|  |
| --- |
| **NAME:** GOKULRAJ V |
| **ROLL\_NO:** 1905014 |

|  |
| --- |
| **DATE:** 12/08/2022 |
| **EXP\_NO:** 03 |

|  |
| --- |
| **A\* Algorithm** |

**Aim**:

To solve the car parking problem with the help of A\* Algorithm.

**Problem Statement:**

n vehicles occupy squares (1, 1) through (n, 1) (i.e., the bottom row) of an n×n grid. The vehicles must be moved to the top row but in reverse order; so the vehicle i that starts in (i, 1) must end up in (n−i+1, n). On each time step, every one of the n vehicles can move one square up, down, left, or right, or stay put; but if a vehicle stays put, one other adjacent vehicle (but not more than one) can hop over it. Two vehicles cannot occupy the same square. Suppose that vehicle i is at (xi, yi); write a nontrivial admissible heuristic hi for the number of moves it will require to get to its goal location (n − i + 1, n), assuming no other vehicles are on the grid.

**CODE:**

#include<bits/stdc++.h>

using namespace std;

typedef pair<int,vector<vector<int>> > pi;

queue <vector<vector<int>> > st;

priority\_queue<pi, vector<pi> , greater<pi> > que;

int find\_heuristics(vector<vector<int>> pos, vector<vector<int>> goal)

{

int temp=0;

for(int i=0;i<pos.size();i++)

{

temp+=abs(pos[i][0]-goal[i][0])+abs(pos[i][1]-goal[i][1]);

}

return temp;

}

bool route(vector<vector<int>>,vector<vector<int>>,int);

int main()

{

vector<vector<int>> mat {{1,0,0},{2,0,0},{3,0,0}};

vector<vector<int>> goal {{0,0,3},{0,0,2},{0,0,1}};

vector<vector<int>> init\_pos {{0,0},{1,0},{2,0}};

vector<vector<int>> goal\_pos {{2,2},{1,2},{0,2}};

cout<<"START STATE"<<endl;

for(int i=0;i<mat.size();i++)

{

for(int j=0;j<mat[0].size();j++)

cout << mat[i][j] << " ";

cout << endl;

}

cout<<endl<<"GOAL STATE"<<endl;

for(int i=0;i<goal.size();i++)

{

for(int j=0;j<goal[0].size();j++)

cout << goal[i][j] << " ";

cout << endl;

}

cout<<endl;

route(init\_pos,goal\_pos,0);

return 0;

}

bool route(vector<vector<int>> pos, vector<vector<int>> goal,int gx)

{

if(pos==goal)

{

cout<< "goal reached "<<gx<<endl;

while(!st.empty())

{

int arr[3][3]={{0,0,0},{0,0,0},{0,0,0}};

vector<vector<int>> temp=st.front();

for(int i=0;i<temp.size();i++)

{

arr[temp[i][0]][temp[i][1]]=i+1;

}

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

cout << arr[i][j] << " ";

cout << endl;

}

cout<<endl;

st.pop();

}

return true;

}

int heuristics;

vector<vector<int>> dir {{0,1},{-1,0},{0,-1},{1,0}};

for(int i=0;i<3;i++)

{

int r=pos[i][0],c=pos[i][1];

for(int j=0;j<4;j++)

{

if(r+dir[j][0] >=0 && r+dir[j][0] <= 2 && c+dir[j][1]>=0 && c+dir[j][1] <=2)

{

int k;

for(k=0;k<3;k++)

{

if(r+dir[j][0]==pos[k][0] && c+dir[j][1]==pos[k][1])

break;

}

if(k==3)

{

pos[i][0]+=dir[j][0];

pos[i][1]+=dir[j][1];

heuristics = find\_heuristics(pos,goal);

que.push(make\_pair(heuristics,pos));

pos[i][0]-=dir[j][0];

pos[i][1]-=dir[j][1];

