

NAME:	Shubham Solanki
UID:	2022301015
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT NO:	08
AIM:	To implement Branch and bound strategy
ALGORITHM:	<p>Branch and Bound Strategy Algorithm</p> <ol style="list-style-type: none"> 1. Declare and initialize the necessary variables including matrices a, t, temp, and r. 2. Read the input matrix a and target matrix t from the user. 3. Check if the input matrix a and the target matrix t are the same using the check() function. If they are the same, exit the loop. Otherwise, proceed to the next step. 4. Find the position of the zero element in the input matrix a. 5. Create a temporary matrix temp by copying the input matrix a. 6. Move the zero element in the up direction and calculate the cost by calling the cal() function. 7. If the calculated cost is less than the current minimum cost d, update the minimum cost d and copy the current matrix configuration to the matrix r. 8. Repeat step 6 and 7 for moving the zero element in the down, right, and left directions. 9. Copy the updated matrix r to the input matrix a. 10. Repeat step 3 to 9 until the input matrix a becomes equal to the target matrix t. 11. Print the final minimum cost.
CODE	<p>Source Code</p> <pre>#include<stdio.h> #include<conio.h> int m=0,n=4; int cal(int temp[10][10],int t[10][10]) { int i,j,m=0; for(i=0;i < n;i++)</pre>

```

        for(j=0;j < n;j++)
        {
            if(temp[i][j]!=t[i][j])
                m++;
        }
        return m;
    }

int check(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 target 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");
    }

    while(!(check(a,t)))
    {
        l++;
        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
        for(i=0;i < n;i++)
            for(j=0;j < n;j++)
                temp[i][j]=a[i][j];

        if(x!=0)
        {
            p=temp[x][y];
            temp[x][y]=temp[x-1][y];
            temp[x-1][y]=p;
        }
        m=cal(temp,t);
        dmin=l+m;
        if(dmin < d)
        {
            d=dmin;
            for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                    r[i][j]=temp[i][j];

```

```

    }

    //To move downwards
    for(i=0;i < n;i++)
        for(j=0;j < n;j++)
            temp[i][j]=a[i][j];
    if(x!=n-1)
    {
        p=temp[x][y];
        temp[x][y]=temp[x+1][y];

        temp[x+1][y]=p;
    }
    m=cal(temp,t);
    dmin=l+m;
    if(dmin < d)
    {
        d=dmin;

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

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

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

    //To move right side

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

        for(j=0;j < n;j++)
            temp[i][j]=a[i][j];
    if(y!=n-1)
    {
        p=temp[x][y];
        temp[x][y]=temp[x][y+1];
        temp[x][y+1]=p;
    }
    m=cal(temp,t);

    dmin=l+m;

    if(dmin < d)

