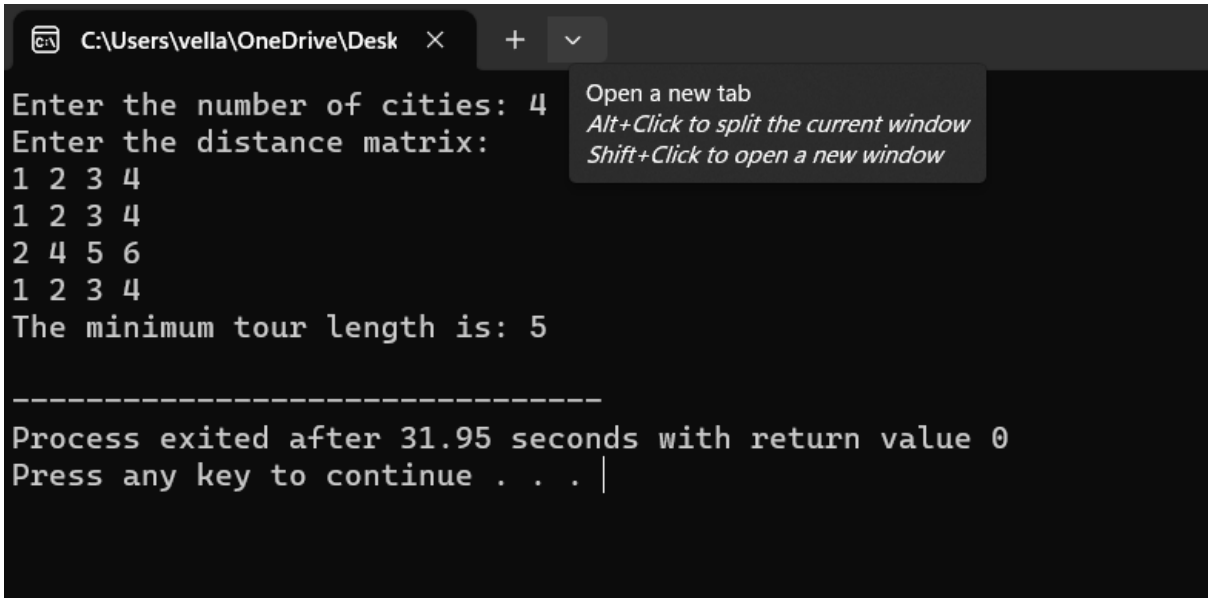
1).Determine an optimal tour in a weighted, directed graph. The weights are nonnegative numbers. The inputs are weighted, directed graph, and n, the number of vertices in the graph. The graph is represented by a twodimensional array W, which has both its rows and columns indexed from 1 to n, where W [i] [j] is the weight on the edge from the ith vertex to the jth vertex. Write a program for travelling salesman problem using dynamic programming for the below given graph.

Program :

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 20
#define INF 99999
int n, d[MAX][MAX], x[MAX];
int best_tour_length = INF, tour_length[MAX];
void backtrack(int curr_pos) {
 int i;
 if (curr_pos == n) {
 tour_length[curr_pos] = d[x[n - 1]][x[0]];
 int tour = 0;
 for (i = 0; i < n; i++) tour += tour_length[i];
 if (tour < best_tour_length) best_tour_length = tour;
 return;
 }
 for (i = 0; i < n; i++) {
 if (x[i] == -1) {
 x[i] = curr_pos;
 tour_length[curr_pos] = d[x[curr_pos - 1]][i];
 backtrack(curr_pos + 1);
 x[i] = -1;
 }
 }
}
int main() {
```

```
int i, j;
printf("Enter the number of cities: ");
scanf("%d", &n);
printf("Enter the distance matrix:\n");
for (i = 0; i < n; i++)
for (j = 0; j < n; j++) {
scanf("%d", &d[i][j]);
x[i] = -1;
}
x[0] = 0;
backtrack(1);
printf("The minimum tour length is: %d\n", best_tour_length);
return 0;
}
```



2) The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order. Write a program for the same.

PROGRAM:

```
#include <stdio.h>
#include <stdbool.h>
#define N 8
int col[N];
```

```
bool check(int row) {
 int i;
 for (i = 0; i < row; i++)
 if (col[i] == col[row] ||
 row - i == col[row] - col[i] ||
 row - i == col[i] - col[row])
 return false;
 return true;
}
void backtrack(int row) {
 int i;
 if (row == N) {
 for (i = 0; i < N; i++) printf("(%d, %d)\n", i, col[i]);
 printf("\n");
 return;
 }
 for (i = 0; i < N; i++) {
 col[row] = i;
 if (check(row)) backtrack(row + 1);
 }
}
int main() {
 backtrack(0);
 return 0;
}
```

