

State University of Bangladesh



Course No: CSE-0408

Course Name: Artificial Intelligence lab

Semester: Summer 2021

Submitted to:

Khan Md.Hasib

Lecturer,

Dept. of CSE, SUB

Submitted by:

Name: Shawon Mia

ID: UG02-44-17-025

Batch: 44

Email: refta6866@gmail.com

Question 1:

Write a program in any language (C, C++, Java,Python, R) to solve the 8-Puzzle problem using heuristic functions.

Solution:

```
#include<bits/stdc++.h>
using namespace std;
struct state{
int grid[3][3]; int cost;
int px,py,x,y,l = 0;
};
int row[4] = { 1, 0, -1, 0 };
int col[4] = { 0, -1, 0, 1 };
bool operator < (state a, state b){
return a.cost+a.l > b.cost+b.l;
}
int costCalculate(int current_state[3][3] ,
int final_state[3][3]){
int c = 0;
for(int i=0;i<3;i++)
for(int j=0;j<3;j++)
```

```

        if (current_state[i][j] &&
current_state[i][j] != final_state[i][j])
c++;

        return c;
    }

bool isSafe(int x, int y){
return (x>=0 && x<3 && y>=0 && y<3);
}

void solution(int initial[3][3],
int final_state[3][3], int x, int y){
priority_queue<state>pq;
    state node;
    node.x = x; node.y = y;
    node.l = 0;
    node.px = -1; node.py = -1;
node.cost = costCalculate(initial,
final_state);
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
node.grid[i][j] = initial[i][j];
    pq.push(node);
    int f = 0;

```

```

while (!pq.empty()) {
    f = 0; node = pq.top(); pq.pop();
    x = node.x; y = node.y;
    for (int k=0; k<4; k++){
int    positionX = x+row[k];
int    positionY = y+col[k];
        state child;
        if (isSafe(positionX, positionY)) {
            if (positionX == node.px &&
                positionY == node.py)
                continue;
            for (int i=0; i<3; i++)
                for (int j=0; j<3; j++)
                    child.grid[i][j] = node.grid[i][j];
            swap(child.grid[x][y]);
            child.grid[positionX][positionY]);
            child.px = node.x;
            child.py = node.y;
            child.x = positionX;
            child.y = positionY;
            child.l = node.l + 1;
            child.cost = costCalculate(child.grid,

```

```

        final_state );
        if ( child.cost == 0 ){
            printf(" Final State: \n");
            for ( int i=0;i<3;i++){
                for ( int j=0;j<3;j++){
                    cout<<child.grid[i][j]<<" ";
                    cout<<endl;
                }
            }
            f = 1;
            break;
        }
        pq.push( child );
    }
}
if ( f ) break;
}
}

int main(){
    int init[3][3] = {{7, 2, 3},{4, 6, 5},
                      {1, 8, 0}};
    int goal[3][3] = {{1, 2, 3},{4, 5, 6},
                      {7, 8, 0}};

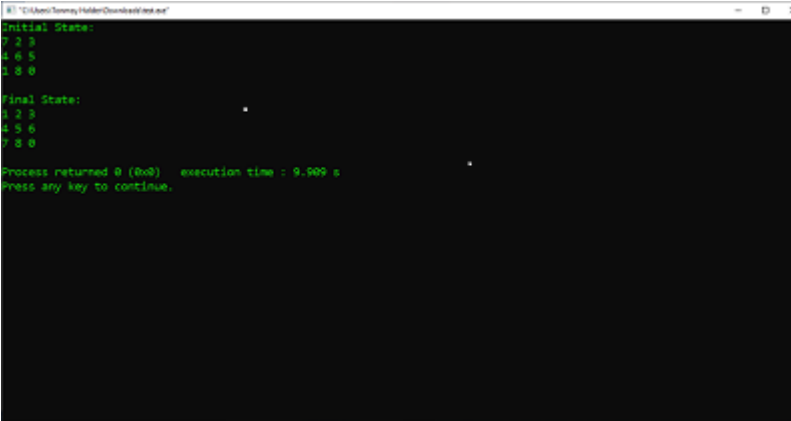
```

```

int x = 2, y = 2;
cout<<"Initial State:"<<endl;
for (int i=0;i<3;i++){
for (int j=0;j<3;j++) cout<<init [ i ][ j]<<" ";
cout<<endl;
}
cout<<endl;
solution (init , goal , x, y);
}

```

Output:



```

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

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

Process returned 0 (No)   execution time : 9.909 s
Press any key to continue.

```

Explanation

This program is based on Heuristic search.

A Heuristic is a technique to solve a problem faster than classic methods, or to find an approximate solution when classic methods cannot.

