Assignment No: 01

CSE-0408 Summer 2021

Joy Sarkar
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
joysarker39@gmail.com

***Abstract*—In practice, an incomplete heuristic search nearly always finds better solutions if it is allowed to search deeper, i.e. expand and heuristically evaluate more nodes in the search tree and determine the best path to take next.**

***Index Terms*—heuristic, puzzle**

I. Introduction

Many problems, such as game-playing and path-finding, can be solved by search algorithms. To do so, the problems are represented by a search graph or tree in which the nodes correspond to the states of the problem. In this assignment we are going to implement a algorithms to solve 8 puzzle problem.

II. Literature Review

Sadikov and Bratko (2006) studied the suitability of pessimistic and optimistic heuristic functions for a real-time search in the 8-puzzle. They discovered that pessimistic functions are more suitable. They also observed the pathology, which was stronger with the pessimistic heuristic function. However, they did not study the influence of other factors on the pathology or provide any analysis of the gain of a deeper search.

III. Proposed Methodology

IV. Conclusion

We tested our code to see how many states it would take to get from the current state to the goal state, and we came up with seven.

Acknowledgment

I would like to thank my honourable**Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] Piltaver, R., Luštrek, M., & Gams, M. (2012). The pathology of heuristic search in the 8-puzzle. Journal of Experimental & Theoretical Artificial Intelligence, 24(1), 65-94.

```cpp
#include<bits/stdc++.h>
using namespace std;
#define D(x) cerr<<__LINE__<<" : "<<#x<<" -> "<<x<<endl
#define rep(i,j) for(int i = 0; i < 3; i++) for(int j = 0; j < 3; j++)
#define PII pair < int, int >
typedef vector<vector<int>> vec2D;

const int MAX = 1e5+7;
int t=1, n, m, l, k, tc;

int dx[4] = {0, 0, 1, -1};
int dy[4] = {1, -1, 0, 0};

vec2D init{
    {8, 1, 2},
    {3, 6, 4},
    {0, 7, 5}
};
vec2D goal{
    {1, 3, 2},
    {8, 0, 4},
    {7, 6, 5}
};
/// using a structure to store information of each state
struct Box {
    vec2D mat{ { 0,0,0 },{ 0,0,0},{ 0,0,0} };
    int diff, level;
    int x, y;
    int lastx, lasty;
    Box(vec2D a,int b = 0, int c = 0, PII p = {0,0}, PII q = {0,0}) {
        rep(i,j) mat[i][j] = a[i][j];
        diff = b;
        level = c;
        x = p.first;
        y = p.second;
        lastx = q.first;
        lasty = q.second;
    }
};

/// operator overload for which bases priority queue work
bool operator < (Box A, Box B) {
    if(A.diff == B.diff) return A.level < B.level;
    return A.diff < B.diff;
}

```

Fig. 1. Proposed Methodology

```
46
47    /// heuristic function to calculate mismatch position
48    int heuristic_function(vec2D a, vec2D b) {
49        int ret(0);
50        rep(i,j) if (a[i][j] != b[i][j]) ret--;
51        return ret;
52    }
53
54    /// checking puzzle boudaries
55    bool check(int i, int j) {
56        return i>=0 and i<3 and j>=0 and j<3;
57    }
58
59    /// this function used to show state status
60    void print(Box a) {
61        rep(i,j)
62        cout << a.mat[i][j] << (j == 2 ? "\n" : " ");
63        cout << " heuristic Value is :  " << -a.diff << "\n";
64        cout << " Current level is : " << -a.level << "\n\n";
65    }
66
67    /// used to get new state which can be jump from current state
68    Box get_new_state(Box now, int xx, int yy) {
69        Box temp = now;
70        swap(temp.mat[temp.x][temp.y], temp.mat[xx][yy]);
71        temp.diff = heuristic_function(temp.mat, goal);
72        temp.level = now.level - 1;
73        temp.x = xx;
74        temp.y = yy;
75        temp.lastx = now.x;
76        temp.lasty = now.y;
77        return temp;
78    }
79
```

Fig. 2. Proposed Methodology

```
80
81  /// this is modified version of dijkstra shortest path algorithms
82  /// basically work on those state first which heuristic value lesser
83  void dijkstra(int x, int y) {
84      map < vec2D,  bool > mp;
85      priority_queue < Box > PQ;
86      int nD = heuristic_function(init, goal);
87      Box src = {init, nD, 0, {x,y}, {-1,-1}};
88      PQ.push(src);
89      int state = 0;
90      while(!PQ.empty()) {
91          state++;
92          Box now = PQ.top();
93          PQ.pop();
94          cout << "Step no : " << state-1 <<"\n";
95          print(now);
96          if(!now.diff) { /// if heuristic value is zero it means we are on goal
97              puts("Goal state has been discovered");
98              cout << "level : " << -now.level << "\n";
99              cout << " Step no : " << state-1 <<"\n";
100             break;
101         }
102         if(mp[now.mat]) continue;
103         mp[now.mat] = true;
104         for(int i = 0; i < 4; i++) {
105             int xx = now.x + dx[i];
106             int yy = now.y + dy[i];
107             if(check(xx, yy)) {
108                 if(now.lastx == xx and now.lasty == yy) continue;
109                 Box temp = get_new_state(now, xx, yy);
110                 PQ.push(temp);
111             }
112         }
113     }
114 }
115
116 signed main() {
117     puts("Current State:");
118     rep(i,j) cout << init[i][j] << (j == 2 ? "\n" : " ");
119     puts("");
120     puts("Goal State:");
121     rep(i,j) cout << goal[i][j] << (j == 2 ? "\n" : " ");
122     puts("\n.............Search Started..............\n");
123     rep(i,j) if(!init[i][j]) dijkstra(i,j); /// this will find zero-th positio
124     return 0;
125 }
```

Fig. 3. Proposed Methodology

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

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

.............Search Started.................

Step no : 0
8 1 2
3 6 4
0 7 5
 heuristic Value is :  6
 Current level is : 0

Step no : 1
8 1 2
3 6 4
7 0 5
 heuristic Value is :  5
 Current level is : 1

Step no : 2
8 1 2
3 0 4
7 6 5
 heuristic Value is :  3
 Current level is : 2

Step no : 3
8 1 2
0 3 4
7 6 5
 heuristic Value is :  4
 Current level is : 3

Step no : 4
0 1 2
8 3 4
7 6 5
 heuristic Value is :  3
 Current level is : 4

Step no : 5
1 0 2
8 3 4
7 6 5
 heuristic Value is :  2
 Current level is : 5

Step no : 6
1 3 2
8 0 4
7 6 5
 heuristic Value is :  0
 Current level is : 6

Goal state has been discovered
level : 6
 Step no : 6

Process returned 0 (0x0)   execution time : 0.003 s
Press ENTER to continue.
```