Khalifa University

MSc Computing and Information Science

Muhammad Muneeb - 100052975

Artificial Intelligence
Assignment-1

October 30, 2020

Contents

List of Figures

List of Tables

A Question 1

A.1 Iterative Deepening Search

To implement iterative deepening search, I have implemented depth limited search. Iterative deepening search will call depth limited search for increasing order of depth and depth limited algorithm will call depth first search algorithm with specific depth. Depth first search algorithm will call graph search algorithm with specific depth limit.
Depth first search is generic, it will call Graph Search Algorithm and explore only those nodes which are in in the depth limit. I have modified the node structure so it can store depth information.
For all other algorithms like BFS,DFS,UCS and A-star depth limit will be infinity. So, we can use same graph search for other algorithms.
Code has been commented for better understanding.

A.2 A search with Euclidean Distance heuristics

To implement A* heuristic, calculate the euclidean distance between every node and the goal node and store that distance in a dictionary, where each node is a key and value is the distance. After that pass that distance and the path cost to Priority Queue.

A.3 Space and time complexity

From these graphs we can tell about space and time complexity and completeness of algorithm.
We have finite states, all paths have cost greater than zero so, all algorithms are complete for romania problem.

History Length will tell us about time complexity.

Solution Length will tell us about algorithm being Optimal. For some states all algorithms are optimal but A-star and UCS are optimal for all states.

Max Frontier Length will tell us about Space complexity.
NOTE:
One important thing to notice for GRAPH search space is the way nodes are added to frontier and processed.
We add node to frontier and then process it.
for snode in successors:
frontier.push(snode)
When we add node to frontier we do not check if it is already explored or not. We check it in processing phase. If we check then DFS frontier size will be less than BFS frontier size.
Other important point is that for UCS and A-star we do not have to check

this condition. Otherwise we will not get optimal solution. Because If node is explored again its priority can be changed.
BFS, BFS is not optimal (Path costs are not 1) and but it is complete.
Time: The number of nodes explored by BFS is bounded by $O(b^s)$.
Space: For BFS space is bounded by $O(b^s)$.

DFS, DFS is not optimal and but it is complete because there are finite states.
Time: DFS takes more time as compared to other algorithms. It explores more nodes as compared to other but still bounded by $O(b^m)$.
Space: For DFS space is bounded by $O(b^d)$. Romania problem does not have tree structure. States are connected with each other with bidirectional links.
Kindly see discussion on bottom of page 3.

Astar, Astar is optimal and it is even complete. Solution length for Astar for starting from all states is same as that of the optimal path length starting from all states to goal state.
Time: A-star explores very less states. It explores only those states which are on the optimal path.
Space: We are using admissible heuristic so A-star space is O(n).

UCS, UCS is optimal for all states and it is even complete. Solution length for UCS is same as that of A-star.
Time: The number of nodes explored by UCS are more as compared to A-star. Number of nodes explored by UCS is bounded by $O(b^{(C/e)})$.
Space: $O(b^{(C/e)})$

IDS, IDS is not optimal but it is complete
Time: For some starting states IDS explores same nodes as that of BFS but for some starting states it sometimes explores less states and sometimes more states but still bounded by $O(b^s)$.
Space: Space is bounded by $O(b^s)$.
Kindly see discussion on bottom of page 3.

A.4 Tree search

If we use tree search instead of graph search then algorithm will try to explore all those nodes which have been already explored. It will increase the execution time and space complexity for algorithm.
In case of A-star, Best First search and UCS there is possibility that algorithm will stuck in infinite loop and will never terminate even for finite number of states.
For maximum frontier list I created one new variable maxFrontier to store the frontier size. When new nodes are added into frontier I compare frontier size with maxFrontier. If size increased update max-frontier.
Apart from that we have to return the maxFrontier variable as a part of solution.

