|  |  |
| --- | --- |
| **Experiment No.** | **8** |
| **Aim** | **Experiment on branch and bound** |
| **Name** | **Rucha Sudhir Kulkarni** |
| **UID No.** | **2021300067** |
| **Class & Division** | **SE Computer Engineering (Div:A)(Batch:D)** |
| **Date of Performance:** | **19.04.2023** |
| **Date of Submission:** | **25.04.2023** |

**Aim: – To implement 15 puzzle problem.**

**Algorithm:**

1. **Start**
2. **Input the initial matrix a and goal matrix t from the user.**
3. **Display the initial and goal state matrices.**
4. **While (a!=t)**
   1. **Increment move.**
   2. **Set d to 1000.**
   3. **Find the location (x, y) of the blank space in the initial matrix.**
   4. **Create a copy of the initial matrix in temp.**
   5. **If(any move possible)**
   6. **Do the respective move**
   7. **Calculate cost of that move**
   8. **If it is less than previous cost, then store the minimum as cost and display it.**
   9. **Display the intermediate matrix value r.**
   10. **Copy the contents of r to a and clear the contents of temp.**
5. **Stop.**

**Program:**

#include <stdio.h>

int m = 0, n = 4;

char mov='a';

int check\_up(int x,int y){

   if(x==0)

   return 0;

   return 1;

}

int check\_down(int x,int y){

   if(x==3)

   return 0;

   return 1;

}

int check\_right(int x,int y){

   if(y==3)

   return 0;

   return 1;

}

int check\_left(int x,int y){

   if(y==0)

   return 0;

   return 1;

}

int cal(int temp[10][10], int t[10][10])

{

    int i, j, m = 0;

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

        {

            if (temp[i][j] != t[i][j])

                m++;

        }

    return m;

}

int unequal(int a[10][10], int t[10][10])

{

    int i, j, f = 1;

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

            if (a[i][j] != t[i][j])

                f = 0;

    return f;

}

void main()

{

    int p, i, j, n = 4, a[10][10], t[10][10], temp[10][10], r[10][10];

    int m = 0, x = 0, y = 0, d = 1000, dmin = 0, l = 0;

    printf("\nEnter the matrix to be solved,space with zero :\n");

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

            scanf("%d", &a[i][j]);

    printf("\nEnter the goal matrix,space with zero :\n");

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

            scanf("%d", &t[i][j]);

    printf("\nEntered Matrix is :\n");

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

            printf("%d\t", a[i][j]);

        printf("\n");

    }

    printf("\nTarget Matrix is :\n");

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

            printf("%d\t", t[i][j]);

        printf("\n");

    }

int move=0;

    while (!(unequal(a, t)))

    {

        move++;

        d = 1000;

        for (i = 0; i < n; i++)

            for (j = 0; j < n; j++)

            {

                if (a[i][j] == 0)

                {

                    x = i;

                    y = j;

                }

            }

        // To move upwards

        if(check\_up(x,y)==1)

        {

            for (i = 0; i < n; i++)

            for (j = 0; j < n; j++)

                temp[i][j] = a[i][j];

            p = temp[x][y];

            temp[x][y] = temp[x - 1][y];

            temp[x - 1][y] = p;

        m = cal(temp, t)-1;

        dmin = move + m;

        if (dmin < d)

        {

            d = dmin;

            for (i = 0; i < n; i++)

                for (j = 0; j < n; j++)

                    r[i][j] = temp[i][j];

                    mov='u';

                    // printf("\n m: for %c is %d\n",mov,m);

        }

        }

        // To move downwards

        if(check\_down(x,y)==1){

        for (i = 0; i < n; i++)

            for (j = 0; j < n; j++)

                temp[i][j] = a[i][j];

            p = temp[x][y];

            temp[x][y] = temp[x + 1][y];

            temp[x + 1][y] = p;

        m = cal(temp, t)-1;

        dmin = move + m;

        if (dmin < d)

        {

            d = dmin;

            mov='d';

            for (i = 0; i < n; i++)

                for (j = 0; j < n; j++)

                    r[i][j] = temp[i][j];

                    // printf("\n m: for %c is %d\n",mov,m);

        }

        }

        // To move right side

        if(check\_right(x,y)==1)