}

}

}

}

if(!que.empty())

{

vector<vector<int>> temp=que.top().second;

int t=que.top().first;

que.pop();

st.push(temp);

route(temp,goal,gx+1);

st.pop();

}

else

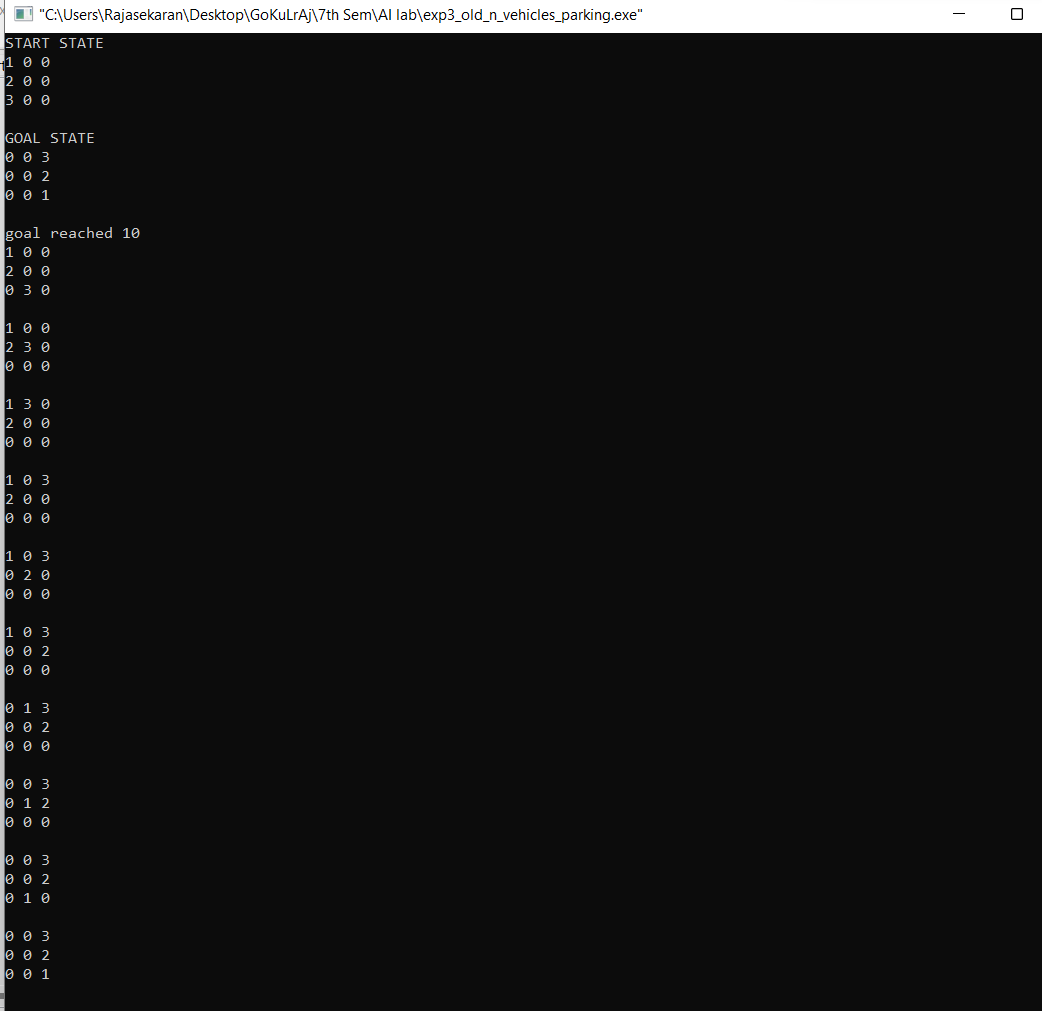
{

cout<<"not possible";

}

}

**OUTPUT:**



**Result:**

Thus, the code has been successfully executed.