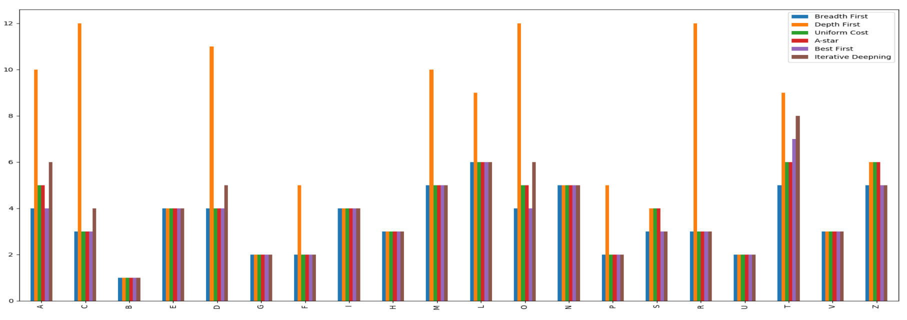


Figure 1: Solution Length

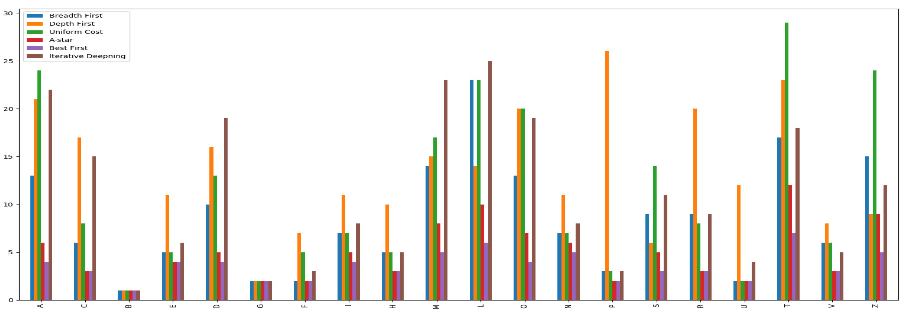


Figure 2: History Length

B Puzzle problem

I tried 5 heuristic function and tested them on 3 by 3 with complete random permutation. Manhattan,Hamming distance, Manhattan plus hamming distance, Euclidean, Numberofinversion (Linear conflicts) + Manhattan distance.
These are the results for all heuristics for 3 by 3 puzzle. 1

start = [(7,2,4),(5,0,6),(8,3,1)]

goal = [(0,1,2),(3,4,5),(6,7,8)]

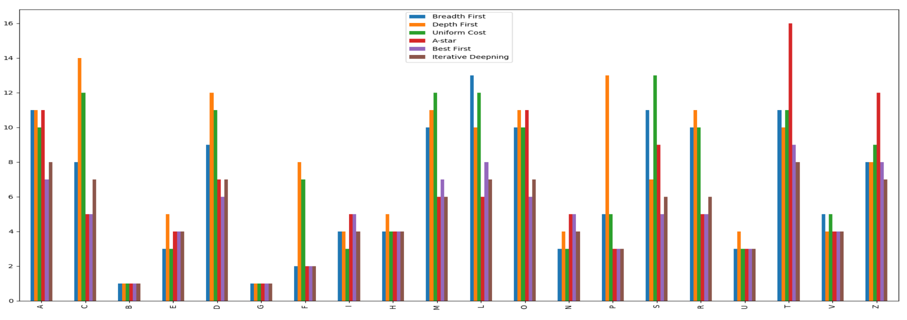


Figure 3: Maximum Frontier Length

	Manhattan	Hamming Distance	Manhattan plus misplaced tiles	Euclidean	Linear Conflict plus Manhattan Distance
Size of closed Set	2773	44681	2023	13583	781
Length of History	4669	83487	3441	23453	1384
Time	1.32	24	1.16	6.3	0.39

Table 1: Results for 3 by 3 Puzzle

B.1 N by N puzzle

I implemented N by N puzzle problem. Below I have attached screenshots of time, history length and number of steps for 4 by 4, 5 by 5 and 6 by 6 for 1 to 20 times varying difficulty in goal state. For 5 by 5 and 6 by 6 I have not included Uniform cost search because it will take time. All values are plotted on log scale. x-axis represents number of mutations in goal state to produce initial state.
I made mutations in goal state to generate the start state. Process is simple, slide "0"(blank) randomly on board using random number generator and allow legal moves only.

