CS260 - Data Structure
Programming Assignment 5: Sliding Puzzle
Assignment Report

Fitsum Alebachew fa496

Question 1 : 6 points

Big Picture: Draft a pictorial representation of your solution to the problem where you indicate the data structures you used. Also provide an explanation of your image. You should be answering the following questions:

(a) [2 points] What does your node structure look like? what information do you store there, why?

> **Solution:** My node is structure storing the board in an integer array and has pointers for its predecessor and different next pointers used in the queue and hash table.

(b) [2 points] How do you keep track of nodes in your program? Do you generate all possible nodes, or generate them as you visit neighboring nodes? Which data structure did you use to store the nodes? How? Explain.

> **Solution:** All the nodes generated in the program are kept in a hash table of pointers. Nodes are generated gradually as the algorithm finds the solution.

(c) [2 points] How did you keep track of which nodes to explore next? What data structure did you use? Explain the details of your implementation.

> **Solution:** The order of nodes to explore are kept in queue with a linked list implementation. The same nodes in the hash table are used to form a linked list with the queue structure pointing to head and tail of the queue.

Question 2 : 8 points

Flow of the algorithm: Starting with an initial board, provide a walk through of your algorithm in plain English (use bullet points to have an easy to follow explanation), by answering;

- how your program generates new nodes,
- in what order/why does it insert the node (or pointer of the node) into the data structures you listed above,
- how does your algorithm determine when to terminate its search?

Solution:

- The initial board is first checked to see if a solution exists. It exits if it is not solvable.
- If it is solvable, it creates a hash table structure and inserts the initial board into it.
- It Then generates next state nodes for the initial board and inserts those nodes into the hash table and queue. It then explores the different next states by using BFS.
- When the solution is found, the algorithm backtracks from the solved node and generates the sequence to solve the initial state.

Question 3 : 6 points

Space and time complexity:

(a) [2 points] What is the time complexity of your algorithm? Make an analysis by explaining your assumptions and reasoning.

Solution: If k is the length of the board, there will be $O(k^2!)$ nodes. BFS will search through these nodes in $O(k^2!)$ time as well since visited nodes will not be visited again. Hash table and queue operations are both $O(1)$.

(b) [2 points] What is the space complexity of your algorithm? Make an analysis by explaining your assumptions and reasoning.

Solution: The algorithm uses memory as it generates nodes. It will also be $O(k^2!)$.

(c) [2 points] Could you have achieved a better space and/or time complexity? If so, how?

Solution: Since both the space and time complexity depend on the number of nodes generated, if we decrease the number of visited states by using A* algorithm, we can reduce both.