        {

           for (i = 0; i < n; i++)

            for (j = 0; j < n; j++)

                temp[i][j] = a[i][j];

            p = temp[x][y];

            temp[x][y] = temp[x][y + 1];

            temp[x][y + 1] = p;

        m = cal(temp, t)-1;

        dmin = move + m;

        if (dmin < d)

        {

            d = dmin;

            mov='r';

            for (i = 0; i < n; i++)

                for (j = 0; j < n; j++)

                    r[i][j] = temp[i][j];

        // printf("\n m: for %c is %d\n",mov,m);

        }

        }

        // To move left

        if(check\_left(x,y)==1){

           for (i = 0; i < n; i++)

            for (j = 0; j < n; j++)

                temp[i][j] = a[i][j];

        if (y != 0)

        {

            p = temp[x][y];

            temp[x][y] = temp[x][y - 1];

            temp[x][y - 1] = p;

        }

        m = cal(temp, t)-1;

        dmin = move + m;

        if (dmin < d)

        {

            d = dmin;

            mov='l';

            for (i = 0; i < n; i++)

                for (j = 0; j < n; j++)

                    r[i][j] = temp[i][j];

                    // printf("\n m: for %c is %d\n",mov,m);

        }

        }

        printf("\nAfter %c move %d :\n",mov,move);

        for (i = 0; i < n; i++)

        {

            for (j = 0; j < n; j++)

                printf("%d\t", r[i][j]);

            printf("\n");

        }

        for (i = 0; i < n; i++)

            for (j = 0; j < n; j++)

            {

                a[i][j] = r[i][j];

                temp[i][j] = 0;

            }

       if(unequal(a,t)==1)

        printf("Cost : %d\n", d+1);