B.2 Linear Conflict plus Manhattan heuristic

This heuristic is much better than Manhattan heuristic. For linear conflict I am calculating number of inversions in a matrix.
The set of nodes expanded by A* with Manhattan-linear-conflict is a subset of the nodes expanded by A* with plain Manhattan.
2 tiles are in linear conflict if both tiles are in their target rows or columns, and in the same setting of the Manhattan heuristic, they must pass over each other in order to reach their final goal position. Since we know that in an actual instance of a game, there is no possible way for tiles to actually slide over each other. If such a conflict arises, then 1 of the tiles would need to move out of the aforementioned row or column, and back in again, adding 2 moves to the sum of their Manhattan distances.

B.3 Inversions plus Manhattan heuristic

We can also use Inversions plus Manhattan heuristic.

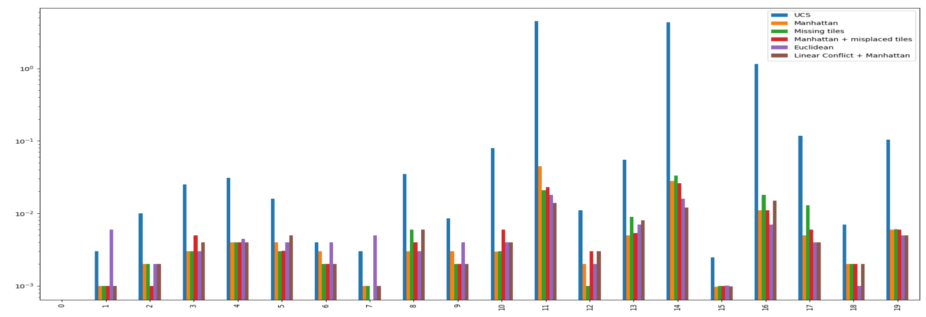


Figure 4: 4 by 4 time

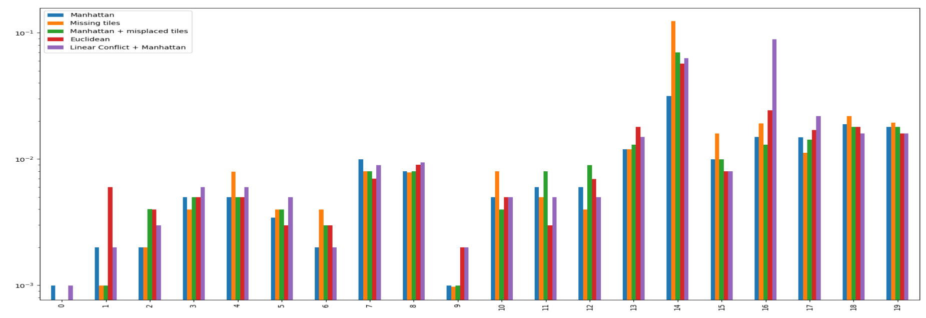


Figure 5: 5 by 5 time

C Solutions

These are the solutions for each algorithm in this order.

[”Breadth First”, ”Depth First”, ”Uniform Cost”,”A-star”,”Best First”,”Iterative Deepning”]

I have also mentioned the starting state on top for each iteration.

