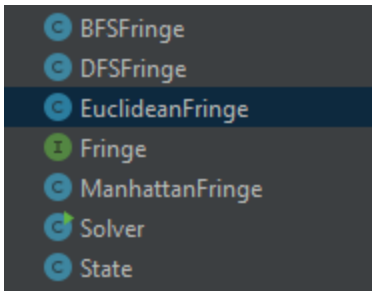
8Puzzle Assignment

01.11.2021

Mohamed Ibrahim	50
Mohamed Salah	52
Marwan Tarek	59

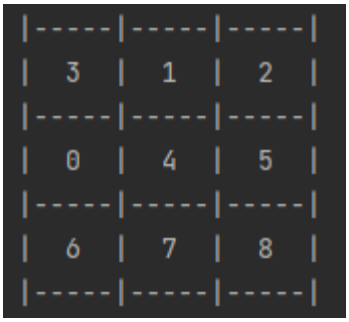
Overview

Classes:



Fringe interface is used for implementing each search technique.

State class is used to represent each state, we assumed that each state is encoded into a single integer, for example the following state:



Is represented by the integer **312045678**

Solver class is used as the main algorithm and is described below

Data structures and algorithms

Solver Algorithm:

1. Check if our **Fringe** is not empty and still has unexplored states.
2. If so, get the next state according to the algorithm used (for **DFS** it is the top of the stack, for **BFS & A*** it is the head of the queue).
3. If this is the goal state then we return.
4. If not, we check the possible moves from that state and add them to the fringe (push the stack for **DFS**, or enqueue for **BFS & A***) then repeat.

Used Data Structures:

1. A hash set is used to store states that were already visited.
2. Other data structures were implemented according to the implementation given in the assignment, priority queue for A*, normal queue for BFS, and a stack for DFS.

Sample runs

1. State **312645078** (a state that is close to the goal)

```
USING DFS :
running time is 35 microseconds
number of expanded nodes is 3
search depth is 2
Path cost is 2

|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  6  |  4  |  5  |
|-----|-----|-----|
|  0  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  0  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  0  |  1  |  2  |
|-----|-----|-----|
|  3  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|
```

```
USING BFS :
running time is 35 microseconds
number of expanded nodes is 7
search depth is 2
Path cost is 2

|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  6  |  4  |  5  |
|-----|-----|-----|
|  0  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  0  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  0  |  1  |  2  |
|-----|-----|-----|
|  3  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|
```

```
USING A* (manhattan as heuristic) :
running time is 34 microseconds
number of expanded nodes is 3
search depth is 2
Path cost is 2

|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  6  |  4  |  5  |
|-----|-----|-----|
|  0  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  0  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  0  |  1  |  2  |
|-----|-----|-----|
|  3  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|
```

```
USING A* (euclidean as heuristic) :
running time is 53 microseconds
number of expanded nodes is 3
search depth is 2
Path cost is 2

|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  6  |  4  |  5  |
|-----|-----|-----|
|  0  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  3  |  1  |  2  |
|-----|-----|-----|
|  0  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|


|-----|-----|-----|
|  0  |  1  |  2  |
|-----|-----|-----|
|  3  |  4  |  5  |
|-----|-----|-----|
|  6  |  7  |  8  |
|-----|-----|-----|
```

2. State 145028763 (random state)

```
USING DFS :
running time is 99818 microseconds
number of expanded nodes is 153655
search depth is 114929
Path cost is 102125
```

```
USING BFS :
running time is 39675 microseconds
number of expanded nodes is 18245
search depth is 17
Path cost is 17
```

```
USING A* (euclidean as heuristic) :
running time is 27213 microseconds
number of expanded nodes is 5240
search depth is 153
Path cost is 153
```

```
USING A* (manhattan as heuristic) :
running time is 12186 microseconds
number of expanded nodes is 5240
search depth is 153
Path cost is 153
```

It is very clear that A* performs better than BFS and DFS.