AI Lab Assignment 02

Problem Statement: *8 Puzzle*

NAME: **Sahil Dattatray Mohite** ROLLNO: **29**
CLASS: TY(IT) BATCH: **02**

Code :

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
#include <bits/stdc++.h>
using namespace std;

bool isValid(vector<int>[]);
int h1(vector<int>[], vector<int>[]);
int h2(vector<int>[], vector<int>[]);
void printState(vector<int>[]);
bool check(vector<int>[], vector<int>[]);
int countPossibleMoves(vector<int>[]);

int emptyPosition;

int ANS = 0; // How many steps it is taking to achieve Goal State

// count Number of misplaced tiles w.r.t goal state.
int h1(vector<int> &Goal, vector<int> &v)
{
    int count = 0;
    for (int i = 0; i < 9; i++)
    {
        if (Goal[i] != v[i])
            count++;
    }
    return count;
}

// Calculating euclidian distace of current state with Goal State
int h2(vector<int> &Goal, vector<int> &v)
{
    int ans = 0;
    for (int i = 0; i < 9; i++)
    {
        ans += pow(Goal[i] - v[i], 2);
    }

    ans = sqrt(ans);
```

```
    return ans;
}

// couting possible moves
int countPossibleMoves(vector<int> &v)
{
    vector<int> possibleMoves = {2, 3, 2, 3, 4, 3, 2, 3, 2}; // store
possible moves considering at each position 0

    for (int i = 0; i < 9; i++)
    {
        if (v[i] == 0)
        {
            emptyPosition = i;
            return possibleMoves[i];
        }
    }
    return 0;
}

// Printing State
void printState(vector<int> v)
{
    int count = 0;
    for (int i = 0; i < 9; i++)
    {
        count++;
        cout << v[i] << " ";
        if (count == 3)
        {
            count = 0;
            cout << endl;
        }
    }

    cout << endl;
}

// checking given state with goal state
bool check(vector<int> &Goal, vector<int> &v)
{
    for (int i = 0; i < 9; i++)
    {
        if (Goal[i] != v[i])
            return false;
    }
    return true;
```

```
}

// function to generate possble moves
vector<int> generatePossibleMove(vector<int> v, int i)
{

    vector<int> s;

    /*
    if empty position is

    at position 0  -- total posible moves = 2  swap(0,1) swap(0,3)
    at position 1  -- total posible moves = 3  swap(1,0) swap(1,2) swpa(1,4)
    at position 2  -- total posible moves = 2  swap(2,1) swap(2,5)

    at position 3  -- total posible moves = 3  swap(3,0) swap(3,4) swap(3,6)
    at position 4  -- total posible moves = 4  swap(4,1) swap(4,5) swap(4,7)
swap(4,3)
    at position 5  -- total posible moves = 3  swap(5,2) swap(5,4) swap(5,8)

    at position 6  -- total posible moves = 2  swap(6,3) swap(6,7)
    at position 7  -- total posible moves = 3  swap(7,4) swap(7,6) swap(7,8)
    at position 9  -- total posible moves = 2  swap(8,5) swap(8,7)
*/

    vector<vector<pair<int, int>>> swapPositions = {
        {{0, 1}, {0, 3}},
        {{1, 0}, {1, 2}, {1, 4}},
        {{2, 1}, {2, 5}},
        {{3, 0}, {3, 4}, {3, 6}},
        {{4, 1}, {4, 5}, {4, 7}, {4, 3}},
        {{5, 2}, {5, 4}, {5, 8}},
        {{6, 3}, {6, 7}},
        {{7, 4}, {7, 6}, {7, 8}},
        {{8, 5}, {8, 7}}};

    swap(v[swapPositions[emptyPosition][i].first],
         v[swapPositions[emptyPosition][i].second]);

    return v;
}

bool isValid(vector<int> &v)
{
    vector<bool> present(9, false);
    for (int i = 0; i < 9; i++)
    {
```