This is a kind of a shortcut as we often trade one of optimal, completeness, accuracy, or precision for speed. A Heuristic (or a heuristic function) takes a look at search algorithms. At each branching step, it evaluates the available information and makes a decision on which branch to follow. It does so by ranking alternatives. The Heuristic is any device that is often effective but will not guarantee work in every case.

Here the task is we have to take 0 to 8 numbers randomly as input then we will get that numbers sorted ascending order.

Question 2:

Write a program about BFS implementation in any language (C, C++, JAVA, PYTHON).

Solution:

```
#include <iostream>
#include<conio.h>
#include<stdlib.h>
#include <bits/stdc++.h>
using namespace std;
int c[10][10],i,j,k,n,q[10];
int front,rare,v,visit[10];
int visited[10];
int main()
{
int m;
cout <<"Enter no of vertices:";
cin >> n;
cout <<"Enter no of edges:";
cin >> m ;
co t <<"\nEIGES \n";
```



```

for (k=1; k<=m; k++)
{
cin >>i>>j;
c[i][j]=1;
}
cout <<"Enter initial vertex
to traverse from:";
cin >>v;
cout <<"Visited vertices:";
cout <<v<<" ";
visited[v]=1;
k=1;
while(k<n)
{
for (j=1; j<=n; j++)
if (c[v][j]!=0 && visited[j]!=1
&& visit[j]!=1)
{
visit[j]=1;
q[rare++]=j;
}
v=q[front++];

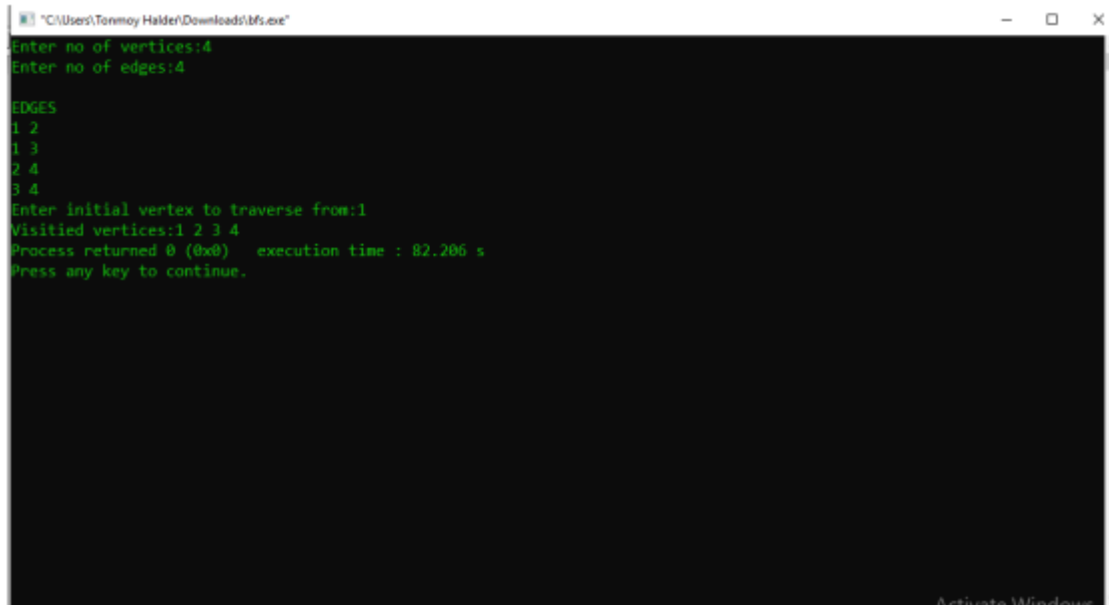
```

```

        cout<<v <<" ";
        k++;
        visit [v]=0;
        visited [v]=1;
    }
    return 0;
}

```

Output:



```

C:\Users\Tonmoy Halder\Downloads\bf.exe
Enter no of vertices:4
Enter no of edges:4

EDGES
1 2
1 3
2 4
3 4
Enter initial vertex to traverse from:1
Visited vertices:1 2 3 4
Process returned 0 (0x0)   execution time : 82.206 s
Press any key to continue.

```

Explanation

This program is based on BFS

- Breadth-first search starts at a given vertex s , which is at level 0.

- In the first stage, we visit all the vertices that are at the distance of one edge away. When we visit there, we paint as "visited," the vertices adjacent to the start vertex s - these vertices are placed into level 1.
- In the second stage, we visit all the new vertices we can reach at the distance of two edges away from the source vertex s . These new vertices, which are adjacent to level 1 vertices and not previously assigned to a level are placed into level 2, and so on.
- The BFS traversal terminates when every vertex has been visited.

Here we took 4 vertexes and 4 edges as input and the edges are

1 2

1 3

2 3

3 4

It started traversal from 1. Then final result is 1 2 3 4 .