

Name:Areeb Ansari

Roll No:A3-B1-01

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.

Code:

```
#include <stdio.h>

#define N 5

int G[N][N] = {
    {0, 1, 1, 0, 1},
    {1, 0, 1, 1, 0},
    {1, 1, 0, 1, 0},
    {0, 1, 1, 0, 1},
    {1, 0, 0, 1, 0}
};

int path[N];
int used[N] = {0};

void printCycle() {
    for (int i = 0; i < N; i++) {
        printf("%c ", 'A' + path[i]);
    }
    printf("%c\n", 'A' + path[0]);
}
```

```

void addToPath(int pos) {
    if (pos == N) {
        if (G[path[pos - 1]][path[0]] == 1) {
            printCycle();
        }
        return;
    }

    for (int v = 1; v < N; v++) {
        if (!used[v] && G[path[pos - 1]][v] == 1) {
            path[pos] = v;
            used[v] = 1;

            addToPath(pos + 1);

            used[v] = 0;
        }
    }
}

int main() {

    path[0] = 0;
    used[0] = 1;

    printf("All possible Hamiltonian cycles:\n");
    addToPath(1);
    return 0;
}

```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\areeb\OneDrive\Desktop\SemIII\OAA\pract 8> & 'c:\Users\areeb\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebuglauncher.exe' '--stdin-Microsoft-MIEngine-In\data\Skyk.bah' '--stdout-Microsoft-MIEngine-Out\jmaxisue.doc' '--stderr-Microsoft-MIEngine-Error-out\k2vsa.cdx' '--pid-Microsoft-MIEngine-Pid-wr\mnm4.nds' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
All possible Hamiltonian cycles:
A B C D E A
A C B D E A
A E D B C A
A E D C B A
PS C:\Users\areeb\OneDrive\Desktop\SemIII\OAA\pract 8>

```

GFG problems

The screenshot displays the GeeksforGeeks (GFG) website interface for a problem-solving session. The browser address bar shows the URL `geeksforgeeks.org/problems/hundunian-paths2542/1`. The page features a navigation bar with links for 'Courses', 'Tutorials', 'Practice', and 'Solve'. The main content area is divided into two panels. The left panel, titled 'Output Window', shows 'Compilation Results' with a green checkmark indicating 'Problem Solved Successfully'. It displays 'Test Cases Passed: 52 / 52', 'Attempts: Correct / Total: 3 / 3', and 'Accuracy: 100%'. The 'Time Taken' is listed as '0.03'. A red warning message states: 'You get marks only for the first correct submission if you solve the problem without viewing the full solution.' Below this, the 'Solve Next' section lists 'Number of Provinces', 'Number of Distinct Islands', and 'Number of Good Components'. The right panel shows a Python code editor with a solution for a graph problem. The code defines a `check` function to build an adjacency list and a `dfs` function to traverse the graph and count components. The `main` function iterates over all vertices to find the total number of components.

```
1 # Python3 template for gfg
2 class Solution:
3     def check(self, n, m, edges):
4
5         graph = [[] for _ in range(n + 1)]
6         for u, v in edges:
7             graph[u].append(v)
8             graph[v].append(u)
9
10        visited = [False] * (n + 1)
11
12
13        def dfs(vertex, count):
14            if count == n:
15                return True
16
17            for neighbor in graph[vertex]:
18                if not visited[neighbor]:
19                    visited[neighbor] = True
20                    if dfs(neighbor, count + 1):
21                        return True
22                    visited[neighbor] = False
23            return False
24
25        for start in range(1, n + 1):
26            visited[start] = True
27            if dfs(start, 1):
28                return 1
29            visited[start] = False
30
31        return 0
```