```
C:\Users\vella\OneDrive\Desk

(1, 1)
(2, 4)
(3, 2)
(4, 0)
(5, 6)
(6, 3)
(7, 5)

(0, 7)
(1, 2)
(2, 0)
(3, 5)
(4, 1)
(5, 4)
(6, 6)
(7, 3)

(0, 7)
(1, 3)
(2, 0)
(3, 2)
(4, 5)
(5, 1)
(6, 6)
(7, 4)
```

3) Writa a C program for binary seach tree and find the time complexity

Program :

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node*left;

struct node*right;

}*root=NULL,*newnode;

struct node*create(struct node*root,int ele)
```

```
{
if(root==NULL)
{
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=ele;
newnode->left=NULL;
newnode->right=NULL;
return(newnode);
}
else if(ele>root->data)
root->right=create(root->right,ele);
else if(ele<root->data)
root->left=create(root->left,ele);
return(root);
}
void inorder(struct node *root)
{
if(root!=NULL)
{
inorder(root->left);
printf("%d\t",root->data);
inorder(root->right);
}
}
void preorder(struct node *root)
{
if(root!=NULL)
{
printf("%d\t",root->data);
preorder(root->left);
preorder(root->right);
}
```

```
}
void postorder(struct node *root)
{
if(root!=NULL)
{
postorder(root->left);
postorder(root->right);
printf("%d\t",root->data);
}
}
int main()
{
   int choice;
   while(1)
  {
   printf("\nMAIN MEANU\n");
  printf("\n1.CREATE\n");
  printf("\n2.INORDER\n");
  printf("\n3.PREORDER\n");
  printf("\n4.POSTORDER\n");
  printf("\n5.EXIT\n");
  printf("\nENTER THE CHOICE:\t");
  scanf("%d",&choice);
  switch(choice)
{
   case 1:
       int ele;
       printf("ENTER THE ELEMENT:");
       scanf("%d",&ele);
       root=create(root,ele);
       break;
   case 2:
```

```
      inorder(root);
      break;
   case 3:
      preorder(root);
      break;
   case 4:
      postorder(root);
      break;
   case 5:
      exit(0);
      break;
   default:
     printf("\nWRONG CHOICE\n");
     break;
    }
    }
}
```

4) Let there be N workers and N jobs. Any worker can be assigned to perform any job, incurring some cost that may vary depending on the work-job assignment. It is required to perform all jobs by assigning exactly one worker to each job and exactly one job to each agent in such a way that the total cost of the assignment is minimized. Write a program to solve a assignment problem for the given data sets using branch and bound.

	Job 1	Job 2	Job 3	Job 4
Person A	12	8	9	10
Person B	11	10	10	9
Person C	9	11	8	12
Person D	11	9	23	7

Program :

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
```

```
typedef struct Job {
char id;
int dead;
int profit;
} Job;
int compare(const void* a, const void* b)
{
Job* temp1 = (Job*)a;
Job* temp2 = (Job*)b;
return (temp2->profit - temp1->profit);
}
int min(int num1, int num2)
{
return (num1 > num2) ? num2 : num1;
}
void printJobScheduling(Job arr[], int n)
{
qsort(arr, n, sizeof(Job), compare);
int result[n];
bool slot[n];
for (int i = 0; i < n; i++)
slot[i] = false;
for (int i = 0; i < n; i++) {
for (int j = min(n, arr[i].dead) - 1; j >= 0; j--) {
if (slot[j] == false) {
result[j] = i;
slot[j] = true;
break;
}
}
}
for (int i = 0; i < n; i++)
```

```
if (slot[i])
printf("%c ", arr[result[i]].id);
}
int main()
{
Job arr[] = {
{ 'a', 12, 8, 9, 10 },
{ 'b', 11, 10, 10, 9 },
{ 'c', 9, 11, 8, 12 },
{ 'd', 11, 9, 23, 7 },

int n = sizeof(arr) / sizeof(arr[0]);
printf(
"Following is maximum profit sequence of jobs \n");
printJobScheduling(arr, n);
return 0;
}
```