```

```

{
    d=dmin;
    for(i=0;i < n;i++)
        for(j=0;j < n;j++)
            r[i][j]=temp[i][j];
}

//To move left

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);
dmin=l+m;
if(dmin < d)
{
    d=dmin;
    for(i=0;i < n;i++)
        for(j=0;j < n;j++)
            r[i][j]=temp[i][j];
}

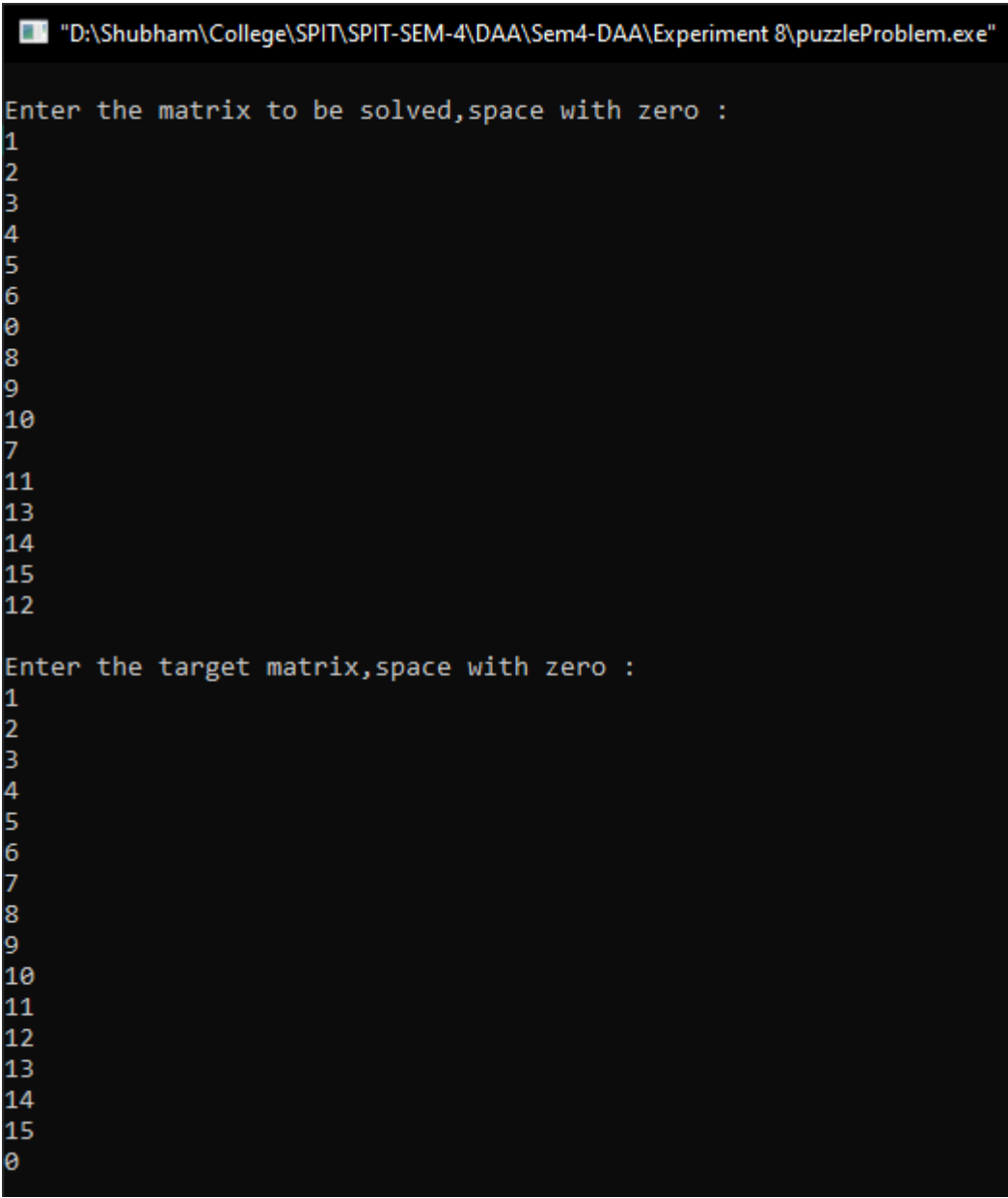
printf("\nCalculated Intermediate Matrix Value :\n");

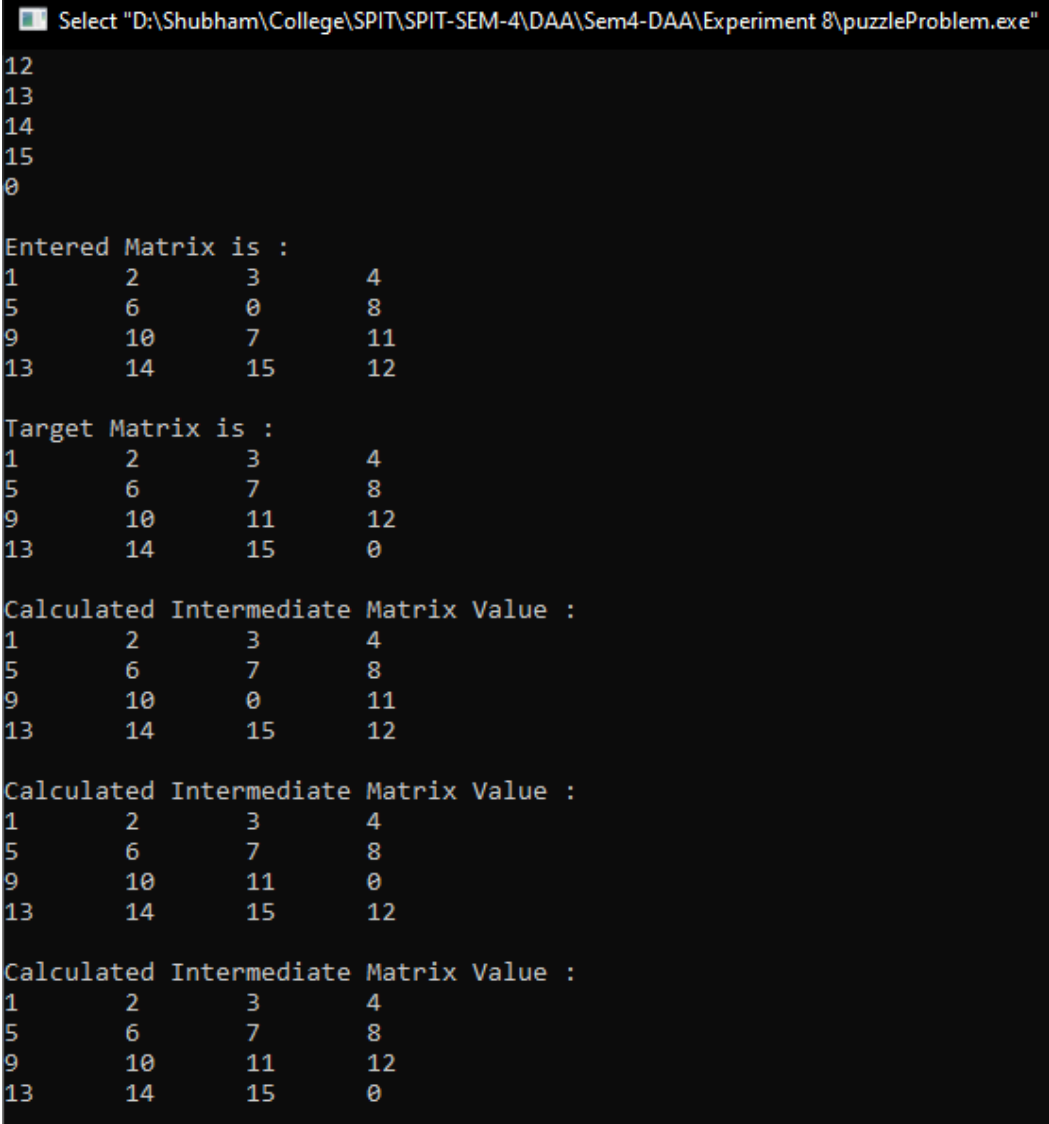
for(i=0;i < n;i++)
{
    for(j=0;j < n;j++)

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

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

