|  |
| --- |
| **Istanbul University - Cerrahpaşa Computer Engineering Artificial Intelligence and Expert Systems** |

A picture containing icon

Description automatically generated



Assoc. Prof. Zeynep ORMAN

Müeyyed ĞARZUDDİN

1306180132

8-Puzzle

ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

Create Problem

**The 8 puzzle consists of 8 numbered, movable tiles put in a 3 X 3 frame . One location of the se is always free thus it makes it possible to move an adjacent numbered tile into the free location.**

## **Example:**

1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |  |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 |  | 5 |
| 7 | 8 | 6 |

|  |  |  |
| --- | --- | --- |
| 1 |  | 3 |
| 4 | 2 | 5 |
| 7 | 8 | 6 |

2

3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |  |

End

Start

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 |  |
| 7 | 8 | 6 |

## Problem Configuration:

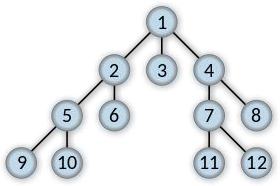
* + **Situations**: it is a n array of size of 9.
  + **Start Point**: new input or random created in file ./samples.txt.
  + **Actions**: movements [ UP , DOWN , LEFT , RIGHT ]
  + One movement can be applied depending on the location of blank
  + **Movement model**: takes [situation+ start point] --------- >  **return**  current state.
  + Representedby *swap* method.
  + **End Check**: check whether it is the End state or not.
  + **Path Cost:** One step costs 1, Thus path cost is the number of steps in the path.

# Search algorithms

**Breadth-first search(BFS)** is a simple strategy in

which the root node is expanded first, then all the

successors of the root node are expanded next,

then their successors, and so on.



## Depth-first search (DFS)

**DFS is an algorithm for exploring or searching tree or graph in data structure. it starts at the root node and expand along as it is possible to the deepest depth before it goes backtracking.**

A picture containing diagram

Description automatically generated

## Depth limited search (DLS)

**DLS is an algorithm that resembles to DFS**, however DLS the depth of it is predetermined . Node at deepest depth are handled as they have no successors.

A picture containing watch

Description automatically generated

## Iterative deepening depth-first search (IDS)

A picture containing blur

Description automatically generated**IDS is a version of DFS which runs which runs repeatedly with increasingly depth limits until the goal is found**

**IDS : is an optimal version which uses less memory 1at each iteration . IDS is an optimal alg such as BFS but uses less memory ; at each loop , it explores nodes in the search tree in the same order as DFS , but the cumulative order in which nodes are first visited is effectively breadth-first.**

