MATH 327 Homework 2

Stephen Hung

1. Item 1: DFS Performance

   Length of solution path found: 40 edges
   40 states expanded.
   MAX_OPEN_LENGTH = 7

   The length of the solution path found is the amount of operations required for the algorithm to go from the initial starting state to the goal state. This is important when comparing search algorithms because it demonstrates which algorithm can find the most efficient solution path. In addition, the length of the solution path may come at the cost of memory.

   The states expanded represents the number of times states had an operation performed to it. This is important when comparing search algorithm because it demonstrates which algorithm is the most efficient for traversing the graph and memory usage.

   MAX_OPEN_LENGTH represents the maximum amount of states that were stored at one time to be checked. This is important when comparing search algorithms because it demonstrates the amount of memory an algorithm needs for the OPEN list while iterating.

2. Item 2: BreadthFS Performance

   Length of solution path found: 15 edges
   70 states expanded.
   MAX_OPEN_LENGTH = 16

3. Item 3: Iterative-Deepening DFS

   Length of solution path found: 15 edges
   475 States
   MAX_OPEN_LENGTH = 7

4. Item 4: Alternative Search Methods for the Towers of Hanoi

Towers of Hanoi, 4 disks

Algorithm Name	Length of Solution Path	Number of Nodes Expanded	MAX OPEN LENGTH
Iterative DFS	40 edges	40 Nodes Expanded	7
Breadth-First Search	15 edges	70 Nodes Expanded	16
IDDFS	15 edges	475 states	7

5. Item 5: Blind Search on My A2 Problem Formulations

Farmer, Fox, etc.

Algorithm Name	Length of Solution Path	Number of Nodes Expanded	MAX OPEN LENGTH
Iterative DFS	5 edges	6 Nodes Expanded	3
Breadth-First Search	5 edges	10 Nodes Expanded	5
IDDFS	5 edges	20 states	3

Find The Number

Algorithm Name	Length of Solution Path	Number of Nodes Expanded	MAX OPEN LENGTH
Iterative DFS	8 edges	12 Nodes Expanded	14
Breadth-First Search	4 edges	90 Nodes Expanded	89
IDDFS	4 edges	34 states	10

6. Item 8: Heuristics for the Eight Puzzle

Eight Puzzle With Heuristics (puzzle10a)

Puzzle Instance Name	Puzzle Instance Permutation	Success (Yes/No)	Count of Expanded Nodes	Aborted (Yes/No)
puzzle10a.py	h_euclidian	Yes	15	No
puzzle10a.py	h_hamming	Yes	43	No
puzzle10a.py	h_manhattan	Yes	12	No
puzzle10a.py	h_custom	Yes	21	No

Eight Puzzle With Heuristics (puzzle12a)

Puzzle Instance Name	Puzzle Instance Permutation	Success (Yes/No)	Count of Expanded Nodes	Aborted (Yes/No)
puzzle12a.py	h_euclidian	Yes	38	No
puzzle12a.py	h_hamming	Yes	90	No
puzzle12a.py	h_manhattan	Yes	20	No
puzzle12a.py	h_custom	Yes	42	No

Eight Puzzle With Heuristics (puzzle14a)

Puzzle Instance Name	Puzzle Instance Permutation	Success (Yes/No)	Count of Expanded Nodes	Aborted (Yes/No)
puzzle14a.py	h_euclidian	Yes	50	No
puzzle14a.py	h_hamming	Yes	164	No
puzzle14a.py	h_manhattan	Yes	31	No
puzzle14a.py	h_custom	Yes	68	No

Eight Puzzle With Heuristics (puzzle16a)

Puzzle Instance Name	Puzzle Instance Permutation	Success (Yes/No)	Count of Expanded Nodes	Aborted (Yes/No)
puzzle16a.py	h_euclidian	Yes	165	No
puzzle16a.py	h_hamming	Yes	547	No
puzzle16a.py	h_manhattan	Yes	115	No
puzzle16a.py	h_custom	Yes	217	No

7. 9 Item 9: Evaluating my Custom Heuristic.

   (a) First explain how your heuristic works, and the thinking behind it.

   My heuristic takes the average of manhattan, hamming, and euclidian heuristics. However, I also multiply manhattan by 0.1 and euclidian by 2. My reasoning behind this is that when testing puzzle10a, i found out that manhattan expands too many states while euclidian produces the least amount and hamming produces an average amount of states. Furtherore, each of the heuristic functions are admissible and thus by taking the average, my custom heuristic should also be admissible. Thus, I decided to reduce the weight that manhattan had on my heuristic and increase the weight of the euclidian heuristic.

   However, as I found out, in other problems besides puzzle10a such as puzzle12a, puzzle14a, and puzzle16a, manhattan ended up being a better heuristic than euclidian or hamming.

   Let m = Manhattan Heuristic
   Let h = Hamming Heuristic
   Let e = Euclidian Heuristic

$$\text{h\_custom} = \frac{(m \cdot 0.1)+(h)+(e \cdot 2)}{3}$$

(b) Second tell whether it actually outperforms any of the other heuristics in terms of lowering the number of expanded states while still solving the problem.

My Custom Heuristic does not outperform all other heuristics. Instead, it outperforms some heuristics while performing less than others. For instance in puzzle10a, custom expands 21 states while hamming expands 43. Thus, custom outperforms hamming. However, euclidian expands 15 and manhattan expands 12 thus custom is outperformed.

(c) Finally, discuss how you believe the computational cost of computing your heuristic compares with the cost of computing the others.

I believe the computational cost of computing my custom heuristic is larger than the others. This is because I calculate the average of all other heuristics and thus call all the other functions. This means, that my computational cost is approximately all of the other heuristics' computational cost combined.