Matt Marshall

Mvm30

Programming Assignment 1

**Code Design**

**EightPuzzleBoard.java**

The creation of the EightPuzzleBoard was relatively straightforward. The data of the puzzle is stored in an int array, and the puzzle state keeps track of where the blank is.

Three different constructors:

**public** EightPuzzleBoard()

**public** EightPuzzleBoard(String s)

**public** EightPuzzleBoard(**int** [] temp, **int** blank)

The first if given nothing will simply create a goal state puzzle. The other two take two different forms of data to create the same puzzle. Both will be set by the following methods:

**public** **void** setData(String input)

**public** **void** setData(**int**[] input, **int** blank)

Both use a for loop to create the initial puzzleboard state. The blank is kept track of to make the move function easier to use. By keeping track of it no loops were needed to move spaces on the board, which helps efficiency.

**public** **int** h1()

**public** **int** h2()

The method to get the heuristics are stored inside this class as well, since this class holds the primary data.

**SearchNode.java**

**public** **class** SearchNode **implements** Comparable<SearchNode>

One of the main design choices was that SearchNode implemented the Comparable class. This was so that when I used a PriorityQueue, every node added would automatically be sorted.

Also the genNext() method:

**public** **static** LinkedList<SearchNode> genNext(SearchNode node, String h)

is in this class, which generates all possible different successors given a SearchNode n. I originally had this in the EightPuzzleBoard class but found myself converting them to SearchNodes anyway. This is all done with O(1) efficiency, as no loops are called.

**AStar.java**

Because the SearchNode class is a bit chunky, this one was beautifully simplistic. The data is stored inside of a PriorityQueue, since the findMin() method (which in this case is always the first, has an O(1) efficiency.

The memory is stored in a linked list, as adding each new bit of data to the front has O(1) efficiency. Unfortunately searching through it does take O(N), but I could think of any way to avoid this barring implementing a search algorithm.

Because it is stored as such, data can be quickly retrieved, and added to. Using two different data structures for mapping and for memory make this possible.

**LocalBeam.java**

Again because SearchNode contained a lot of the bulky code, modifying the A-Star class to do beam search was not very difficult. This class also uses a PriorityQueue to store the map data and a LinkedList to store the memory.

There are two main approach differences between the A-Star method (beyond the whole “they do two different things” part). The first is that local beam has temporary memory that gets overwritten in the form of another PriorityQueue, the second is that it has two main constructors because I can’t code efficiently. This doesn’t hurt the actual runtime, but makes for more code.

**Solver.java**

Basically it implements a scanner.

**Code Correctness**

**setState <state>**

When given a sting in the desired format of (1-8) + b, the result is as follows:

Example 1:

- setState b12345678

- printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

- setState 1234b5678

- printState

[1, 2, 3, 4, 0, 5, 6, 7, 8]

If an invalid amount of characters are entered, a user will get the following response, and the state will not be set:

-setState bbbb

Invalid String Format

If anything that isn’t a b is in it, it will also give a response and not set the state:

setState 12345678i

Invalid String Format

Bugs:

System will allow input that contains multiples of the same numbers:

setState b12345667

System Set

**randomizeState n**

(Every time I tried to set the seed Eclipse crashed. I blame communist Russia)

Base Case:

- setState b12345678

System Set

- randomizeState 0

- printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

1 Case:

- randomizeState 1

- printState

[3, 1, 2, 0, 4, 5, 6, 7, 8]

N Case:

randomizeState 10000

[4, 0, 7, 5, 1, 8, 3, 6, 2]

Invalid (if given anything that is not an int), will return following message:

randomizeState b

Invalid input

**printState**

Base Case:

printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

(Goal is default)

N Case

setState 1234b5678

System Set

printState

[1, 2, 3, 4, 5, 0, 6, 7, 8]

**move <direction>**

Up

setState 12345678b

printState

[1, 2, 3, 4, 5, 6, 7, 8, 0 ]

move up

printState

[3, 1, 2, 0, 4, 5, 6, 7, 8]

move up

printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

move up

printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

setState 1234567b8

printStat

[1, 2, 3, 4, 5, 6, 7, 0, 8]

move up

printState

[1, 2, 3, 4, 0, 6, 7, 5, 8]

move up

printState

[1, 0, 3, 4, 2, 6, 7, 5, 8]

move up

printState

[1, 0, 3, 4, 2, 6, 7, 5, 8]

setState 123456b78

printState

[1, 2, 3, 4, 5, 6, 0, 7, 8]

move up

printState

[1, 2, 3, 0, 5, 6, 4, 7, 8]

move up

printState

[0, 2, 3, 1, 5, 6, 4, 7, 8]

move up

printState

[0, 2, 3, 1, 5, 6, 4, 7, 8]

