Name: Mihir Thakkar

Class: TY A

Roll No: 59

SRN: 201901267

AI ASSIGNMENT-3

Hill Climbing Problem for 8 Puzzle:

1)Choose initial configuration such that the algorithm terminates with all tiles in position.

CODE:

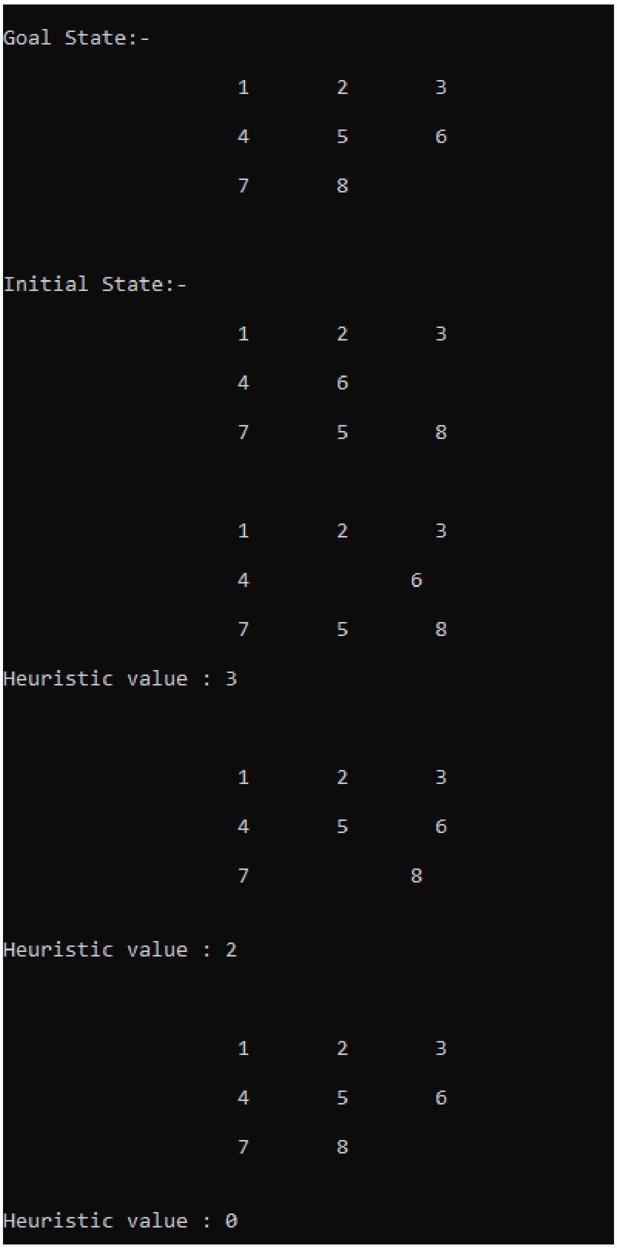
```
#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

int finalstate[3][3], initialstate[3][3];

void final_state(int num)

{

if(num == 1)

{

finalstate[0][0] = 1;

finalstate[0][1] = 2;

finalstate[0][2] = 3;

finalstate[1][0] = 4;

finalstate[1][1] = 5;

finalstate[1][2] = 6;

finalstate[2][0] = 7;

finalstate[2][1] = 8;

finalstate[2][2] = -1;

}

else


{

initialstate[0][0] = 1;

initialstate[0][1] = 2;
initialstate[0][2] = 3;
initialstate[1][0] = 4;
initialstate[1][1] = 6;
initialstate[1][2] = -1;
initialstate[2][0] = 7;
initialstate[2][1] = 5;
initialstate[2][2] = 8;
}
}
void state_display(int arr[3][3])
{
int i,j;
printf("\n\n");
for(i=0;i<3;i++)
{
printf("\t\t");
for(j=0;j<3;j++)
{
if(arr[i][j] == -1)
printf("      ");
else
printf("   %d    ",arr[i][j]);
}

printf("\n\n");
}
}
int heuristicval(int arr[3][3])
{
int i, j, h=0;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
if(arr[i][j]!=finalstate[i][j])
h++;
return h;
}
int hillclimbing(int arr[3][3])
{
int m,n,i,j, k=0,
m1,n1,m2,n2,m3,n3,m4,n4,hinit,hleft,hright,hup,hdown,min = 9;
hinit = heuristicval(arr);
if(hinit==0)
{
return;
}
for(i=0;i<3;i++)
for(j=0;j<3;j++)
if(arr[i][j] == -1)
{
m = i;

n = j;
}
while(k<4)
{
if(k==0)
{
m1=m;
n1=n;
n1--;
if(n1>-1)
{
```

```
arr[m][n] = arr[m1][n1];

arr[m1][n1] = -1;

hleft = heuristicval(arr);

if(hinit>hleft)

{

state_display(arr);

printf("Heuristic value : %d\n",hleft);

hillclimbing(arr);

break;

}

arr[m1][n1] = arr[m][n];

arr[m][n] = -1;

}

}

else if(k==1)


{

m2=m;

n2=n;

m2--;

if(m2>-1)

{

arr[m][n] = arr[m2][n2];

arr[m2][n2] = -1;

hup = heuristicval(arr);

if(hinit>hup)

{

state_display(arr);

printf("Heuristic value : %d\n",hup);

hillclimbing(arr);

break;
}
arr[m2][n2] = arr[m][n];
arr[m][n] = -1;
}

printf("\n");
}
else if(k==2)
{
m3=m;
n3=n;

n3++;
if(n3<3)
{
arr[m][n] = arr[m3][n3];
arr[m3][n3] = -1;
hright = heuristicval(arr);
if(hinit>hright)
{
state_display(arr);

printf("\n");
printf("Heuristic value : %d\n",hright);
hillclimbing(arr);
break;
}
arr[m3][n3] = arr[m][n];
arr[m][n] = -1;
}
}
```

```
else if(k==3)
{
m4=m;
n4=n;
m4++;

if(m4<3)
{
arr[m][n] = arr[m4][n4];
arr[m4][n4] = -1;
hdown = heuristicval(arr);
if(hinit>hdown)
{
state_display(arr);
printf("\n");
printf("Heuristic value : %d\n",hdown);
hillclimbing(arr);
break;
}
arr[m4][n4] = arr[m][n];
arr[m][n] = -1;
}
}
k++;
}
}
int main()
{
final_state(1);
printf("\nGoal State:- ");
state_display(finalstate);
```

```
final_state(2);


printf("\n");


printf("\nInitial State:- ");
state_display(initialstate);
heuristicval(initialstate);
hillclimbing(initialstate);
return 0;
}
```