Start = A Goal = B

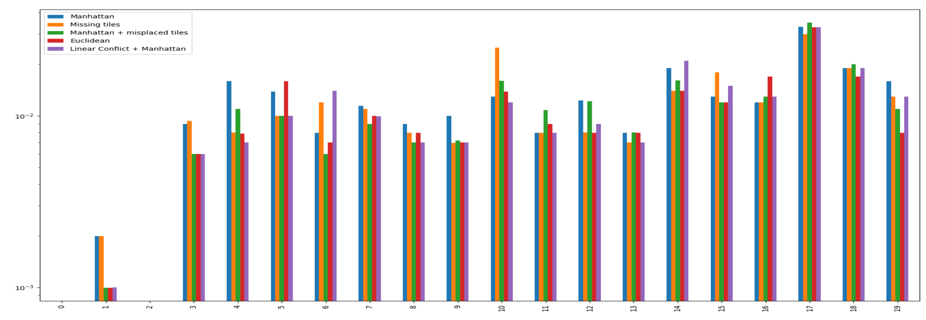


Figure 6: 6 by 6 time

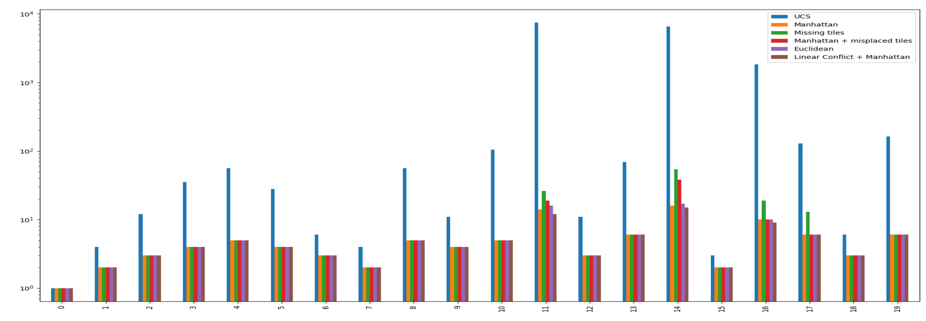


Figure 7: 4 by 4 History length

```
[('A', None), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]

[('A', None), ('T', 'go to T'), ('L', 'go to L'), ('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('R', 'go to R'), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]

[('A', None), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]]

[('A', None), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]]

[('A', None), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]

No Solution at depth 0
No Solution at depth 1
```

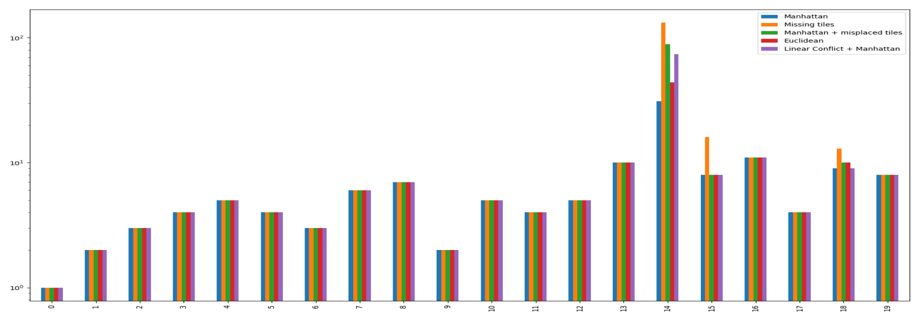


Figure 8: 5 by 5 History length

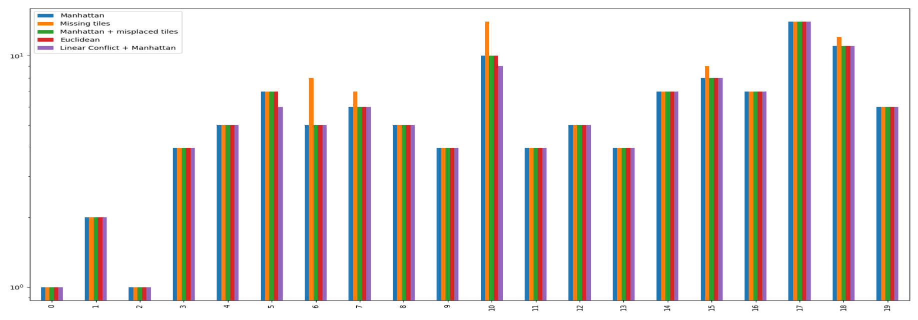


Figure 9: 6 by 6 History length

No Solution at depth 2
No Solution at depth 3
No Solution at depth 4
[('A', None), ('Z', 'go to Z'), ('O', 'go to O'), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]

Start = C Goal = B

[('C', None), ('P', 'go to P'), ('B', 'go to B')]]

[('C', None), ('D', 'go to D'), ('M', 'go to M'), ('L', 'go to L'), ('T', 'go to T'), ('A', 'go to A'), ('Z', 'go to Z'), ('O', 'go to O'), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]]

[('C', None), ('P', 'go to P'), ('B', 'go to B')]]

