Lab 2

Problem Statement: Simple Hill Climbing

NAME: Harshita Bhagat ROLLNO: 31

CLASS: TY - IT A BATCH: 2

Code:

```
package AI;


import java.util.Scanner;


public class SimpleHillClimbing {
   static int goal[][] = new int[3][3];
   static int arr[][] = new int[4][2];
   static int temp[][] = new int[3][3];
   static int mov;
   static float ans = 100;


   static float evaluate(int goal[][], int current[][]) {
       int dist = 0;
       for (int i = 0; i < 3; i++) {
           for (int j = 0; j < 3; j++) {
               dist += Math.pow(goal[i][j] - current[i][j], 2);
           }
       }
```

```
        float hval = (float) Math.sqrt(dist);
        dist = 0;
        return hval;
    }

    static void drawMatrix(int[][] current, int x, int y, int p,
int q) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (i == x && j == y) {
                    temp[i][j] = current[p][q];
                } else if (i == p && j == q) {
                    temp[i][j] = 0;
                } else {
                    temp[i][j] = current[i][j];
                }
            }
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter goal board state: ");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

```
                goal[i][j] = sc.nextInt();
            }
        }
        System.out.print("Enter initial board state: ");
        int current[][] = new int[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                current[i][j] = sc.nextInt();
            }
        }
        ans = evaluate(goal, current);
        System.out.println("Initial Hvalue: " + ans);
        if (ans == 0) {
            System.out.println("Initial state is the goal
state.");
        } else {
            // boolean IsGoal = false;
            // while (!IsGoal) {
            int x = -1, y = -1;
            for (int i = 0; i < 3; i++) {
                for (int j = 0; j < 3; j++) {
                    if (current[i][j] == 0) {
                        x = i;
                        y = j;
                    }
```

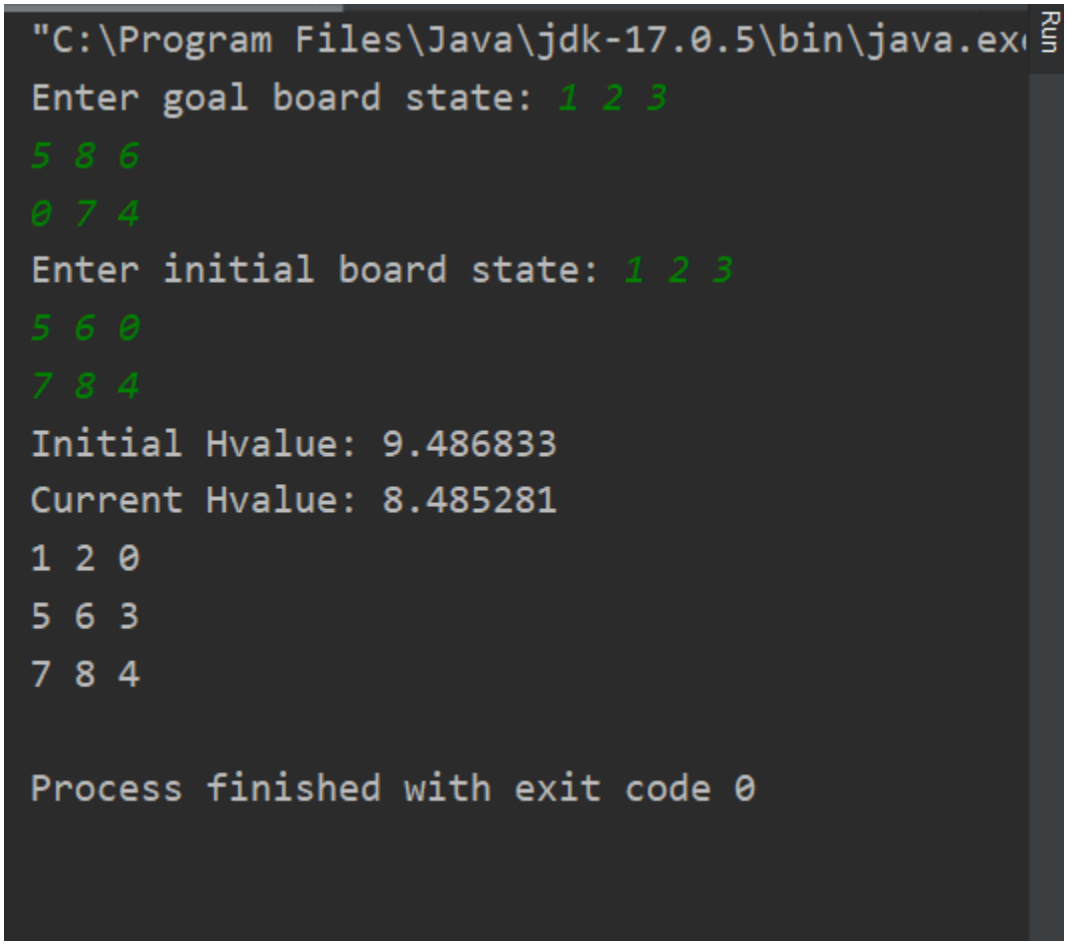
```
    }
}
mov = findIndex(x, y, arr);
float bestHval = ans;
int[][] betterState = current;
for (int i = 0; i < mov; i++) {
    drawMatrix(current, x, y, arr[i][0], arr[i][1]);
    float tans = evaluate(goal, temp);
    if (tans < bestHval) {
        bestHval = tans;
        betterState = temp;
        break;
    }
}
if (bestHval >= ans) {
    System.out.println("No operater left");
    // break;
}
current = betterState;
ans = bestHval;
System.out.println("Current Hvalue: " + ans);
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        System.out.print(current[i][j] + " ");
    }
```

```
            System.out.println();
        }
        if (ans == 0) {
            System.out.println("Goal state reached.");
            // break;
        }
    }
    // }
}

static int findIndex(int i, int j, int arr[][]) {
    int k = 0, cnt = 0;
    if ((3 > (i - 1) && i - 1 >= 0) && (3 > j && j >= 0)) {
        arr[k][0] = i - 1;
        arr[k][1] = j;
        k++;
        cnt++;
    }
    if (3 > i + 1 && i + 1 >= 0 && 3 > j && j >= 0) {
        arr[k][0] = i + 1;
        arr[k][1] = j;
        k++;
        cnt++;
    }
    if ((3 > i && i >= 0) && (3 > (j + 1) && j + 1 >= 0)) {
```

```
            arr[k][0] = i;
            arr[k][1] = j + 1;
            k++;
            cnt++;
        }
        if ((3 > i && i >= 0) && (3 > j - 1 && j - 1 >= 0)) {
            arr[k][0] = i;
            arr[k][1] = j - 1;
            k++;
            cnt++;
        }
        mov = cnt;
        return mov;
    }
}
```

Output:


"C:\Program Files\Java\jdk-17.0.5\bin\java.ex
Run
Enter goal board state: 1 2 3
5 8 6
0 7 4
Enter initial board state: 1 2 3
5 6 0
7 8 4
Initial Hvalue: 9.486833
Current Hvalue: 8.485281
1 2 0
5 6 3
7 8 4
Process finished with exit code 0