Down

setState b12345678

printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

move down

printState

[3, 1, 2, 0, 4, 5, 6, 7, 8]

move down

printState

[3, 1, 2, 6, 4, 5, 0, 7, 8]

move down

printState

[3, 1, 2, 6, 4, 5, 0, 7, 8]

setState 1b2345678

printState

[1, 0, 2, 3, 4, 5, 6, 7, 8]

move down

printState

[1, 4, 2, 3, 0, 5, 6, 7, 8]

move down

printState

[1, 4, 2, 3, 7, 5, 6, 0, 8]

move up

printState

[1, 4, 2, 3, 7, 5, 6, 0, 8]

setState 12b345678

printState

[1, 2, 0, 3, 4, 5, 6, 7, 8]

move down

printState

[1, 2, 5, 3, 4, 0, 6, 7, 8]

move down

printState

[1, 2, 5, 3, 4, 8, 6, 7, 0]

move down

printState

[1, 2, 5, 3, 4, 8, 6, 7, 0]

Right

setState b12345678

printState

[0, 1, 2, 3, 4, 5, 6, 7, 8]

move right

printState

[1, 0, 2, 3, 4, 5, 6, 7, 8]

move right

printState

[1, 2, 0, 3, 4, 5, 6, 7, 8]

move right

printState

[1, 2, 0, 3, 4, 5, 6, 7, 8]

setState 123b45678

printState

[1, 2, 3, 0, 4, 5, 6, 7, 8]

move right

printState

[1, 2, 3, 4, 0, 5, 6, 7, 8]

move right

printState

[1, 2, 3, 4, 5, 0, 6, 7, 8]

move right

printState

[1, 2, 3, 4, 5, 0, 6, 7, 8]

setState 123456b78

printState

[1, 2, 3, 4, 5, 6, 0, 7, 8]

move right

printState

[1, 2, 3, 4, 5, 6, 7, 0, 8]

move right

printState

[1, 2, 3, 4, 5, 6, 7, 8, 0]

move right

printState

[1, 2, 3, 4, 5, 6, 7, 8, 0]

Left

setState 12b345678

printState

[1, 2, 0, 3, 4, 5, 6, 7, 8]

move left

printState

[1, 0, 2, 3, 4, 5, 6, 7, 8]

move left

printState

[1, 2, 0, 3, 4, 5, 6, 7, 8]

move left

printState

[1, 2, 0, 3, 4, 5, 6, 7, 8]

setState 123b45678

printState

[1, 2, 3, 0, 4, 5, 6, 7, 8]

move left

printState

[1, 2, 3, 4, 0, 5, 6, 7, 8]

move left

printState

[1, 2, 3, 4, 5, 0, 6, 7, 8]

move left

printState

[1, 2, 3, 4, 5, 0, 6, 7, 8]

setState 123456b78

printState

[1, 2, 3, 4, 5, 6, 0, 7, 8]

move left

printState

[1, 2, 3, 4, 5, 6, 7, 0, 8]

move left

printState

[1, 2, 3, 4, 5, 6, 7, 8, 0]

move left

printState

[1, 2, 3, 4, 5, 6, 7, 8, 0]

**solve A-star <heuristic>**

**Base Case**

setState b12345678

solve A-Star h1

The total cost was: 0

Total moves: 0

The path was:

search A-Star h2

The total cost was: 0

Total moves: 0

The path was:

**N 10 Case**

setState 31264578b

solve A-Star h1

The total cost was: 4

Total moves: 4

The path was: left left up up

solve A-Star h2

The total cost was: 4

Total moves: 4

The path was: left left up up

**N 100 Case**

setState 51b236748

solve A-Star h1

The total cost was: 570

Total moves: 138

The path was: down left down left up right right up left left down right up right down left up left down right up right down left left up right down right up left left down right up right down left left up right down right up left left down right down left up up right right down left up left down right right up left down left up right right down left down right up up left down down right up left down right up up left down down right up left up right down down left up right up left left down right down left up up right down left down right up left up right down down left up right down left up up right down left down right up up left down right down left up up

search A-Star h2

The total cost was: 845

Total moves: 118

The path was: left left down down up down down left down left up down down up left left down down down up down up left left down down up down down left up down down left left up down down down left up left down down down left up up down down down up left left down down down up left down left down down up left down down up left up down down down left down down up up left left down down up down down left left up down down down left up left down down down down left up left up down down down left down down up up left down down down left up up left