OUTPUT:



```
Goal State:-

                   1       2       3

                   4       5       6

                   7       8



Initial State:-

                   1       2       3

                   4       6

                   7       5       8



                   1       2       3

                   4             6

                   7       5       8

Heuristic value : 3



                   1       2       3

                   4       5       6

                   7             8


Heuristic value : 2



                   1       2       3

                   4       5       6

                   7       8


Heuristic value : 0
```

2)Choose initial configuration such that the algorithm terminates with in either a local maxima or a plateau.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int finalstate[3][3], initialstate[3][3];
void final_state(int num)
{
if(num == 1)
{
finalstate[0][0] = 1;
finalstate[0][1] = 2;
finalstate[0][2] = 3;
finalstate[1][0] = 4;
finalstate[1][1] = 5;
finalstate[1][2] = 6;
finalstate[2][0] = 7;
finalstate[2][1] = 8;
finalstate[2][2] = -1;
}
else

{
initialstate[0][0] = 1;
initialstate[0][1] = 2;
initialstate[0][2] = 3;
initialstate[1][0] = 7;
initialstate[1][1] = -1;
initialstate[1][2] = 5;
initialstate[2][0] = 4;
initialstate[2][1] = 8;
initialstate[2][2] = 6;
}
}
void state_display(int arr[3][3])
{
int i,j;
printf("\n\n");
for(i=0;i<3;i++)
{
printf("\t\t");
for(j=0;j<3;j++)
{
if(arr[i][j] == -1)
printf("      ");
else
printf("   %d    ",arr[i][j]);
}

printf("\n\n");
}
}
int heuristicval(int arr[3][3])
{
int i, j, h=0;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
if(arr[i][j]!=finalstate[i][j])
h++;
return h;
}
```

