Experiment No. 11
15 puzzle problem
Date of Performance:
Date of Submission:

Experiment No. 11

Title: 15 Puzzle

Aim: To study and implement 15 puzzle problem

Objective: To introduce Backtracking and Branch-Bound methods

Theory:

The 15 puzzle problem is invented by Sam Loyd in 1878.

- In this problem there are 15 tiles, which are numbered from 0 – 15.
- The objective of this problem is to transform the arrangement of tiles from initial arrangement to a goal arrangement.
- The initial and goal arrangement is shown by following figure.

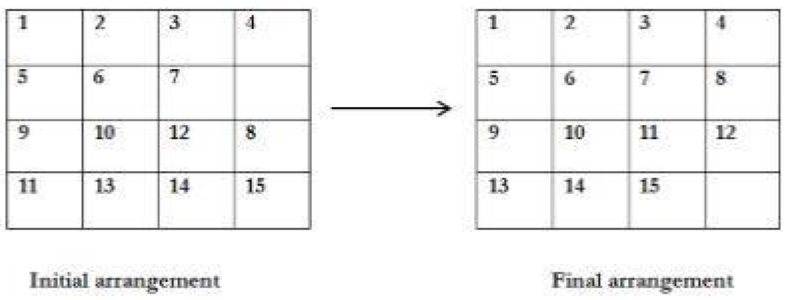
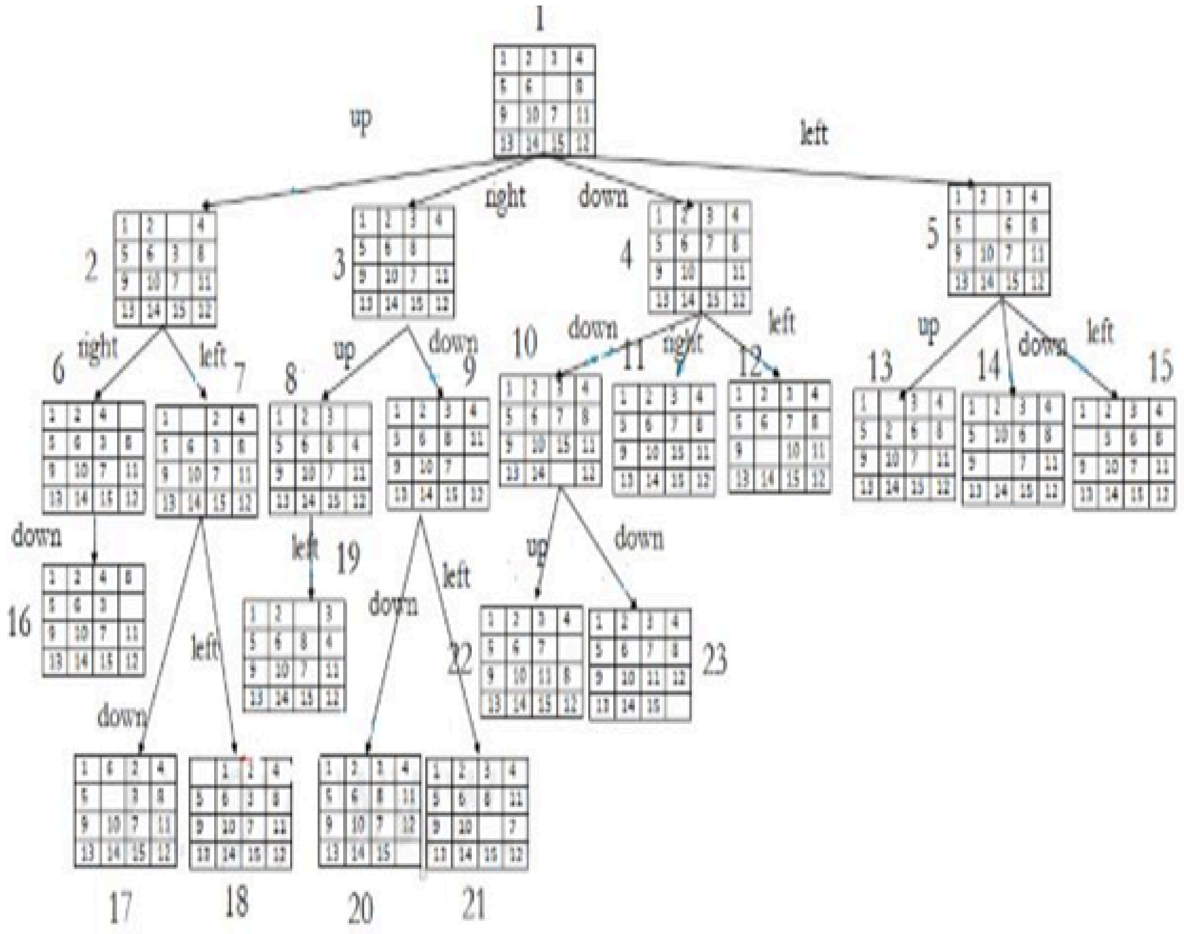