**N 1000 Case**

setState 52b463781

solve A-Star h1

The total cost was: 751

Total moves: 172

The path was: left down left down right right up left left down right up up left down right right down left left up right down right up left up left down right right up left left down right up right down left left up right down right down left up right down left left up right right down left left up right right down left up left down right up right down left up left down right up left down right right up left left down right up right down left left up up right down left down right up up left down down right up left up right down down left up up right down left up right down left down right up right down left left up right right down left up left down right right up left down right up up left down right down left up up right down left down right up up left left down right up right down left left up right down right up left left

solve A-Star h2

The total cost was: 178

Total moves: 76

The path was: left left down down up down down left left up down down down left up left down down down up left left down down up down down down left left up up down down down left up down down left up left down down down up left down down up left left down down up left down down down up left left down down up down down left left up down down down up left left

**N 100,000 Case**

setState 528743b61

solve A-Star h1

The total cost was: 871

Total moves: 182

The path was: right right up up left left down right right down left up right down left up up left down right down left up up right down left down right up up left down right up left down right down left up up right down down left up right up left down down right up up left down right down left up up right down right up left down left up right right down left up left down right down left up right right down left left up right down right up left down left up up right down right up left left down right right up left down left up right down left down right up up left down down right up left up right right down left down right up left down right up up left left down right up right down left left up right right down left up left down right right down left up right down left up left up right right down left down right up up left left down right right down left up left up

solve A-Star h2

The total cost was: 182

Total moves: 54

The path was: up up down down down left up down down left left up down down down left left up down down down up left left down down down down up left down left up up down down down left up up down down left down down up up left down down down left up up

**solve beam <k>**

**Base Case**

setState b12345678

solve beam 3

The total cost was: 0

No Path, was goal state

**N 10 Case**

setState 31264578b

solve beam 3

Reached Goal

The total cost was: 3446

Total moves: 266

The path was: up up left left down down up left down down up down down left left up down down left up down down down down left left up up down down down left up left down down down up left up down down down up left left down down up left down down up left down down down left up up down down down left left up down down left up down down down left up left down down down up up left down down down up left left down down up left down down up left down down down left up up down down down left left up down down down left up left down down down up left up down down down up left left down down up left down down up left down down down left up up down down down left left up down down down up left left down down down down up left left up down down down down left left up up down down down left up left down down down down up up left left down down up left down down down up left left down down down up left left down down down down left left up up down down down left left up down down down left up left down down down up up left down down up left down down down left up up down down down left up left down down down up left up down down down left up up down down left up down down left up down down down up left left

**N 100 Case**

setState 51b236748

solve beam 3

Reached Goal

The total cost was: 14142

Total moves: 98

The path was: down left left up down down down left up up down down down left up left down down down up left left down down down up left up down down down left up left down down down left up up down down down left up left down down down up left up down down down left up up down down down up left left down down up left down down up left down down down up left left down down down up left left down down down up left left down down up down down left left up

**N 1000 Case**

setState 52b463781

solve beam 3

Reached Goal

The total cost was: 15269

Total moves: 106

The path was: down left left up down down down down left left up up down down down left left up down down down left left up down down down left up left down down down up left up down down down left up left down down down left up up down down down left up left down down down up left up down down down left up up down down down up left left down down up left down down up left down down down up left left down down down up left left down down down up left left down down up down down left left up

**N 100,000 Case**

setState 528743b61

solve beam 3

Reached Goal

The total cost was: 24129

Total moves: 808

(trust me on the path, there were 808 of them)

**Experiments**

**H1 vs H2**

**B/C**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 10N | H1TIME | H2TIME | H1COST | H2COST | H1PATH | H2PATH |  |
| 1 | **2.1 ms** | **0.18 ms** | **6.57 N** | **6.29 N** | **4.86 N** | **4.86 N** |  |
| 2 | **706 ms** | **237 ms** | **540 N** | **407 N** | **102 N** | **76 N** |  |
| 3 | **3.6 s** | **512 ms** | **1062 N** | **533 N** | **179 N** | **104 N** |  |
| 4 | **698 ms** | **403 ms** | **1085 N** | **487 N** | **154 N** | **98 N** |  |

**D-**

All problems were solvable as they were generated from the goal state.