1 Search Implementation

Implement the generic graph search algorithm from figure 3.7. Store the frontier as:

1. A LIFO
2. A FIFO
3. A priority queue, prioritized on f values (thereby implementing A*)

Test your code on the 8 puzzle. Compare the number nodes expanded for each of the above methods of storing the frontier and 3 different starting configurations of your choosing. For your A* implementation, use the Manhattan ($L_1$) distance as your heuristic. Be sure to pick a starting configuration from which the goal state is actually reachable. You can ensure this by starting from the solved game and making a few legal moves to generate your starting position.

Turn in your code, your initial board configurations, and a summary of the nodes expanded in each case.

Implementation notes: You are free to use any reasonable programming language you want. You are encouraged to use built-in libraries for things like the priority queue. (If you use an external library that is not part of the standard distribution, you should cite your source.) A quick and easy way to implement the explored set is by using a hash table. The actual implementation of the search algorithm must be your own.

2 Search with negative path costs

Do question 3.8 from the text.

3 Iterative Deepening

Do question 3.18. Also, explain why this result is not inconsistent with what we proved in class about IDDFS.

4 Consistent Heuristics

Do question 3.29 from the text.

5 Alternate Search Methods

Do problem 4.1 from the text.