

Programming Project 1

Team: Dikshali Margaj - 801076473

Upasana Pattnaik – 801081007

Language used - Java

Tool used - Eclipse

1. Introduction
2. Code Structure Explanation
3. Sample Input and Output

1. Introduction:

The 8-puzzle is a game which contains 8 numbered tiles and one empty slot in a square board. The empty slot can be used to slide the tiles around the board. The goal of the game is to slide the tiles till the numbers are ordered in ascending order/descending order/specified order.

For example, if the order of tiles is:

2,4,3

1,5,7

6,[],8

The goal state would be to order:

1,2,3

4,5,6

7,8,[]

The tiles can be moved up, down, left and right depending on the location of the empty slot[].

In this program, our aim is to sort the 8-puzzle game by calculating the heuristic and cost of every move and working through the puzzle based on the lowest heuristic cost path.

2. Code Structure

- Demo.java: This contains the main function which has the initial state and goal state. The program starts here. The Node object is created and passed into Solve class which passes the initial state and starts the program. The heuristic is to be calculated based on the option selected. 1- Manhattan Distance 2- Misplaced Tiles.
- Node.java: The Node is initialized. The input, child, parent, goal cost, heuristic and total cost of the node is initialized here.
- Solve.java: Creates openList and closedList. It adds the recently created node to current node. Then we use a while loop to check if the current node is not null and not equal to the goal state. In this case we generate successors of the node, calculate the distance (using Movement.java), calculate heuristic and add them to the openList based on the lowest heuristic value. If the current node is not the goal state, we add it to the

closedList. We compare the values of the nodes based on cost and replace them in the priority queue(openList) based on lowest cost.

- Movement.java: Calculates the possible moves based on location of 0.
 1. If the index is 0, 0 can move right, down
 2. If the index is 1, 0 can move left, down, up
 3. If the index is 2, 0 can move left, down
 4. If the index is 3, 0 can move right, down, up
 5. If the index is 4, 0 can move right, down, up, left
 6. If the index is 5, 0 can move down, up, left
 7. If the index is 6, 0 can move up, right
 8. If the index is 7, 0 can move right, up, left
 9. If the index is 8, 0 can move up, left
- CompareCost.java: This class is used to compare the cost of two nodes and return a value based on which one is greater or lesser.
- ParentData.java: This is used to initialize the parent node.

3. **Sample Input and Output**(nodes generated(openList) and nodes expanded(openList +closedList) are in the sample outputs)

Sample Input 1:

Initial State:

123

745

680

Goal State:

123

864

750

Manhattan Distance

```
<terminated> Demo [Java App]
Reached Goal
cost : 0
-----
123
745
680
cost : 5
-----
123
740
685
cost : 5
-----
123
704
685
cost : 6
-----
123
784
605
cost : 6
-----
123
784
065
cost : 7
-----
123
084
765|
cost : 7
-----
123
```

```
-----  
123  
084  
765|  
cost : 7  
-----
```

```
-----  
123  
804  
765  
cost : 8  
-----
```

```
-----  
123  
864  
705  
cost : 8  
-----
```

```
-----  
123  
864  
750  
-----
```

```
-----  
Number of moves: 8  
Total cost: 52  
Close List: 34  
Open List: 168  
Total Nodes Generated: 202
```

Sample 2:

Initial State:

281
346
750

Goal State:

321
804
756

Manhattan Distance

```
<terminated> Demo [Java Applica
Reached Goal
cost : 0
-----
281
346
750
cost : 4
-----
281
340
756
cost : 4
-----
281
304
756
cost : 5
-----
201
384
756
cost : 5
-----
201
```

```
-----  
201  
384  
756  
cost : 5  
-----  
021  
384  
756  
cost : 6  
-----  
321  
084  
756  
cost : 6  
-----  
321  
804  
756  
-----  
Number of moves: 6  
Total cost: 30  
Close List: 10  
Open List: 28  
Total Nodes Generated: 38  
Time :14 ms
```

MISPLACED TILES

<terminated> Demo [Java Application] C:\Pr

Reached Goal

cost : 0

281

346

750

cost : 4

281

340

756

cost : 7

281

304

756

cost : 6

201

384

756

cost : 5

001

```
384
756
cost : 5
-----
021
384
756
cost : 3
-----
321
084
756
cost : 1
-----
321
804
756
-----
Number of moves: 6
Total cost: 26
Close List: 7
Open List: 13
Total Nodes Generated: 20
Time :0 ms
```

Sample 3
Initial State
123
405
678
Goal State
123
456
780

<terminated> Demo [Java Application]

Reached Goal

cost : 0

123

405

678

cost : 5

123

450

678

cost : 6

123

458

670

cost : 8

123

458

607

cost : 8

123

458

067

cost : 9

123

058

467

cost : 11

100

```
-----  
123  
058  
467  
cost : 11  
-----  
123  
508  
467  
cost : 12  
-----  
123  
568  
407  
cost : 12  
-----  
123  
568  
470  
cost : 13  
-----  
123  
560  
478  
cost : 13  
-----  
123  
506  
478  
cost : 13  
-----  
---
```

```
500
478
cost : 13
-----
123
056
478
cost : 14
-----
123
456
078
cost : 14
-----
123
456
708
cost : 14
-----
123
456
780
-----
Number of moves: 14
Total cost: 152
Close List: 389
Open List: 37947
Total Nodes Generated: 38336
```