```
        if (v[i] < 0 || v[i] > 8)
        {
            cout << "You have entered invalid position. It must be between 0
to 8";
            return false;
        }
        if (present[v[i]] == true)
        {
            cout << "You have entered repeated element at position " << i;
            return false;
        }

        present[v[i]] = true;
    }

    cout << endl;
    return true;
}

int main()
{
    // Storing all states:
    set<vector<int>> allStates;

    // Goal State
    vector<int> Goal{1, 2, 3, 8, 0, 4, 7, 6, 5};
    cout << "\nGoal State is: " << endl;
    printState(Goal);

    // Initial State
    //vector<int> v = {0, 2, 3, 4, 1, 5, 6, 7, 8};
    // vector<int> v = {2, 0, 3, 4, 1, 5, 6, 7, 8};
    //vector<int> v = {3, 2, 0, 4, 1, 5, 6, 7, 8};
    // vector<int> v = {4, 2, 3, 0, 1, 5, 6, 7, 8};
     vector<int> v = {1, 2, 3, 4, 0, 5, 6, 7, 8};
    // vector<int> v = {5, 2, 3, 4, 1, 0, 6, 7, 8};
    // vector<int> v = {6, 2, 3, 4, 1, 5, 0, 7, 8};
    // vector<int> v = {7, 2, 3, 4, 1, 5, 6, 0, 8};
    // vector<int> v = {8, 2, 3, 4, 1, 5, 6, 7, 0};

    if (!isValid(v))
        return 0;

    cout << "Initial State is: " << endl;
    printState(v);

    // if(h1(Goal,v)==0 || h2(Goal,v) == 0)
    // {
```

```
    //     cout<<"Congratulation!! You Have Achieved Goal State.";
    //     return 0;
    // }


    if (check(Goal, v))
    {
        cout << "Congratulation!! You Have Achieved Goal State.";
        return 0;
    }


    cout << "Number of misplaced tiles are: " << h1(Goal, v) << endl <<
endl;

    int possibleMoves = countPossibleMoves(v);
    cout << "Number of Possible moves: " << possibleMoves << endl << endl;
    cout << "Possible moves are as follows: " << endl << endl;

    vector<int> scores;
    map<int, vector<int>> mp;

    for (int i = 0; i < possibleMoves; i++)
    {
        vector<int> vv = generatePossibleMove(v, i);
        printState(vv);
        int d = h2(Goal, vv);
        scores.push_back(d);
        cout << i + 1 << ")Euclidiean distance for possible move " << i + 1
<< " is : " << d;

        mp[i] = vv;

        cout << endl << endl << endl;
    }

    int mini = INT_MAX;
    int mini_index = 0;

    for (int i = 0; i < scores.size(); i++)
    {
        if (scores[i] < mini)
        {
            mini_index = i;
            mini = scores[i];
            if (allStates.count(mp[mini_index]))
            {
                mini = INT_MAX;
```

```
            }
        }
    }

    cout << "Smallest euclidian distance is: " << mini << endl;
    cout << "So, best possible move is: " << endl<< endl;

    // check as euclidian distance is repeating
    printState(mp[mini_index]);

    allStates.insert(mp[mini_index]);

    v.assign(mp[mini_index].begin(), mp[mini_index].end());

    scores.clear();
    mp.clear();
    return 0;
}
```

Output:

```
Goal State is:
1 2 3
8 0 4
7 6 5



Initial State is:
1 2 3
4 0 5
6 7 8

Number of misplaced tiles are: 5

Number of Possible moves: 4

Possible moves are as follows:

1 0 3
4 2 5
6 7 8

1)Euclidiean distance for possible move 1 is : 6


1 2 3
```

4 5 0
6 7 8

2)Euclidiean distance for possible move 2 is : 8

1 2 3
4 7 5
6 0 8

3)Euclidiean distance for possible move 3 is : 10

1 2 3
0 4 5
6 7 8

4)Euclidiean distance for possible move 4 is : 9

Smallest euclidian distance is: 6
So, best possible move is:

1 0 3
4 2 5
6 7 8