

Practical-7

Aim: Implement Hamiltonian Cycle using Backtracking.

Problem Statement:

The Smart City Transportation Department is designing a night-patrol route for security vehicles.

Each area of the city is represented as a vertex in a graph, and a road between two

areas is represented as an edge.

The goal is to find a route that starts from the main headquarters (Area A), visits

each area exactly once, and returns back to the headquarters — forming a

Hamiltonian Cycle.

If such a route is not possible, display a suitable message.

1) Adjacency Matrix

A B C D E

A 0 1 1 0 1

B 1 0 1 1 0

C 1 1 0 1 0

D 0 1 1 0 1

E 1 0 0 1 0

Code:

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

```
#define V 5 // Number of vertices
```

```
// Function to check if the vertex v can be added at position 'pos' in  
the Hamiltonian Path
```

```
int isSafe(int v, int graph[V][V], int path[], int pos)
```

```

{
    // Step 1: Check if current vertex is adjacent to the previous vertex
    if (graph[path[pos - 1]][v] == 0)
        return 0;

    // Step 2: Check if vertex has already been included
    for (int i = 0; i < pos; i++)
        if (path[i] == v)
            return 0;

    return 1; // Safe to add
}

```

```

// Recursive function to find Hamiltonian Cycle
int hamiltonianCycleUtil(int graph[V][V], int path[], int pos)
{
    // Base Case: All vertices included
    if (pos == V)
    {
        // Check if last vertex connects to the first
        if (graph[path[pos - 1]][path[0]] == 1)
            return 1;
        else
            return 0;
    }
}

```

```

// Try different vertices as the next candidate
for (int v = 1; v < V; v++)
{
    if (isSafe(v, graph, path, pos))
    {
        path[pos] = v; // Add vertex to path
    }
}

```

```

    // Recur to build rest of path
    if (hamiltonianCycleUtil(graph, path, pos + 1) == 1)
        return 1;

    // Backtrack: remove vertex if it doesn't work
    path[pos] = -1;
}
}

// If no vertex can be added
return 0;
}

// Function to solve Hamiltonian Cycle problem
void hamiltonianCycle(int graph[V][V])
{
    int path[V];

    // Initialize all vertices as not visited
    for (int i = 0; i < V; i++)
        path[i] = -1;

    // Start at first vertex (T = 0)
    path[0] = 0;

    if (hamiltonianCycleUtil(graph, path, 1) == 0)
    {
        printf("No Hamiltonian Cycle exists\n");
        return;
    }

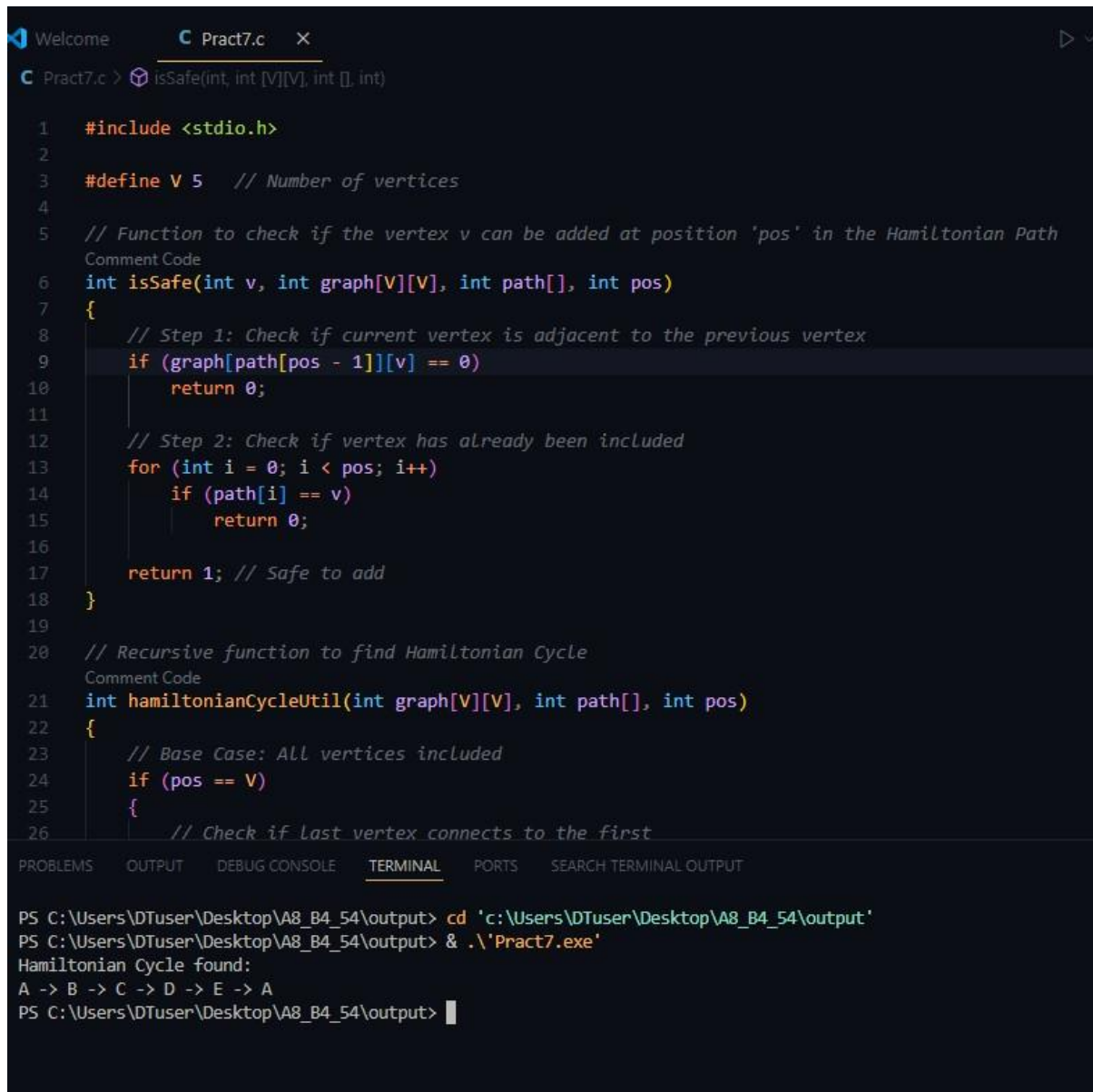
    // Print the Hamiltonian Cycle
    printf("Hamiltonian Cycle found:\n");

