Description of breadth-first implementation

In order to implement the Breath first search, three new classes were needed.

The first class that was needed was “*RunEpuzzleBFS.java*”. This class facilitated calling the search operations and linking together the other classes. It implemented such classes as “*EpuzzleSearch*” and “*EpuzzleState*” which both played a huge part in the actual running of the search algorithm.

EpuzzleSearch

One of the previously mentioned classes, “EpuzzleSearch”, was the first class to be declared inside “*RunEpuzzleBFS*”. Its extends search and its main goal was to define the goal predicate. In this implementation the goal predicate would be a solved puzzle where each number is in the correct position (The correct positions being, {{1,2,3}, {4,5,6}, {7,8,0}}). The target goal is the only parameter taken into the constructor and is the only value that could be returned from the class when “*getTarget()*” is ran.

EpuzzleState

“*EpuzzleState*” is the next class to be declared in “*RunEpuzzleBFS*”*.* The state class is the largest class out of the three. The class extends SearchState and it defines each state in the puzzle. It is what allows us to move from node to node in order to attain the path to the goal predicate. The constructor for the class takes in the current board layout in the form of a 3D integer array. This private variable can be retrieved using the getBoard() public function.

The goalPredicate() public function checks to see if the current node is the goal node. This is done simply by comparing each position to see if it matches with the position in the target board. If any position doesn’t match then false is returned.

In order to retrieve the successors the getSuccessors() function is called. This function returns an array list of SearchStates that are accessible from the current board layout. In order to do this, it first must locate the blank space as the only numbers that can move are the numbers adjacent to the blank space. Once the space is located the if statement lists every possible move, and therefore the associated EpuzzleState, based upon its position and adds them to the array list of EpuzzleStates.

When declaring the new puzzleState the new board associated with that is needed and therefore the private function moveBoardPiece is called. For parameters it takes the blank space coordinates and the value being swapped’s coordinates. It then performs a deep copy of the current board and swaps the places, returning the new board layout.

At the end of the function, the EpuzzleStates are converted to SearchStates and returned.

Running of the algorithm

Both of these classes are key in the functional operation of the breath first search and solving the puzzle. In order to start the search, a string is created to save the output and its value is set to the return value from runSearch with the parameters of the initial search state and ‘breadthFirst’. The value of the string is then printed to the console.

Description of the A\* Search Algorithm