```

	<pre> a[i][j]=r[i][j]; temp[i][j]=0; } } getch(); } </pre>
Output	 <p>The screenshot shows a Windows command prompt window titled "D:\Shubham\College\SPIT\SPIT-SEM-4\DAA\Sem4-DAA\Experiment 8\puzzleProblem.exe". The program prompts the user to "Enter the matrix to be solved,space with zero :". The user has entered a 16x16 matrix of numbers. The prompt then asks to "Enter the target matrix,space with zero :". The user has entered a 16x16 matrix of numbers, with the last row being all zeros.</p> <pre> Enter the matrix to be solved,space with zero : 1 2 3 4 5 6 0 8 9 10 7 11 13 14 15 12 Enter the target matrix,space with zero : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 </pre>

	 <pre> Select "D:\Shubham\College\SPIT\SPIT-SEM-4\DAA\Sem4-DAA\Experiment 8\puzzleProblem.exe" 12 13 14 15 0 Entered Matrix is : 1 2 3 4 5 6 0 8 9 10 7 11 13 14 15 12 Target Matrix is : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 Calculated Intermediate Matrix Value : 1 2 3 4 5 6 7 8 9 10 0 11 13 14 15 12 Calculated Intermediate Matrix Value : 1 2 3 4 5 6 7 8 9 10 11 0 13 14 15 12 Calculated Intermediate Matrix Value : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 </pre>
CONCLUSION	Thus we have implemented branch and bound strategy and we have solved the 15 puzzle problem using branch and bound strategy