```

```
    for (int i = 0; i < V; i++)  
        printf("%c -> ", 'A' + path[i]); // Convert 0→T, 1→M, etc.  
    printf("%c\n", 'A' + path[0]);  
}
```

```
int main()  
{  
    // Adjacency matrix for A,B,C,D,E  
    int graph[V][V] = {  
        {0, 1, 1, 0, 1}, // A  
        {1, 0, 1, 1, 0}, // B  
        {1, 1, 0, 1, 0}, // C  
        {0, 1, 1, 0, 1}, // D  
        {1, 0, 0, 1, 0} // E  
    };  
  
    hamiltonianCycle(graph);  
  
    return 0;  
}
```

Output:



```
1  #include <stdio.h>
2
3  #define V 5    // Number of vertices
4
5  // Function to check if the vertex v can be added at position 'pos' in the Hamiltonian Path
6  // Comment Code
7  int isSafe(int v, int graph[V][V], int path[], int pos)
8  {
9      // Step 1: Check if current vertex is adjacent to the previous vertex
10     if (graph[path[pos - 1]][v] == 0)
11         return 0;
12
13     // Step 2: Check if vertex has already been included
14     for (int i = 0; i < pos; i++)
15         if (path[i] == v)
16             return 0;
17
18     return 1; // Safe to add
19 }
20
21 // Recursive function to find Hamiltonian Cycle
22 // Comment Code
23 int hamiltonianCycleUtil(int graph[V][V], int path[], int pos)
24 {
25     // Base Case: All vertices included
26     if (pos == V)
27     {
28         // Check if last vertex connects to the first
29         if (graph[path[pos - 1]][path[0]] == 1)
30             return 1;
31         return 0;
32     }
33
34     for (int v = 0; v < V; v++)
35     {
36         if (isSafe(v, graph, path, pos) && !path[v])
37         {
38             path[pos] = v;
39             if (hamiltonianCycleUtil(graph, path, pos + 1))
40                 return 1;
41             path[pos] = -1;
42         }
43     }
44
45     return 0;
46 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT

```
PS C:\Users\DTuser\Desktop\A8_B4_54\output> cd 'c:\Users\DTuser\Desktop\A8_B4_54\output'
PS C:\Users\DTuser\Desktop\A8_B4_54\output> & .\'Pract7.exe'
Hamiltonian Cycle found:
A -> B -> C -> D -> E -> A
PS C:\Users\DTuser\Desktop\A8_B4_54\output> |
```