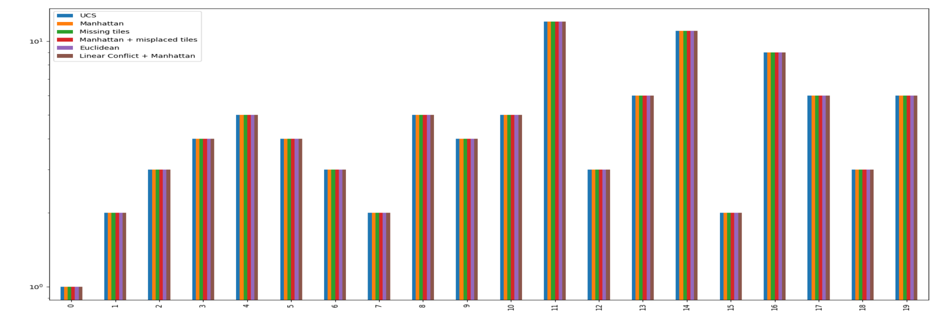


Figure 10: 4 by 4 Steps

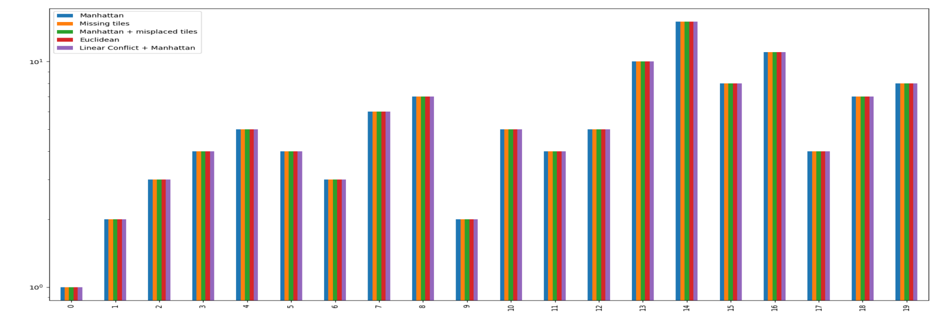


Figure 11: 5 by 5 Steps

```
[('C', None), ('P', 'go to P'), ('B', 'go to B')]]

[('C', None), ('P', 'go to P'), ('B', 'go to B')]]

No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
[('C', None), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]
```

