

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;
        }
    }
}

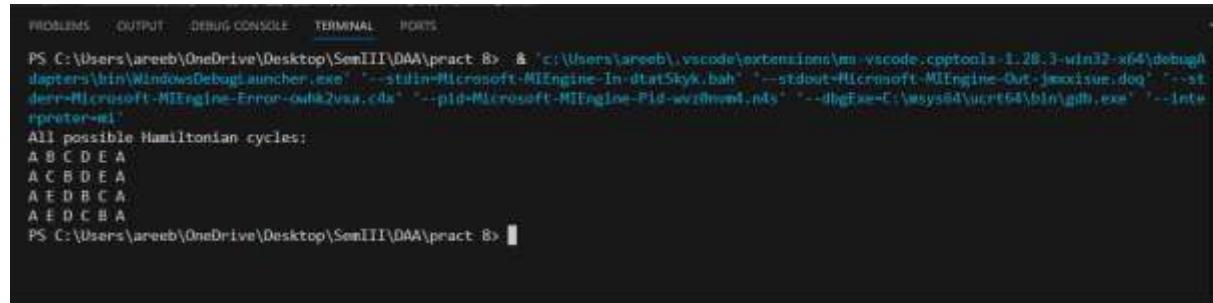
int main() {

    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\DA\pract_8> & "c:\Users\areeb\vscode\extensions\ms-vscode\cpptools-1.28.3-win32-x64\debug\scripts\MinWinDebugLauncher.exe" "--stdin=Microsoft-MIEngine-In\statSkyk.bah" "--stdout=Microsoft-MIEngine-Out\execIssue.dog" "--stderr=Microsoft-MIEngine-Error\out\err2USA.dll" "--pid=Microsoft-MIEngine-Pid\wzI8mmt.nls" "--digiFile=C:\wsys64\user64\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\DA\pract_8>

```

# GFG problems

The screenshot shows a browser window for the GeeksforGeeks Practice platform. The URL is [geeksforgeeks.org/problems/minimun-path-practice/1](https://www.geeksforgeeks.org/problems/minimun-path-practice/1). The page displays a solved problem with the following details:

- Output Window:** Shows the status "Problem Solved Successfully".
- Compilation Results:** Current input: "TC.GF", Current output: "TC.GF", Status: "Accepted".
- Attempts:** Correct / Total: 3 / 3, Accuracy: 100%.
- Time Taken:** 0.03.
- Note:** You get points only for this first correct submission if you solve the problem without viewing the full solution.
- Solve Next:** Buttons for "Number of Provinces", "Number of Distinct Islands", and "Number of Good Components".
- Code:** Python code for the solution:

```
Python -> Start Timer (0)
1: #user function template for python
2: class Solution:
3:     def check(self, n, m, edges):
4:
5:         graph = []
6:         for _ in range(n + 1):
7:             for u, v in edges:
8:                 graph[u].append(v)
9:                 graph[v].append(u)
10:
11:         visited = [False] * (n + 1)
12:
13:         def dfs(vertex, count):
14:
15:             if count == m:
16:                 return True
17:
18:             for neighbor in graph[vertex]:
19:                 if visited[neighbor] == False:
20:                     visited[neighbor] = True
21:                     if dfs(neighbor, count + 1):
22:                         return True
23:                     visited[neighbor] = False
24:
25:
26:         for start in range(1, n + 1):
27:             visited[start] = True
28:             if dfs(start, 1):
29:                 return 1
30:             visited[start] = False
31:
32:
33:
34:
35:
36:
37:
38:
39:
39: return 0
```