**Experiment No. : 8**

**Title: 15 puzzle problem using Branch and bound**

**Batch: B2 Roll No.: 16010421119 Experiment No.: 8**

**Aim:** To Implement 8/15 puzzle problem using Branch and bound.

**Algorithm of 15 puzzle problem using Branch and bound:**

**The 15 puzzle problem is a popular sliding puzzle game where you have to slide tiles on a board to arrange them in a particular order. The game is played on a 4x4 board with 15 tiles numbered from 1 to 15 and a blank tile.**

**The Branch and Bound algorithm is an optimization technique that is used to solve combinatorial optimization problems. The idea behind this algorithm is to explore the search space in a systematic way, while using a lower bound to prune branches that will not lead to the optimal solution.**

**Here is the algorithm of 15 puzzle problem using Branch and bound:**

1. **Start with an initial state of the puzzle.**
2. **Compute the heuristic value of the initial state. This heuristic value will be used as a lower bound.**
3. **Add the initial state to the priority queue.**
4. **While the priority queue is not empty, do the following:**
5. **Remove the state with the lowest f-value (f-value = g-value + heuristic value) from the priority queue.**

**b. If the state is the goal state, return the solution.**

**c. Generate all possible successors of the state.**

**d. For each successor, compute its g-value (number of moves to reach the state) and its heuristic value.**

**e. If the g-value + heuristic value is less than the current best solution, add the successor to the priority queue.**

**f. Repeat steps a-f until a solution is found or the priority queue is empty.**

1. **If the priority queue is empty and no solution is found, return failure.**

**In this algorithm, the priority queue is used to store the states of the puzzle. The f-value is used to prioritize the states in the priority queue. The g-value represents the number of moves required to reach the current state, and the heuristic value is a lower bound on the number of moves required to reach the goal state.**

**The key idea of this algorithm is to explore the search space in a systematic way while using the lower bound to prune branches that will not lead to the optimal solution. By doing this, we can avoid exploring the entire search space and find the optimal solution more efficiently.**

**Problem Statement**

Find the following 15 puzzle problem using branch and bound technique and show each steps in detail using state space tree.



**Also verify your answer by simulating steps of same question on following link.**

[**http://www.sfu.ca/~jtmulhol/math302/puzzles-15.html**](http://www.sfu.ca/~jtmulhol/math302/puzzles-15.html)

**Solution**

**Derivation of 15 puzzle problem using Branch and bound:**

Time complexity Analysis

**Program(s) of 15 puzzle problem using Branch and bound:**

**Output(o) of 15 puzzle problem using Branch and bound:**

**Post Lab Questions:-** Explain how to solve the Knapsack problem using branch and bound.

**Conclusion: (Based on the observations):**

**Outcome:**

**References:**

1. Richard E. Neapolitan, " Foundation of Algorithms ", 5th Edition 2016, Jones & Bartlett Students Edition
2. Harsh Bhasin , " Algorithms : Design & Analysis", 1st Edition 2013, Oxford Higher education, India
3. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
4. Jon Kleinberg, Eva Tardos, " Algorithm Design", 10th Edition 2013, Pearson India Education Services Pvt. Ltd.