Start = B Goal = B

```
[('B', None)]
[('B', None)]
[('B', None)]
```

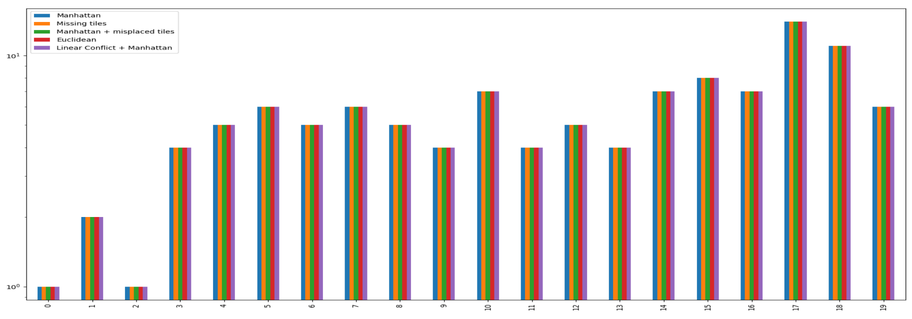


Figure 12: 6 by 6 Steps

```
[('B', None)]
[('B', None)]
[('B', None)]
```

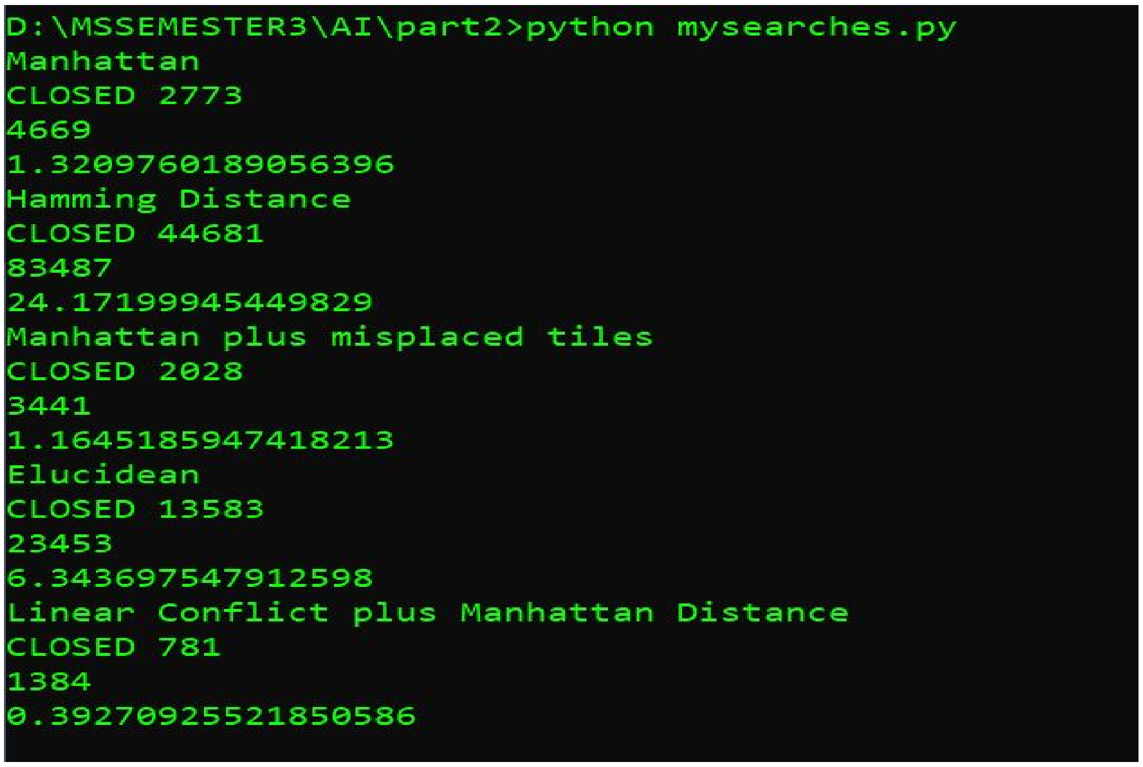
Start = E Goal = B

```
[('E', None), ('H', 'go to H'), ('U', 'go to U'), ('B', 'go to B')]
[('E', None), ('H', 'go to H'), ('U', 'go to U'), ('B', 'go to B')]
[('E', None), ('H', 'go to H'), ('U', 'go to U'), ('B', 'go to B')]
[('E', None), ('H', 'go to H'), ('U', 'go to U'), ('B', 'go to B')]
[('E', None), ('H', 'go to H'), ('U', 'go to U'), ('B', 'go to B')]
No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
[('E', None), ('H', 'go to H'), ('U', 'go to U'), ('B', 'go to B')]
```

Start = D Goal = B

```
[('D', None), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to B')]
[('D', None), ('M', 'go to M'), ('L', 'go to L'), ('T', 'go to T'), ('A', 'go to A'),
('Z', 'go to Z'), ('O', 'go to O'), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'),
('B', 'go to B')]

[('D', None), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to B')]
[('D', None), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to B')]
[('D', None), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to B')]
No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
```



```
D:\MSSEMESTER3\AI\part2>python mysearches.py
Manhattan
CLOSED 2773
4669
1.3209760189056396
Hamming Distance
CLOSED 44681
83487
24.17199945449829
Manhattan plus misplaced tiles
CLOSED 2028
3441
1.1645185947418213
Elucidean
CLOSED 13583
23453
6.343697547912598
Linear Conflict plus Manhattan Distance
CLOSED 781
1384
0.39270925521850586
```

Figure 13: Screen output for 3 by 3

No Solution at depth 3
[('D', None), ('C', 'go to C'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]

Start = G Goal = B

[('G', None), ('B', 'go to B')]
[('G', None), ('B', 'go to B')]
[('G', None), ('B', 'go to B')]
[('G', None), ('B', 'go to B')]
[('G', None), ('B', 'go to B')]

No Solution at depth 0
[('G', None), ('B', 'go to B')]

Start = F Goal = B

[('F', None), ('B', 'go to B')]
[('F', None), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]

```
[('F', None), ('B', 'go to B')]
[('F', None), ('B', 'go to B')]
[('F', None), ('B', 'go to B')]

    No Solution at depth 0 [('F', None), ('B', 'go to B')]
```

Start = I Goal = B

```
    [('I', None), ('V', 'go to V'), ('U', 'go to U'), ('B', 'go to B')]
[('I', None), ('V', 'go to V'), ('U', 'go to U'), ('B', 'go to B')]
[('I', None), ('V', 'go to V'), ('U', 'go to U'), ('B', 'go to B')]
[('I', None), ('V', 'go to V'), ('U', 'go to U'), ('B', 'go to B')]
[('I', None), ('V', 'go to V'), ('U', 'go to U'), ('B', 'go to B')]
No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
[('I', None), ('V', 'go to V'), ('U', 'go to U'), ('B', 'go to B')]
```