```
int hillclimbing(int arr[3][3])
{
int m,n,i,j, k=0,
m1,n1,m2,n2,m3,n3,m4,n4,hinit,hleft,hright,hup,hdown,min = 9;
hinit = heuristicval(arr);
if(hinit==0)
{
return;
}
for(i=0;i<3;i++)
for(j=0;j<3;j++)
if(arr[i][j] == -1)
{
m = i;

n = j;
}
while(k<4)
{
if(k==0)
{
m1=m;
n1=n;
n1--;
if(n1>-1)
{
arr[m][n] = arr[m1][n1];
arr[m1][n1] = -1;
hleft = heuristicval(arr);
if(hinit>hleft)
{
```

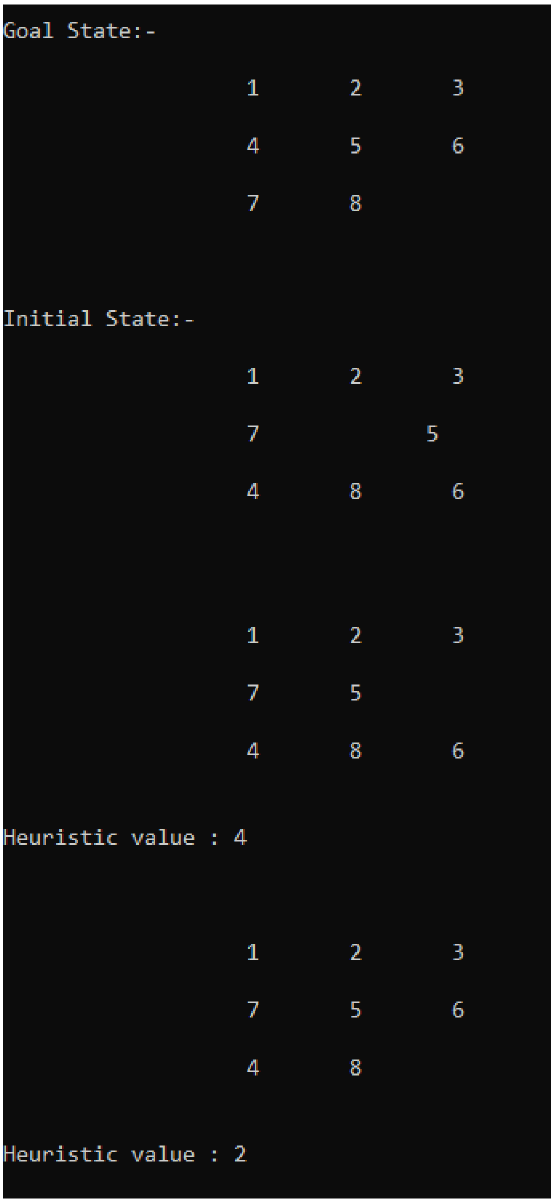
```
state_display(arr);
printf("Heuristic value : %d\n",hleft);
hillclimbing(arr);
break;
}
arr[m1][n1] = arr[m][n];
arr[m][n] = -1;
}
}
else if(k==1)

{
m2=m;
n2=n;
m2--;
if(m2>-1)
{
arr[m][n] = arr[m2][n2];
arr[m2][n2] = -1;
hup = heuristicval(arr);
if(hinit>hup)
{
state_display(arr);
printf("Heuristic value : %d\n",hup);
hillclimbing(arr);
break;
}
arr[m2][n2] = arr[m][n];
arr[m][n] = -1;
}
```

```
printf("\n");
}
else if(k==2)
{
m3=m;
n3=n;

n3++;
if(n3<3)
{
arr[m][n] = arr[m3][n3];
arr[m3][n3] = -1;
hright = heuristicval(arr);
if(hinit>hright)
{
state_display(arr);

printf("\n");
printf("Heuristic value : %d\n",hright);
hillclimbing(arr);
break;
}
arr[m3][n3] = arr[m][n];
arr[m][n] = -1;
}
}
else if(k==3)
{
m4=m;
n4=n;
m4++;
```

```
if(m4<3)
{
arr[m][n] = arr[m4][n4];
arr[m4][n4] = -1;
hdown = heuristicval(arr);
if(hinit>hdown)
{
state_display(arr);
printf("\n");
printf("Heuristic value : %d\n",hdown);
hillclimbing(arr);
break;
}
arr[m4][n4] = arr[m][n];
arr[m][n] = -1;
}
}
k++;
}
}
int main()
{
final_state(1);
printf("\nGoal State:- ");
state_display(finalstate);
final_state(2);

printf("\n");

printf("\nInitial State:- ");
```

```
state_display(initialstate);

heuristicval(initialstate);

hillclimbing(initialstate);

return 0;

}
```

OUTPUT:



```
Goal State:-

                   1       2       3

                   4       5       6

                   7       8


Initial State:-

                   1       2       3

                   7             5

                   4       8       6



                   1       2       3

                   7       5

                   4       8       6

Heuristic value : 4


                   1       2       3

                   7       5       6

                   4       8

Heuristic value : 2
```