Figure 12

- There is always an empty slot in the initial arrangement.
- The legal moves are the moves in which the tiles adjacent to ES are moved to either left, right, up or down.
- Each move creates a new arrangement in a tile.
- These arrangements are called as states of the puzzle.
- The initial arrangement is called as initial state and goal arrangement is called as goal state.
- The state space tree for 15 puzzle is very large because there can be 16! Different arrangements.
- A partial state space tree can be shown in figure.
- In state space tree, the nodes are numbered as per the level.

- Each next move is generated based on empty slot positions.
- Edges are label according to the direction in which the empty space moves.
- The root node becomes the E – node.
- The child node 2, 3, 4 and 5 of this E – node get generated.
- Out of which node 4 becomes an E – node. For this node the live nodes 10, 11, 12 gets generated.
- Then the node 10 becomes the E – node for which the child nodes 22 and 23 gets generated.
- Finally we get a goal state at node 23.
- We can decide which node to become an E – node based on estimation formula.

Example:

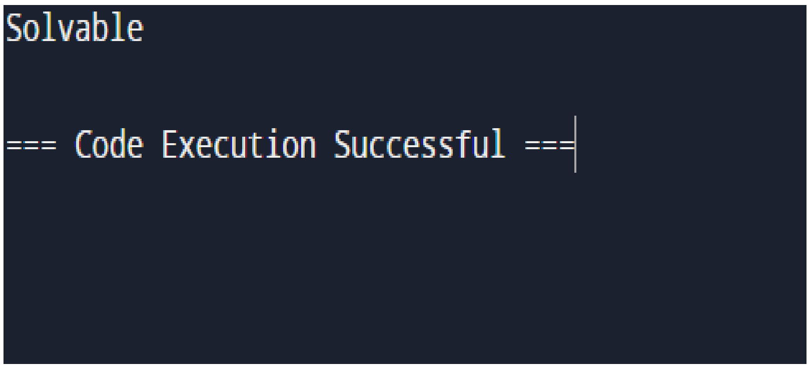


Implementation:

```
#include <iostream>
#define N 4
using namespace std;
int getInvCount(int arr[])
{
	int inv_count = 0;
	for (int i = 0; i < N * N - 1; i++)
	{
		for (int j = i + 1; j < N * N; j++)
		{
			if (arr[j] && arr[i] && arr[i] > arr[j])
				inv_count++;
		}}
	return inv_count;
}
int findXPosition(int puzzle[N][N])
{	for (int i = N - 1; i >= 0; i--)
		for (int j = N - 1; j >= 0; j--)
			if (puzzle[i][j] == 0)
				return N - i;
}
bool isSolvable(int puzzle[N][N])
{
	int invCount = getInvCount((int*)puzzle);
	if (N & 1)
		return !(invCount & 1);

	else		{
```

```
            int pos = findXPosition(puzzle);
            if (pos & 1)
                    return !(invCount & 1);
            else
                    return invCount & 1;
        }}
int main()
{

        int puzzle[N][N] =
        {
                {12, 1, 10, 2},
                {7, 11, 4, 14},
                {5, 0, 9, 15}, // Value 0 is used for empty space
                {8, 13, 6, 3},
        };
        /*
        int puzzle[N][N] = {{1, 8, 2},
                                {0, 4, 3},
                                {7, 6, 5}};

        int puzzle[N][N] = {
                                {13, 2, 10, 3},
                                {1, 12, 8, 4},
                                {5, 0, 9, 6},
                                {15, 14, 11, 7},
                        };

        int puzzle[N][N] = {
                                {6, 13, 7, 10},
                                {8, 9, 11, 0},
                                {15, 2, 12, 5},
                                {14, 3, 1, 4},
                        };



        int puzzle[N][N] = {
                                {3, 9, 1, 15},
                                {14, 11, 4, 6},
                                {13, 0, 10, 12},
                                {2, 7, 8, 5},
                        };
        */

        isSolvable(puzzle)? cout << "Solvable":
        cout << "Not Solvable";
        return 0;
}
```

Output:



```
Solvable

=== Code Execution Successful ===
```

Conclusion: The Subset Sum problem is a fundamental computational problem with various algorithmic approaches for solving it. While brute force methods can work for small input sizes, dynamic programming and other optimized algorithms offer more efficient solutions for larger instances of the problem.