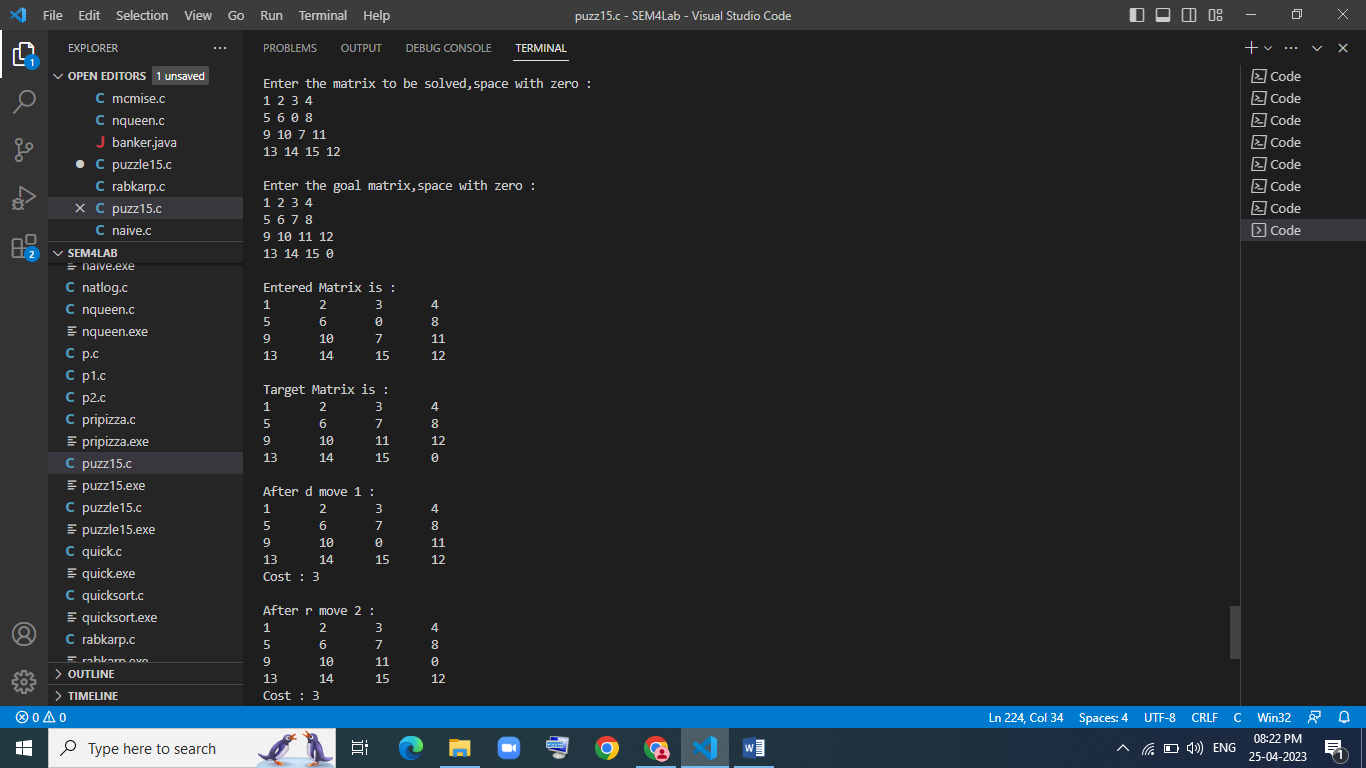
        else

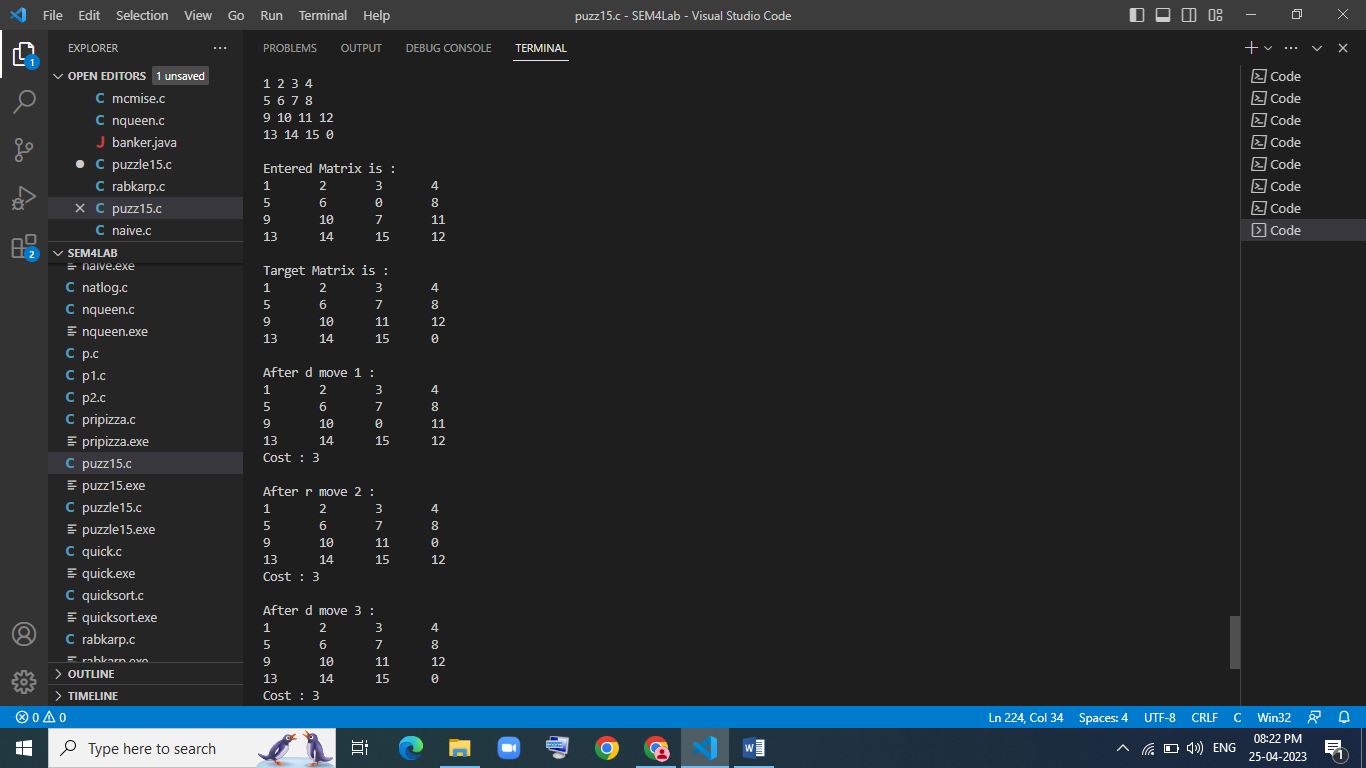
        printf("Cost : %d\n", d);

    }

}

**Output and Observation:**

****



**Conclusion:**

After performing the above experiment, I got to know the branch and bound strategy and how to implement 15 puzzle problem solution using it.