**Applying the algorithm on 10 random problems:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PROBLEM** | **Type** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| **TIME** | BFS | 0.352 | 0.001 | 0.083 | 0.004 | 0.002 | 0.023 | 0.005 | 1.722 | 0.078 | 0.016 | 0.228 |
| DFS | 151.9 | 0.102 | 25.Haz | 0.181 | 0.001 | 27.Haz | Haz.95 | 12.Haz | Mar.69 | 0.041 | 22.86 |
| DLS [5] | - | 0.001 | - | - | 0.001 | - | 0.001 | - | - | - | 0.001 |
| DLS [10] | - | 0.005 | 0.049 | 0.015 | 0.027 | 0.013 | 0.04 | - | - | 0.021 | 0.024 |
| DLS [15] | 0.066 | 0.05 | 0.21 | 0.096 | 0.067 | 0.11 | 0.032 | - | 0.284 | 0.238 | 0.128 |
| DLS [20] | 0.837 | 0.12 | 0.91 | 0.153 | 0.692 | 0.827 | 0.072 | 1.796 | 0.242 | 0.96 | 0.660 |
| DLS [25] | 1.134 | 1.304 | Şub.37 | 1.643 | 1.986 | 2.565 | 0.625 | 4.268 | 0.77 | 2.974 | 1.963 |
| IDS | 0.67 | 0.002 | 0.067 | 0.007 | 0.001 | 0.022 | 0.004 | 2.877 | 0.132 | 0.024 | 0.380 |
| **DEPTH** | BFS | 15 | 3 | 10 | 6 | 3 | 8 | 5 | 18 | 11 | 8 | 8.Tem |
| DFS | 88939 | 829 | 40574 | 2194 | 27 | 40568 | 23753 | 28878 | 17771 | 458 | 24399.1 |
| DLS [5] | - | 3 | - | - | 3 | - | 5 | - | - | - | 3.666 |
| DLS [10] | - | 9 | 10 | 6 | 3 | 8 | 5 | - | - | 8 | 7 |
| DLS [15] | 15 | 9 | 10 | 12 | 15 | 8 | 11 | - | 11 | 8 | 11 |
| DLS [20] | 19 | 19 | 20 | 20 | 19 | 8 | 19 | 20 | 17 | 20 | 18.Oca |
| DLS [25] | 25 | 19 | 24 | 22 | 21 | 24 | 25 | 24 | 23 | 24 | 23.Oca |
| IDS | 15 | 3 | 10 | 6 | 3 | 8 | 5 | 18 | 11 | 8 | 8.Tem |
| **EXPANDED** | BFS | 5667 | 13 | 590 | 68 | 15 | 219 | 55 | 22458 | 1177 | 236 | 3049.8 |
| DFS | 94292 | 845 | 41788 | 2240 | 27 | 41782 | 24361 | 29617 | 18191 | 467 | 25361 |
| DLS [5] | - | 19 | - | - | 19 | - | 31 | - | - | - | 23 |
| DLS [10] | - | 45 | 263 | 194 | 265 | 87 | 427 | - | - | 306 | 226.7 |
| DLS [15] | 978 | 489 | 3073 | 1410 | 1032 | 1017 | 473 | - | 4196 | 3493 | 1795.6 |
| DLS [20] | 10439 | 1769 | 12361 | 2361 | 10109 | 10092 | 1168 | 22801 | 3785 | 13227 | 8811.2 |
| DLS [25] | 14511 | 16566 | 30966 | 22587 | 26060 | 30276 | 6769 | 51335 | 10933 | 38568 | 24857.1 |
| IDS | 9631 | 12 | 881 | 81 | 12 | 309 | 64 | 41310 | 1819 | 310 | 5442.9 |
| **FRINGE** | BFS | 5820 | 20 | 612 | 80 | 18 | 237 | 69 | 20590 | 1201 | 248 | 2889.5 |
| DFS | 113352 | 1060 | 51628 | 2800 | 39 | 51619 | 30212 | 36722 | 22604 | 589 | 31062.5 |
| DLS [5] | - | 8 | - | - | 9 | - | 9 | - | - | - | 8.Haz |
| DLS [10] | - | 15 | 15 | 14 | 15 | 15 | 15 | - | - | 15 | 14.Ağu |
| DLS [15] | 22 | 21 | 21 | 21 | 23 | 21 | 23 | - | 21 | 21 | 21.May |
| DLS [20] | 29 | 28 | 27 | 27 | 29 | 29 | 29 | 27 | 22 | 27 | 27.Nis |
| DLS [25] | 35 | 34 | 34 | 34 | 36 | 35 | 36 | 33 | 34 | 34 | 34.5 |
| IDS | 22 | 5 | 15 | 9 | 7 | 13 | 9 | 25 | 16 | 12 | 13.Mar |

# Conclusion

By comparing these algorithms , keeping in mind cost =1 , and each one has its pros and cons

BFS DFS

Pros

1. Every time find solution

CONS

1. BIG USE OF MEMORY
2. Explore a lot of nodes

Pros

1. Every time find solution
2. Solution would be optimized with regarding to depth

CONS

1. BIG USE OF MEMORY

DLS IDS

Pros

1. Every time find solution
2. Optimized use of memory
3. Optimized solution

CONS

1. It would be not quick if the solution is exit

Pros

1. High speed in running ( depends on limit of depth)

CONS

1. Not able to get solution if case depth is small