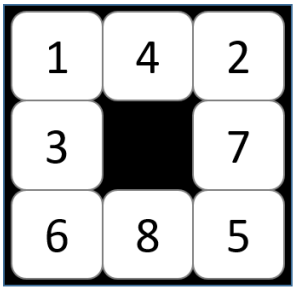
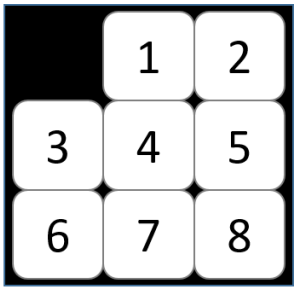
Target Topic: Breadth-First Search (BFS)

Case Study : 8-PUZZLE





INPUT: A permutation of {0, .. , 8} arranged in 3x3 format. 0 represents the hole.

OUTPUT: The sequence of moves required to reach the goal state, with the minimum number of moves

NOTE:

- BFS will not solve practically solve every test case, however, due to excessive memory usage.
- The simplest BFS only works for cases that can be solved in a few moves. For more complex cases, avoiding repeated states is essential. However, checking whether a state is repeated is a challenging coding work.

Provided materials:

- 8puzzle_bfs2_template.py : the incomplete program.
  - 8puzzle_bfs2_template_EZer.py is the easier version, for students struggling with programming
- 8-puzzle testcases.zip : 8 test cases, each has the solution included.

Task:

1. Study the provided 8puzzle_bfs2_template.py program for what are required to complete the program.

2. Complete the program so that it solves the 8-puzzle problem with BFS technique.

   - The first step is probably to find the minimum number of moves to solve.
   - Once the above step is accomplished, then attempt to make the program compute the optimal sequence of moves.

3. Test the program with the provided test cases. Note the limitation of the program.

4. [Challenging] Modify the program so that the BFS does not repeat the search on the state that has been visited already.
   Hint: Utilize set() and tuple()