Start = H Goal = B

```
[('H', None), ('U', 'go to U'), ('B', 'go to B')]
[('H', None), ('U', 'go to U'), ('B', 'go to B')]
[('H', None), ('U', 'go to U'), ('B', 'go to B')]
[('H', None), ('U', 'go to U'), ('B', 'go to B')]
[('H', None), ('U', 'go to U'), ('B', 'go to B')]
No Solution at depth 0
No Solution at depth 1
[('H', None), ('U', 'go to U'), ('B', 'go to B')]
```

Start = M Goal = B

```
[('M', None), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to
B')]

[('M', None), ('L', 'go to L'), ('T', 'go to T'), ('A', 'go to A'), ('Z', 'go to Z'),
('O', 'go to O'), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]

[('M', None), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to
B')]

[('M', None), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to
B')]

[('M', None), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to
B')]

No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
No Solution at depth 3
No Solution at depth 4
[('M', None), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to P'), ('B', 'go to B')]
```

Start = L Goal = B

```
    [('L', None), ('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to
P'), ('B', 'go to B')]

[('L', None), ('T', 'go to T'), ('A', 'go to A'), ('Z', 'go to Z'), ('O', 'go to O'),
('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]

[('L', None), ('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to
P'), ('B', 'go to B')]

[('L', None), ('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to
P'), ('B', 'go to B')]

[('L', None), ('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to
P'), ('B', 'go to B')]

No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
No Solution at depth 3
No Solution at depth 4
No Solution at depth 5
[('L', None), ('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('P', 'go to P'),
('B', 'go to B')]
```

Start = O Goal = B

```
[('O', None), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]
[('O', None), ('Z', 'go to Z'), ('A', 'go to A'), ('T', 'go to T'), ('L', 'go to L'),
('M', 'go to M'), ('D', 'go to D'), ('C', 'go to C'), ('R', 'go to R'), ('S', 'go to
S'), ('F', 'go to F'), ('B', 'go to B')]

[('O', None), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]
[('O', None), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'), ('B', 'go to B')]
[('O', None), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]
```

```
No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
No Solution at depth 3
No Solution at depth 4
[(’O’, None), (’Z’, ’go to Z’), (’A’, ’go to A’), (’S’, ’go to S’), (’F’, ’go to F’),
(’B’, ’go to B’)]
```

Start = N Goal = B

```
[(’N’, None), (’I’, ’go to I’), (’V’, ’go to V’), (’U’, ’go to U’), (’B’, ’go to B’)]
[(’N’, None), (’I’, ’go to I’), (’V’, ’go to V’), (’U’, ’go to U’), (’B’, ’go to B’)]
[(’N’, None), (’I’, ’go to I’), (’V’, ’go to V’), (’U’, ’go to U’), (’B’, ’go to B’)]
[(’N’, None), (’I’, ’go to I’), (’V’, ’go to V’), (’U’, ’go to U’), (’B’, ’go to B’)]
[(’N’, None), (’I’, ’go to I’), (’V’, ’go to V’), (’U’, ’go to U’), (’B’, ’go to B’)]
No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
No Solution at depth 3
[(’N’, None), (’I’, ’go to I’), (’V’, ’go to V’), (’U’, ’go to U’), (’B’, ’go to B’)]
```

Start = P Goal = B

```
[(’P’, None), (’B’, ’go to B’)]
[(’P’, None), (’R’, ’go to R’), (’S’, ’go to S’), (’F’, ’go to F’), (’B’, ’go to B’)]
[(’P’, None), (’B’, ’go to B’)]
[(’P’, None), (’B’, ’go to B’)]
[(’P’, None), (’B’, ’go to B’)]

No Solution at depth 0
[(’P’, None), (’B’, ’go to B’)]
```

