|  |  |
| --- | --- |
| **Experiment-3.2** | |
| **Student Name: Shubham Kumar** | **UID: 21BCS7104** |
| **Branch: CSE** | **Section/Group: 614-A** |
| **Semester: 6th** | **Date of Performance: 03-04-2024** |
| **Subject Name: Advanced Programming Lab-2** | **Subject Code: 21CSP-351** |

**1. Aim:** Backtracking.

A) Given an integer turnedOn which represents the number of LEDs that are currently on (ignoring the PM), return all possible times the watch could represent. You may return the answer in any order.(401. Binary Watch)

B) Given a directed acyclic graph (DAG) of n nodes labeled from 0 to n - 1, find all possible paths from node 0 to node n - 1 and return them in any order.( 797 - All Paths From Source to Target)

**2. Source Code/Output:**

**A) Code:**

class Solution {

public:

vector<string> readBinaryWatch(int turnedOn) {

vector<string> ans;

for (int i = 0; i < 12; ++i) {

for (int j = 0; j < 60; ++j) {

if (\_\_builtin\_popcount(i) + \_\_builtin\_popcount(j) == turnedOn) {

ans.push\_back(to\_string(i) + ":" + (j < 10 ? "0" : "") + to\_string(j));

}

}

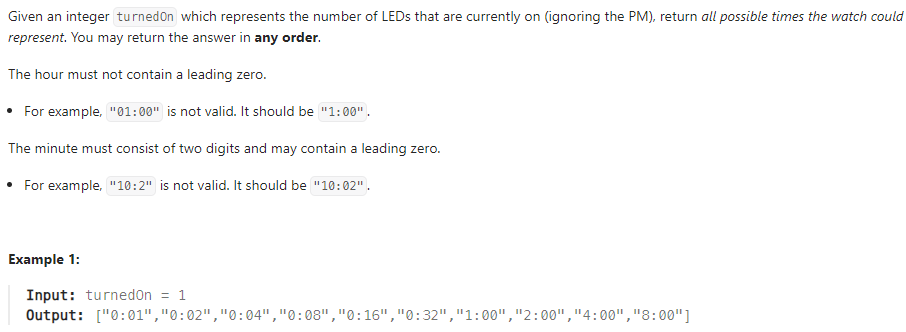
}

return ans;

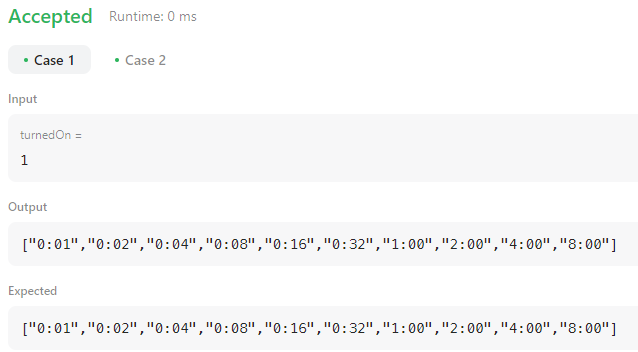
}

};

**Description:**

****

**Output:**



**B)** **Code:**

class Solution {

public:

vector<vector<int>> graph;

vector<vector<int>> ans;

vector<vector<int>> allPathsSourceTarget(vector<vector<int>>& graph) {

this->graph = graph;

vector<int> path;

path.push\_back(0);

dfs(0, path);

return ans;

}

void dfs(int i, vector<int> path) {

if (i == graph.size() - 1) {

ans.push\_back(path);

return;

}

for (int j : graph[i]) {

path.push\_back(j);

dfs(j, path);

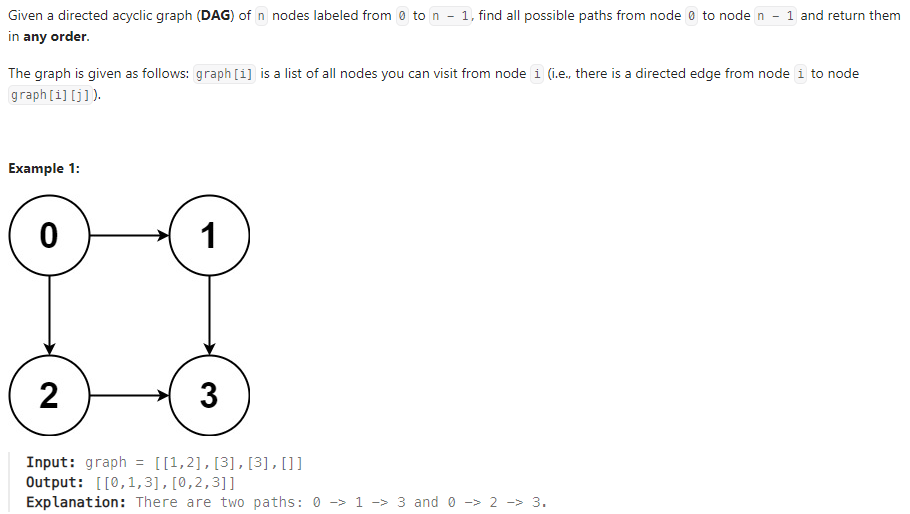
path.pop\_back();

}

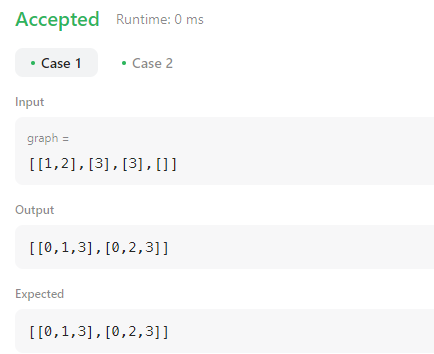
}

};

**Description:**



**Output:**

****