	3
8	0	4
7	6	5

12/03/25 AI - Postlab -3

a)

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

A* algo:

f(n) = g(n) + h(n)

left

2	8	3
1	6	4
7		5

1+5 = 6

right

2	8	3
1	6	4
7		5

f(n) = 1+5 = 6

up

2	8	3
1		4
7	6	5

f(n) = 1+3 = 4

up

2		3
1	8	4
7	6	5

f(n) = 5

right

2	8	3
1	4	
7	6	5

f(n) = 6

left

2	8	3
	1	4
7	6	5

f(n) = 5

left

	2	3
1	8	4
7	6	5

f(n) = 6

right

2	3	
1	8	4
7	6	5

f(n) = 7

up

	8	3
2	1	4
7	6	5

f(n) = 6

down

2	8	3
7	1	4
	6	5

f(n) = 7

down

1	2	3
	8	4
7	6	5

f(n) = 5

right

1	2	3
8		4
7	6	5

Goal output

right

8		5
2	1	4
7	6	5

f(n) = 7

down

1	2	3
7	8	4
	6	5

f(n) = 7

Greedy - best first search:-