For the Given Input i.e. 673842105, we are not able to reach the goal using any of the heuristic function.

The reason is because this input can not be solved by any of the algorithm.  
Below is the table of comparison of above-mentioned matrix.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Heuristic function** | **Number of states explored** | **Total number of states on optimal path** | **Optimal Path** | **Optimal Cost of the Path** | **Total time taken for execution (hh:mm:ss.ms)** |
| h(1) | 181440 | Not able to solve | Not able to solve | Not able to solve | 0:12:02.912468 |
| h(2) | 181440 | Not able to solve | Not able to solve | Not able to solve | 0:12:05.033437 |
| h(3) | 181440 | Not able to solve | Not able to solve | Not able to solve | 0:12:33.553659 |
| h(4) | 181440 | Not able to solve | Not able to solve | Not able to solve | 0:14:09.219398 |

But if we take another input matrix i.e. 538124760, then we are able to reach the goal. Below is the comparison of each heuristic function.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Heuristic function** | **Number of states explored** | **Total number of states on optimal path** | **Optimal Path** | **Optimal Cost of the Path** | **Total time taken for execution**  **(hh:mm:ss.ms)** |
| h(1) | 26243 | 18 | Printed in the Jupiter notebook | g(n): 171  f(n): 171 | 0:00:12.995637 |
| h(2) | 1202 | 18 | Printed in the Jupiter notebook | g(n): 171  f(n): 264 | 0:00:00.137786 |
| h(3) | 339 | 18 | Printed in the Jupiter notebook | g(n): 171  f(n): 319 | 0:00:00.082441 |
| h(4) | 521 | 18 | Printed in the Jupiter notebook | g(n): 171  f(n):289 | 0:00:00.134083 |

Keeping the above results in mind we can compare each heuristic function in bellow manner.

1. **h1(n) = 0:**   
   This heuristic always returns 0. It is an admissible but not informative heuristic.  
   Admissible means that it will never overestimates the cost to reach the goal, which is a crucial property for A\* to guarantee optimality.  
   It's not informative because it provides no guidance for the search as the heuristic is 0 for all the states.  
   As a result, A\* using h1 will behave like uniform cost search (UCS).  
   As it goes not guide so it may take a long time to reach the goal for complex puzzles.  
   Time Complexity: Constant (O(1))
2. **h2(n)= number of tiles displaced from their destined position.**  
   By using this heuristic function we count the number of tiles that are not in their goal positions.  
   It is admissible because it never overestimates the cost. But it may underestimate the true cost, especially when multiple tiles need to be moved to their goal positions in a single move.  
   This heuristic is not optimal because this consider the number of tiles misplaced. Many times, we may need to move more times (which result in more misplaced cost) to reach the goal state.  
   But this is better than h1(n) as as it involves counting tiles. We can also say that it can speed up the search compared to h1.  
   Time Complexity: Linear (O(N))
3. **h3(n) = sum of the Manhattan distance of each tile from the goal position.**  
   The Manhattan distance heuristic adds the distances of each tile is away from its goal position in terms of the number of moves required.   
   It is admissible and more informative than h1(n) and h2(n).   
   It often provides a better estimate of the true cost, guiding the search more effectively towards the goal as compared to h(1) and h(2).  
   h3 could be better solution because it makes a good balance between informativeness and efficiency.  
   From results also we can see that it has find the result much faster and by exploring less number of states as compared to other heuristic function.  
   Time Complexity: Linear (O(N))
4. **h4(n) = sum of the Euclidean distance of each tile from the goal position**  
   The Euclidean distance heuristic uses the straight-line distance between each tile's current position and its goal position.  
   It is admissible but there is very high possibility that it will overestimate the cost in some cases because it assumes tiles can move directly to their goal positions but in real we have to use the blank tile to move around.  
   This is less efficient as compared to h3 as we can see that it has explored more number of states and has taken more time to reach the goal.  
   Time Complexity: Linear (O(N))

So, based on these observations, we can say that the h3(n) heuristic function is better than others. It has taken very less time to reach the goal. It has also explored very less number of states.