Vashisth Zatakia **60003200163(B3)**

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

COURSE CODE: DJ19ITC403 **DATE: 2nd June 2022**
COURSE NAME: Design and Analysis of Algorithm **CLASS: S.Y.B.Tech(Sem IV)**

Experiment 10

AIM/OBJECTIVE: To implement 15 puzzle problem using Branch & Bound

Theory:

Given a 4×4 board with 15 tiles (every tile has one number from 1 to 15) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space.

Branch and Bound

The search for an answer node can often be speeded by using an "intelligent" ranking function, also called an approximate cost function to avoid searching in sub-trees that do not contain an answer node. It is similar to the backtracking technique but uses a BFS-like search.

There are basically three types of nodes involved in Branch and Bound

1. Live node is a node that has been generated but whose children have not yet been generated.
2. E-node is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.
3. Dead node is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.

Cost function:

Each node X in the search tree is associated with a cost. The cost function is useful for determining the next E-node. The next E-node is the one with the least cost. The cost function is defined as

$C(X) = g(X) + h(X)$ where

$g(X)$ = cost of reaching the current node from the root

$h(X)$ = cost of reaching an answer node from X.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Code:

```
#include <stdio.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;
}

int 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);
   }
   return 0;
}
```

Output:

```
❯ clang -o program exp10.c
❯ ./program

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

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
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      0       11
13      14      15      12
Minimum cost : 4

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      0
13      14      15      12
Minimum cost : 4

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      0
Minimum cost : 3
```

Conclusion: Thus, 15 Puzzle Problem is solved using Branch and Bound.