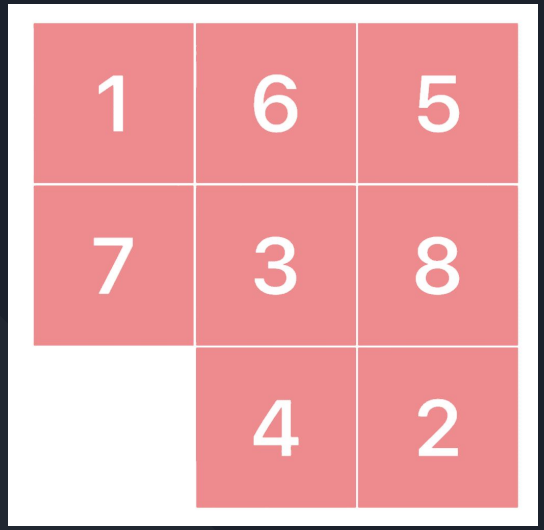
COMP 472

Assignment 2



Team members

Adrian Patterson *40048841*

Khalil Nijaoui *40092653*

Michel Abboud *40025378*

Karl-Joey Chami *27736657*

Depth-First Search

- Searches a graph data structure
  - Starts at root state
  - Explores the children states to the deepest level using stack implementation
    - Backtracks to visit all possible children states
- Guarantees a solution that is not optimal with high execution time
- If any of the children nodes are the goal state the algorithm will take high execution time before finding it
- The algorithm might have to try all 10! (in the case of a 3x3 puzzle) possible states before finding the goal state
- DFS doesn't have any logic to determine how far the current state is from the goal state

Iterative Deepening

- Improvement over Depth-first Search
- This algorithm recursively runs depth first search on the graph with an increasing maximum depth until the goal is found.
- More optimal than Depth-first Search alone, it also requires less execution time to find the goal.

Hamming Distance

- Hamming Distance Search: Our first Heuristic Search
  - A heuristic search provides an algorithm a metric by which to decide on which path to take
  - In other words, gives the algorithm a sense of “correctness”
- How does it work?
  - Hamming compares differences between two arrays
  - For every distance, the distance is incremented by one
- When a state is **expanded,** each resulting state has its hamming distance computed
    - These are then sorted to allow the state with smallest distance to be expanded first
- When does it fail?
  - Hamming may return a low distance for an array where there are little differences, but the nodes are still far from where they need to be

Manhattan Distance

- This is a logic based algorithm, will calculate the sum of the displacements that exists between a node's position and it's goal position.
- Based on this, it will provide the best next state to explore and keep on repeating until goal state is found.
- This has proven to be one of the most optimal algorithms to solve an n*n puzzle

Analysis

Analysis of 20 3x3 puzzles				
Metric	Depth-first Search	Iterative Deepening	Hamming Distance	Manhattan Distance
Total solution length	4707	650370	739	261
Average solution length	4707	325185	36.95	13.05
Total solution cost	112968	15608739	17736	6264
Average solution cost	112968	7804369	886.80	313.20
Average execution time	45.10 seconds	22.04 seconds	0.32 seconds	0.01 seconds
Number of unsolved puzzles	19	18	0	0
Percentage of unsolved puzzles	95%	90%	0%	0%

Scaled-Up Experiment Results

Analysis of 20 4x4 puzzles				
Metric	Depth-first Search	Iterative Deepening	Hamming Distance	Manhattan Distance
Total solution length	0	0	547	2970
Average solution length	0	0	182.33	174.71
Total solution cost	0	0	26256	142560
Average solution cost	0	0	8752	8385.88
Average execution time	0 seconds	0 seconds	17.14 seconds	9.58 seconds
Number of unsolved puzzles	20	20	17	3
Percentage of unsolved puzzles	100%	100%	85%	15%