Start = S Goal = B

```
[(’S’, None), (’F’, ’go to F’), (’B’, ’go to B’)]
[(’S’, None), (’R’, ’go to R’), (’P’, ’go to P’), (’B’, ’go to B’)]
[(’S’, None), (’R’, ’go to R’), (’P’, ’go to P’), (’B’, ’go to B’)]
[(’S’, None), (’R’, ’go to R’), (’P’, ’go to P’), (’B’, ’go to B’)]
[(’S’, None), (’F’, ’go to F’), (’B’, ’go to B’)]
No Solution at depth 0
No Solution at depth 1
[(’S’, None), (’F’, ’go to F’), (’B’, ’go to B’)]
```

Start = R Goal = B

```
[(’R’, None), (’P’, ’go to P’), (’B’, ’go to B’)] [(’R’, None), (’S’, ’go to S’),
(’O’, ’go to O’), (’Z’, ’go to Z’), (’A’, ’go to A’), (’T’, ’go to T’), (’L’, ’go to L’),
(’M’, ’go to M’), (’D’, ’go to D’), (’C’, ’go to C’), (’P’, ’go to P’), (’B’, ’go to
B’)][(’R’, None), (’P’, ’go to P’), (’B’, ’go to B’)]
[(’R’, None), (’P’, ’go to P’), (’B’, ’go to B’)]
[(’R’, None), (’P’, ’go to P’), (’B’, ’go to B’)]
No Solution at depth 0
No Solution at depth 1
[(’R’, None), (’P’, ’go to P’), (’B’, ’go to B’)]
```

Start = U Goal = B

```
[(’U’, None), (’B’, ’go to B’)]
[(’U’, None), (’B’, ’go to B’)]
[(’U’, None), (’B’, ’go to B’)]
[(’U’, None), (’B’, ’go to B’)]
[(’U’, None), (’B’, ’go to B’)]
No Solution at depth 0
[(’U’, None), (’B’, ’go to B’)]
```

Start = T Goal = B

```
[(’T’, None), (’A’, ’go to A’), (’S’, ’go to S’), (’F’, ’go to F’), (’B’, ’go to B’)]

[(’T’, None), (’L’, ’go to L’), (’M’, ’go to M’), (’D’, ’go to D’), (’C’, ’go to
C’), (’R’, ’go to R’), (’S’, ’go to S’), (’F’, ’go to F’), (’B’, ’go to B’)]

[(’T’, None), (’A’, ’go to A’), (’S’, ’go to S’), (’R’, ’go to R’), (’P’, ’go to P’),
(’B’, ’go to B’)]

[(’T’, None), (’A’, ’go to A’), (’S’, ’go to S’), (’R’, ’go to R’), (’P’, ’go to P’),
(’B’, ’go to B’)]

[(’T’, None), (’L’, ’go to L’), (’M’, ’go to M’), (’D’, ’go to D’), (’C’, ’go to
C’), (’P’, ’go to P’), (’B’, ’go to B’)]

No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
No Solution at depth 3
No Solution at depth 4
No Solution at depth 5
No Solution at depth 6 [(’T’, None), (’L’, ’go to L’), (’M’, ’go to M’), (’D’, ’go
to D’), (’C’, ’go to C’), (’R’, ’go to R’), (’P’, ’go to P’), (’B’, ’go to B’)]
```

Start = V Goal = B

```
[('V', None), ('U', 'go to U'), ('B', 'go to B')]
[('V', None), ('U', 'go to U'), ('B', 'go to B')]
[('V', None), ('U', 'go to U'), ('B', 'go to B')]
[('V', None), ('U', 'go to U'), ('B', 'go to B')]
[('V', None), ('U', 'go to U'), ('B', 'go to B')]
No Solution at depth 0
No Solution at depth 1
[('V', None), ('U', 'go to U'), ('B', 'go to B')]
```

Start =Z Goal = B

```
[('Z', None), ('A', 'go to A'), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]


[('Z', None), ('O', 'go to O'), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'),
('B', 'go to B')]

[('Z', None), ('A', 'go to A'), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'),
('B', 'go to B')]

[('Z', None), ('A', 'go to A'), ('S', 'go to S'), ('R', 'go to R'), ('P', 'go to P'),
('B', 'go to B')]

[('Z', None), ('A', 'go to A'), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]

No Solution at depth 0
No Solution at depth 1
No Solution at depth 2
No Solution at depth 3
[('Z', None), ('O', 'go to O'), ('S', 'go to S'), ('F', 'go to F'), ('B', 'go to B')]
```

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891–921, 1905.

[3] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`