

AI LAB ASSIGNMENT 6

NAME: PRATHAPANI SATWIK

REG NO: 20BCD7160

Solve 8-puzzle problem by using best first search

CODE:

```
package proplayer; import
java.util.*; public class
eightpuzzlebfs{ public
static int N = 3; public
static class Node{ Node
parent; int mat[][] = new
int[N][N]; int x, y; int cost;
int level;}

public static void printMatrix(int mat[][]){
for(int i = 0; i < N; i++){ for(int j = 0; j < N;
j++){
System.out.print(mat[i][j]+" ");}
System.out.println("");
}}

public static Node newNode(int mat[][], int x, int y,
int newX, int newY, int level,
Node parent){
Node node = new Node();
node.parent = parent;
```

```

node.mat = new int[N][N];
for(int i = 0; i < N; i++){ for(int
j = 0; j < N; j++){
node.mat[i][j] = mat[i][j];
}}
int temp = node.mat[x][y];
node.mat[x][y] = node.mat[newX][newY];
node.mat[newX][newY]=temp; node.cost
= Integer.MAX_VALUE; node.level = level;
node.x = newX; node.y = newY; return
node;
}

public static int row[] = { 1, 0, -1, 0 }; public
static int col[] = { 0, -1, 0, 1 };
public static int calculateCost(int initialMat[][], int finalMat[][]){
int count = 0; for (int i = 0; i < N; i++) for (int j = 0; j < N; j++)
if (initialMat[i][j]!=0 && initialMat[i][j] != finalMat[i][j])
count++; return count;}

public static int isSafe(int x, int y){
return (x >= 0 && x < N && y >= 0 && y < N)?1:0;
}

public static void printPath(Node root){
if(root == null){ return;}
printPath(root.parent);
printMatrix(root.mat);
System.out.println("");

```

```

}

public static class comp implements Comparator<Node>{

@Override

public int compare(Node lhs, Node rhs){ return

(lhs.cost + lhs.level) > (rhs.cost+rhs.level)?1:-1;

}}

public static void solve(int initialMat[][], int x, int

y, int finalMat[][]){

PriorityQueue<Node> pq = new PriorityQueue<>(new comp());

Node root = newNode(initialMat, x, y, x, y, 0, null); root.cost =

calculateCost(initialMat,finalMat); pq.add(root);

while(!pq.isEmpty()){ Node min = pq.peek(); pq.poll();

if(min.cost == 0){ printPath(min); return;

}

for (int i = 0; i < 4; i++){

if (isSafe(min.x + row[i], min.y + col[i])>0){

Node child = newNode(min.mat, min.x, min.y, min.x + row[i],min.y + col[i],

min.level + 1, min);

child.cost = calculateCost(child.mat, finalMat);

pq.add(child);

}}}}

public static void main (String[] args){ int

initialMat[][] ={

{1, 2, 3},

{5, 6, 0},

{7, 8, 4}

```

```
};  
int finalMat[][] =  
{1, 2, 3},  
{5, 8, 6},  
{0, 7, 4}  
};  
int x = 1, y = 2;  
solve(initialMat, x, y,finalMat);  
}  
}
```

```

eightpuzzlebfs.java × +
1  package proplayer;
2  import java.util.*;
3  public class eightpuzzlebfs{
4      public static int N = 3;
5      public static class Node{
6          Node parent;
7          int mat[][] = new int[N][N];
8          int x, y;
9          int cost;
10         int level;
11     }
12     public static void printMatrix(int mat[][]){
13         for(int i = 0; i < N; i++){
14             for(int j = 0; j < N; j++){
15                 System.out.print(mat[i][j]+" ");
16             }
17             System.out.println("");
18         }
19     }
20     public static Node newNode(int mat[][], int x, int y,
21         int newX, int newY, int level,
22         Node parent){
23         Node node = new Node();
24         node.parent = parent;
25         node.mat = new int[N][N];
26         for(int i = 0; i < N; i++){
27             for(int j = 0; j < N; j++){
28                 node.mat[i][j] = mat[i][j];
29             }
30         }
31         int temp = node.mat[x][y];
32         node.mat[x][y] = node.mat[newX][newY];
33         node.mat[newX][newY] = temp;
34         node.cost = Integer.MAX_VALUE;
35     }
36     node.level = level;
37     node.x = newX;
38     node.y = newY;
39     return node;
40 }
41 public static int row[] = { 1, 0, -1, 0 };
42 public static int col[] = { 0, -1, 0, 1 };
43 public static int calculateCost(int initialMat[][], int finalMat[][]){
44     int count = 0;
45     for (int i = 0; i < N; i++)
46         for (int j = 0; j < N; j++)
47             if (initialMat[i][j]!=0 && initialMat[i][j] != finalMat[i][j])
48                 count++;
49     return count;
50 }
51 public static int isSafe(int x, int y){
52     return (x >= 0 && x < N && y >= 0 && y < N)?1:0;
53 }
54 public static void printPath(Node root){
55     if(root == null){
56         return;
57     }
58     printPath(root.parent);
59     printMatrix(root.mat);
60     System.out.println("");
61 }
62 public static class comp implements Comparator<Node>{
63     @Override
64     public int compare(Node lhs, Node rhs){
65         return (lhs.cost + lhs.level) > (rhs.cost+rhs.level)?1:-1;
66     }
67 }
68 public static void solve(int initialMat[][], int x,

```

```

61 ▼ int y, int finalMat[][]){
62     PriorityQueue<Node> pq = new PriorityQueue<>(new comp());
63     Node root = newNode(initialMat, x, y, x, y, 0, null);
64     root.cost = calculateCost(initialMat,finalMat);
65     pq.add(root);
66 ▼ while(!pq.isEmpty()){
67     Node min = pq.peek();
68     pq.poll();
69 ▼ if(min.cost == 0){
70     printPath(min);
71     return;
72 }
73 ▼ for (int i = 0; i < 4; i++){
74 ▼ if (isSafe(min.x + row[i], min.y + col[i])>0){
75     Node child = newNode(min.mat, min.x, min.y, min.x + row[i],min.y + col[i],
76     min.level + 1, min);
77     child.cost = calculateCost(child.mat, finalMat);
78     pq.add(child);
79     }}}
80 ▼ public static void main (String[] args){
81 ▼ int initialMat[][] ={
82     {1, 2, 3},
83     {5, 6, 0},
84     {7, 8, 4}
85 };
86 ▼ int finalMat[][] ={
87     {1, 2, 3},
88     {5, 8, 6},
89     {0, 7, 4}
90 };

```

```

91     int x = 1, y = 2;
92     solve(initialMat, x, y,finalMat);
93 }
94 }
95
96 |

```

OUTPUT:

```

>_ Console × +
✱ java eightpuzzlebfs.java
1 2 3
5 6 0
7 8 4

1 2 3
5 0 6
7 8 4

1 2 3
5 8 6
7 0 4

1 2 3
5 8 6
0 7 4

✱ []

```