

|  |  |
| --- | --- |
| **Name** | Vedant Deshmukh |
| **UID No.** | 2021300025 |
| **Class** | COMPS A (B batch) |
| **Experiment No.** | 8 |

**Aim:** To use Branch and bound to solve the 15 Puzzle problem

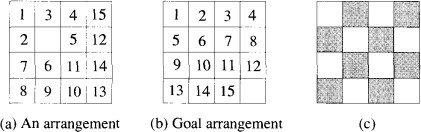
# Theory:

* **Branch and Bound Algorithm:**

Branch and bound is a general algorithmic technique used in optimization problems where an exhaustive search is infeasible. The technique divides the search space into smaller subspaces that can be explored efficiently. This method optimizes the search by eliminating partial solutions that cannot possibly be completed to a better solution than the current best solution found so far.

# 15 Puzzle Problem

The 15 puzzle problem is a well-known sliding puzzle game. The puzzle is played on a 4x4 grid that contains 15 numbered tiles and one empty tile. The goal of the game is to rearrange the tiles to get them in order from 1 to 15, with the empty space in the bottom right corner, in the least number of moves.





# Algorithm:

Solve15Puzzle(arr)

*move\_count = 0*

*misplaced\_count = misplaced(arr) cost = 0*

*prev\_move = ' '*

*while (misplaced\_count != 0) move\_count++*

*cost\_up = ∞ cost\_down = ∞ cost\_left = ∞ cost\_right = ∞*

*if (prev\_move != 'd' AND can\_move(arr, 'u')) move(arr, 'u')*

*cost\_up = misplaced(arr) + move\_count move(arr, 'd')*

*if (prev\_move != 'u' AND can\_move(arr, 'd')) move(arr, 'd')*

*cost\_down = misplaced(arr) + move\_count move(arr, 'u')*

*if (prev\_move != 'r' AND can\_move(arr, 'l')) move(arr, 'l')*

*cost\_left = misplaced(arr) + move\_count move(arr, 'r')*

*if (prev\_move != 'l' AND can\_move(arr, 'r')) move(arr, 'r')*

*cost\_right = misplaced(arr) + move\_count move(arr, 'l')*

*if (cost\_up < cost\_down AND cost\_up < cost\_left AND cost\_up < cost\_right AND can\_move(arr, 'u'))*



*move(arr, 'u') prev\_move = 'u' cost = cost\_up*

*else if (cost\_down < cost\_left AND cost\_down < cost\_right AND can\_move(arr, 'd'))*

*move(arr, 'd') prev\_move = 'd' cost = cost\_down*

*else if (cost\_left < cost\_right AND can\_move(arr, 'l')) move(arr, 'l')*

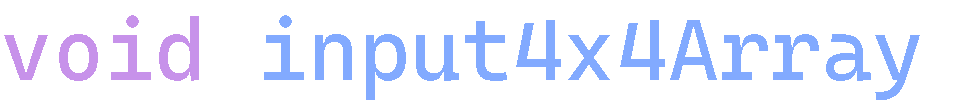
*prev\_move = 'l' cost = cost\_left*

*else*

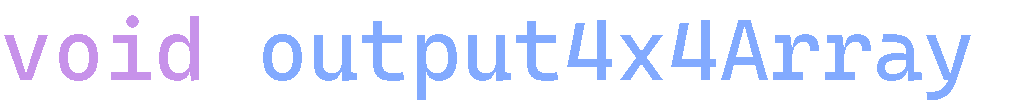
*move(arr, 'r') prev\_move = 'r' cost = cost\_right*

*misplaced\_count = misplaced(arr) return arr*

# Code:







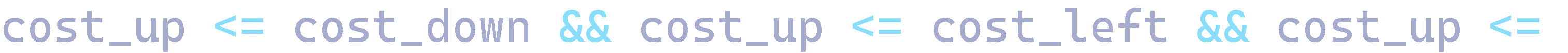










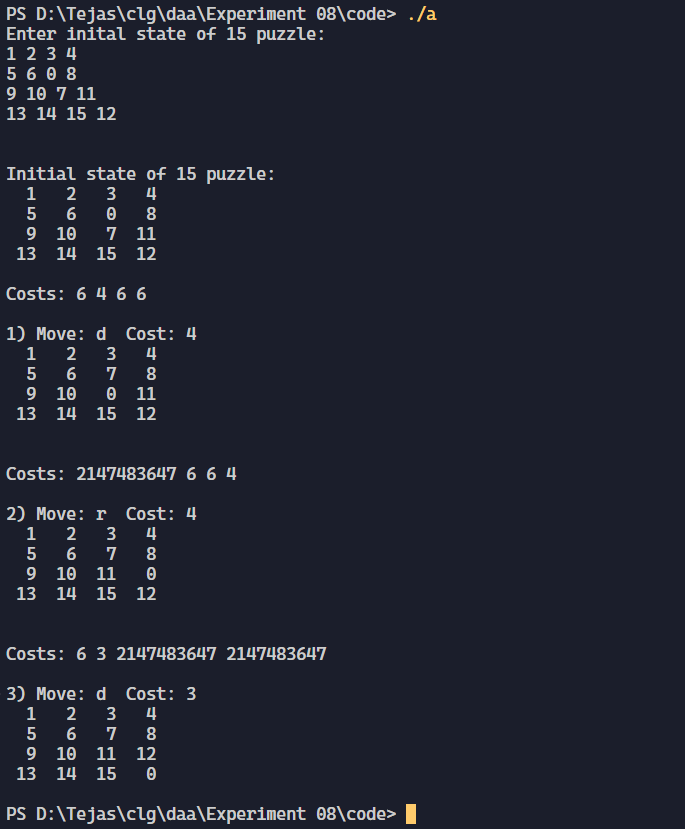






**Output:**







# Conclusion:

In conclusion, the branch and bound algorithm is an effective technique for solving optimization problems, including the 15 puzzle problem. By dividing the search space into smaller subspaces and eliminating partial solutions that cannot lead to a better solution, this algorithm reduces the computational complexity of the problem, resulting in a more efficient and optimal solution.