Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Information Technology Engineering Department

Academic Year: 2022-2023

Class: TE **Sem.:** V **Course:** Artificial Intelligence/Machine Learning

2020400048	Karan Shah

Experiment 2:

PROBLEM SOLVING USING UNINFORMED SEARCH STRATEGY

Aim:

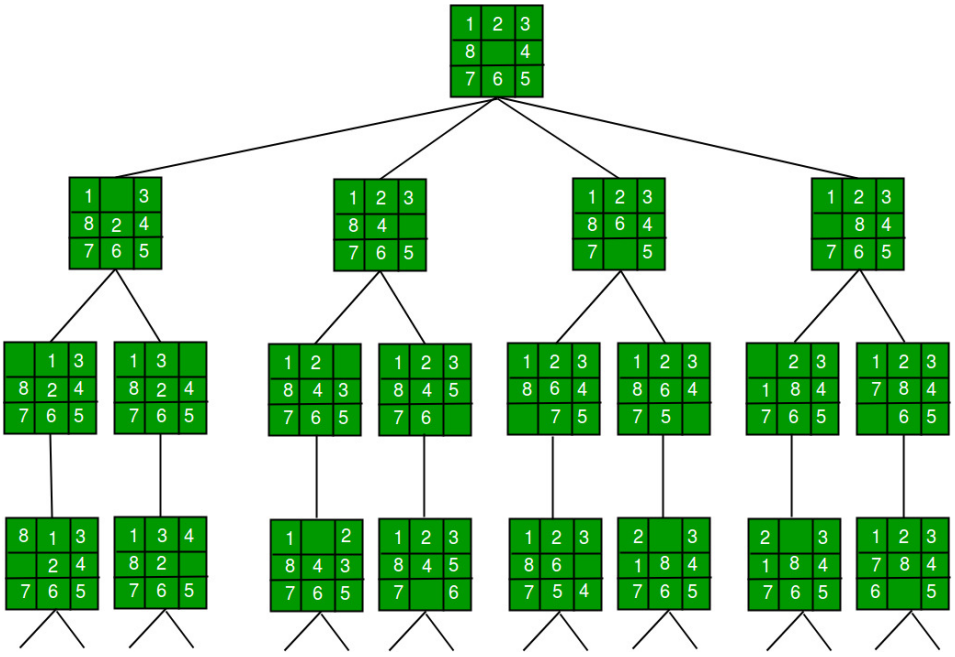
To solve the 8-puzzle problem using the DFS(uninformed search) algorithm.

Introduction:

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space.

Depth First Search :

We can perform a depth-first search on state-space (Set of all configurations of a given problem i.e. all states that can be reached from the initial state) tree.



State Space Tree for 8 Puzzle

In this solution, successive moves can take us away from the goal rather than bringing us closer. The search of a state-space tree follows the leftmost path from the root regardless of the initial state. An answer node may never be found in this approach.

2. BFS (Brute-Force)

We can perform a Breadth-first search on the state space tree. This always finds a goal state nearest to the root. But no matter what the initial state is, the algorithm attempts the same sequence of moves like DFS.

3. Branch and Bound

The search for an answer node can often be speeded by using an "intelligent" ranking function, also called an approximate cost function to avoid searching in sub-trees that do not contain an answer node. It is similar to the backtracking technique but uses a BFS-like search.

Program:

```
#include <bits/stdc++.h>

using namespace std;

#define N 3


bool isReached(vector<vector<int>> initial, vector<vector<int>> final)

{

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            if (initial[i][j] != final[i][j])

            {

                return 0;

            }
```

```
        }
    }
    return 1;
}

string converter(vector<vector<int>> Mat)
{
    string ans = "";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            ans += Mat[i][j];
        }
    }
    return ans;
}

unordered_map<string, bool> visited(0);

void printMat(vector<vector<int>> arr)
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
```

Information Technology Engineering Department

Academic Year: 2022-2023

Class: TE **Sem.:** V **Course:** Artificial Intelligence/Machine Learning

```
        {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}

vector<int> findZero(vector<vector<int>> Mat)
{
    vector<int> ans;
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            if (Mat[i][j] == 0)
            {
                ans.push_back(i);
                ans.push_back(j);
                return ans;
            }
        }
    }
    cout << "Give some sensible input" << endl;
    return ans;
```

```
}

int counter = 0;

void solve(vector<vector<int>> initial, vector<vector<int>> final)

{

    if (isReached(initial, final))

    {

        cout << "Reached" << endl;

        printMat(initial);

        exit(0);

    }

    if (visited[converter(initial)])

    {

        cout << "Go back" << endl;

    }

    else

    {

        visited[converter(initial)] = 1;

        vector<int> posZero = findZero(initial);

        int x = posZero[0];

        int y = posZero[1];

        // Up

        if (x - 1 >= 0)

        {

            swap(initial[x][y], initial[x - 1][y]);

            if (!visited[converter(initial)])
```

```
        {
            printMat(initial);
            solve(initial, final);
        }
        swap(initial[x][y], initial[x - 1][y]);
    }
    // Down
    if (x + 1 <= 2)
    {
        swap(initial[x][y], initial[x + 1][y]);
        if (!visited[converter(initial)])
        {
            printMat(initial);
            solve(initial, final);
        }
        swap(initial[x][y], initial[x + 1][y]);
    }
    // right
    if (y + 1 <= 2)
    {
        swap(initial[x][y], initial[x][y + 1]);
        if (!visited[converter(initial)])
        {
            printMat(initial);
            solve(initial, final);
```

Information Technology Engineering Department

Academic Year: 2022-2023

Class: TE **Sem.:** V **Course:** Artificial Intelligence/Machine Learning

```
            }
            swap(initial[x][y], initial[x][y + 1]);
        }
        // left
        if (y - 1 >= 0)
        {
            swap(initial[x][y], initial[x][y - 1]);
            if (!visited[converter(initial)])
            {
                printMat(initial);
                solve(initial, final);
            }
            swap(initial[x][y], initial[x][y - 1]);
        }
    }
}

int main()
{
    vector<vector<int>> initial = {{1, 2, 5}, {3, 4, 0}, {6, 7, 8}};
    vector<vector<int>> final = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
    solve(initial, final);
    return 0;
}
```

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Information Technology Engineering Department

Academic Year: 2022-2023

Class: TE **Sem.:** V **Course:** Artificial Intelligence/Machine Learning

Output:

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

3 1 2
6 4 0
7 8 5

3 1 2
6 4 5
7 8 0

3 1 2
6 4 5
7 0 8

3 1 2
6 4 5
0 7 8

3 1 2
0 4 5
6 7 8

0 1 2
3 4 5
6 7 8

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

Conclusion:

In this experiment I learned how to implement an 8-puzzle program using UNINFORMED SEARCH STRATEGY.

Thank You