Name: VISHAL BHARTI Banner ID : 000132619

C.Sc. 74011 Artificial Intelligence

Assignment 1

Table listing the maximum nodes expanded and time taken by each search technique:

	A* with misplaced tile heuristics	A* with Manhattan distance heuristics	DFS Branch and Bound #	Iterative Deepening A*
Number of Nodes Expanded	Easy : 7 Medium : 32 Hard : 82 Worst: 108262	Easy : 6 Medium : 16 Hard : 28 Worst: 1329	Easy : 12 Medium : 9336 Hard : 9312 Worst: 20706	Easy : 9 Medium : 23 Hard : 132 Worst: 8998
Time Taken (in ms)	Easy : 7 Medium : 34 Hard : 91 Worst: 512072	Easy : 6 Medium : 13 Hard : 32 Worst: 1647	Easy : 6 (11) Medium : 10225 (10232) Hard : 10190 (10198) Worst: 22665 (22772)	Easy : 7 Medium : 18 Hard : 106 Worst: 6743

- The second time inside the parenthesis gives the time at which the algorithm terminates

The A* with Manhattan Distance heuristics was the fastest algorithm, but its space complexity was higher than IDA*. The IDA* beat the A* algorithm with the misplaced tiles heuristics in terms of both time and space complexity. The DFSBNB performs much better than the A* (with misplaced tile heuristics) in the worst case.

Optimal Sequence of moves for each board (from left to right):

Easy:

1 3 4 8 6 2 7 0 5	1 3 4 8 0 2 7 6 5	1 3 4 8 2 0 7 6 5	1 3 0 8 2 4 7 6 5	1 0 3 8 2 4 7 6 5	1 2 3 8 0 4 7 6 5

Medium:

2 8 1 0 4 3 7 6 5	0 8 1 2 4 3 7 6 5	8 0 1 2 4 3 7 6 5	8 1 0 2 4 3 7 6 5	8 1 3 2 4 0 7 6 5	8 1 3 2 0 4 7 6 5	8 1 3 0 2 4 7 6 5	0 1 3 8 2 4 7 6 5	1 0 3 8 2 4 7 6 5	1 2 3 8 0 4 7 6 5

Hard:

2 8 1 4 6 3 0 7 5	2 8 1 4 6 3 7 0 5	2 8 1 4 0 3 7 6 5	2 8 1 0 4 3 7 6 5	0 8 1 2 4 3 7 6 5	8 0 1 2 4 3 7 6 5	8 1 0 2 4 3 7 6 5	8 1 3 2 4 0 7 6 5	8 1 3 2 0 4 7 6 5	8 1 3 0 2 4 7 6 5	0 1 3 8 2 4 7 6 5	1 0 3 8 2 4 7 6 5	1 2 3 8 0 4 7 6 5

Worst:

5 6 7 4 0 8 3 2 1	5 6 7 4 2 8 3 0 1	5 6 7 4 2 8 3 1 0	5 6 7 4 2 0 3 1 8	5 6 0 4 2 7 3 1 8	5 0 6 4 2 7 3 1 8	0 5 6 4 2 7 3 1 8	4 5 6 0 2 7 3 1 8	4 5 6 3 2 7 0 1 8	4 5 6 3 2 7 1 0 8
4 5 6 3 2 7 1 8 0	4 5 6 3 2 0 1 8 7	4 5 0 3 2 6 1 8 7	4 0 5 3 2 6 1 8 7	0 4 5 3 2 6 1 8 7	3 4 5 0 2 6 1 8 7	3 4 5 1 2 6 0 8 7	3 4 5 1 2 6 8 0 7	3 4 5 1 2 6 8 7 0	3 4 5 1 2 0 8 7 6
3 4 0 1 2 5 8 7 6	3 0 4 1 2 5 8 7 6	0 3 4 1 2 5 8 7 6	1 3 4 0 2 5 8 7 6	1 3 4 8 2 5 0 7 6	1 3 4 8 2 5 7 0 6	1 3 4 8 2 5 7 6 0	1 3 4 8 2 0 7 6 5	1 3 0 8 2 4 7 6 5	1 0 3 8 2 4 7 6 5
1 2 3 8 0 4 7 6 5									

Questions:

1. What is the number of possible states of the board?

Ans: The total number of possible states of the board is given by 9! = 9*8*7*6*….*1. However, only half of these are solvable i.e., there are 9!/2 solvable states on the board.

2. What is the average number of possible moves from a given position of the board?

Ans: For each state the total possible moves are the possible next stages in the solution tree. For each state the number of possible move differ based on the position of the blank tile. Therefore, the average number of moves from a given possible position on the board would be the average branching factor of the solution tree.

3. Estimate how many moves might be required for an optimal (minimum number of moves) solution to a "worst-case" problem (maximum distance between starting and goal states). Explain how you made your estimate (Note this is an open-ended question; any logical answer may suffice).

Ans: The minimum number of moves for an optimal solution will be the depth of the solution as for an optimal solution will find the solution by going through the best sequence of nodes without expanding any other nodes and hence will have cost equal to the depth.

4. Assuming the answer to question #2 is the "average branching factor" and a depth as in the answer to question #3, estimate the number of nodes that would have to be examined to find an answer by the brute force breadth-first search.

Ans: The total number of nodes which we need to visit to solve the problem by brute force approach is b^d, where b is the branching factor and d is the depth of the solution.

5. Assuming that your computer can examine one move per millisecond, would such a blind-search solution to the problem terminate before the end of the semester?

Ans: Suppose we have an average branching factor of 3 and a depth of 30 as in our worst case. Then we need to examine 3^{30} nodes to get to the result by brute force approach. If our processor can process 1 move per millisecond, then the total time required is:

205891132094649/(1000*3600*365) = 156690 years.

And by no means before the end of this semester.

6. The "worst" example problem given above is actually one of the easiest for humans to solve. Why do you think that is the case? What lessons for AI programs are suggested by this difference between human performance and performance of your search program?

Ans: A problem solving algorithm however smart it can just predict if the current stage is the goal or not and it needs to transition from one stage to another continuously to reach a solution. The problem is that the moves the algorithm chooses are either brute force or based on some heuristics, which make some assumptions. Humans on the other hand can easily predict the outcome of a move and its consequent moves far more accurately than a computer. We use our guesswork and intuitions together, which help us solve a problem faster. On the other hand, a computer makes decisions using heuristics which by no means are comparable to the efficiency of human decision making power.

7. Compare A*, DFBnB, and IDA* and discuss their advantages and disadvantages.

Ans: The A* algorithm uses a heuristic function to predict the cost of a move and chooses the move that has the least cost among all alternatives. The advantage of A* is that we can quickly reach a solution, however the A* algorithm has an exponential space complexity. Another important aspect of the A* algorithm is the choice of heuristic function. In our case the misplaced

tile heuristics and the Manhattan block heuristics, had a large difference in performance for the worst case, with the Manhattan block heuristic totally outperforming the misplaced tile heuristics.

The DFBNB uses the DFS algorithm to find a solution but once it achieves the solution it does not stop continue searching the remaining nodes and only expands if cost associated with the node is lower than the previous cost. It has a better space complexity as compared to A* but it suffers from an exponentially high time complexity. However, for our case the DFS BNB has better time complexity as compared to the A* algorithm (with misplaced tile heuristics).

The IDA* uses the space efficiency of DFS and the quickness of the A*. It reaches a solution almost as quickly as A* and at the same time has a better space complexity. The IDA* however sometimes may be slower as it goes through the same set of nodes again and again, as there is no check for repetitions.