|  |  |
| --- | --- |
| **NAME:** | Vaishnavi Bhagawan Borkar |
| **UID:** | 2021300016 |
| **SUBJECT** | Data Analysis and Algorithm |
| **EXPERIMENT NO:** | Experiment 8 |
| **DATE OF PERFORMANCE** | 10/04/23 |
| **AIM:** | Experiment using branch and bound strategy: To find the 15-puzzle solution. |
| **THEORY:** |  |
| **ALGORITHM:** |  |
| **PROGRAM:** | *#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++)  *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++)              {                  a[i][j] = r[i][j];                  temp[i][j] = 0;              }          printf("Minimum cost : %d\n", d);      }      getch();  } |
| **RESULT:** | |
| **CONCLUSION:** | By performing the above experiment, I was able to implement the 15-box puzzle problem to print the solved puzzle solution by moving the empty spaces in four different direction and finding out minimum cost. |