|  |  |
| --- | --- |
| **Experiment-2.2** | |
| **Student Name: Shubham Kumar** | **UID: 21BCS7104** |
| **Branch: CSE** | **Section/Group: 614-A** |
| **Semester: 6th** | **Date of Performance: 28-02-2024** |
| **Subject Name: Advanced Programming Lab-2** | **Subject Code: 21CSP-351** |

**1. Aim:** Graph.

A) A graph is bipartite if the nodes can be partitioned into two independent sets A and B such that every edge in the graph connects a node in set A and a node in set B.

Return true if and only if it is bipartite.(785. Is Graph Bipartite?)

B) Given an integer n, return any valid n-bit gray code sequence..(89. Gray Code)

**2. Source Code/Output:**

**A) Code:**

class Solution {

public:

bool isBipartite(vector<vector<int>>& graph) {

int len = graph.size();

stack<int> s;

vector<int> vis(len);

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

if (vis[i] > 0) continue;

vis[i] = 1;

s.push(i);

while (s.size() > 0) {

int curr = s.top();

s.pop();

vector<int> edges = graph[curr];

for (int next:edges)

if (vis[next] == 0) {

vis[next] = vis[curr] ^ 3;

s.push(next);

} else if (vis[curr] == vis[next]) return false;

}

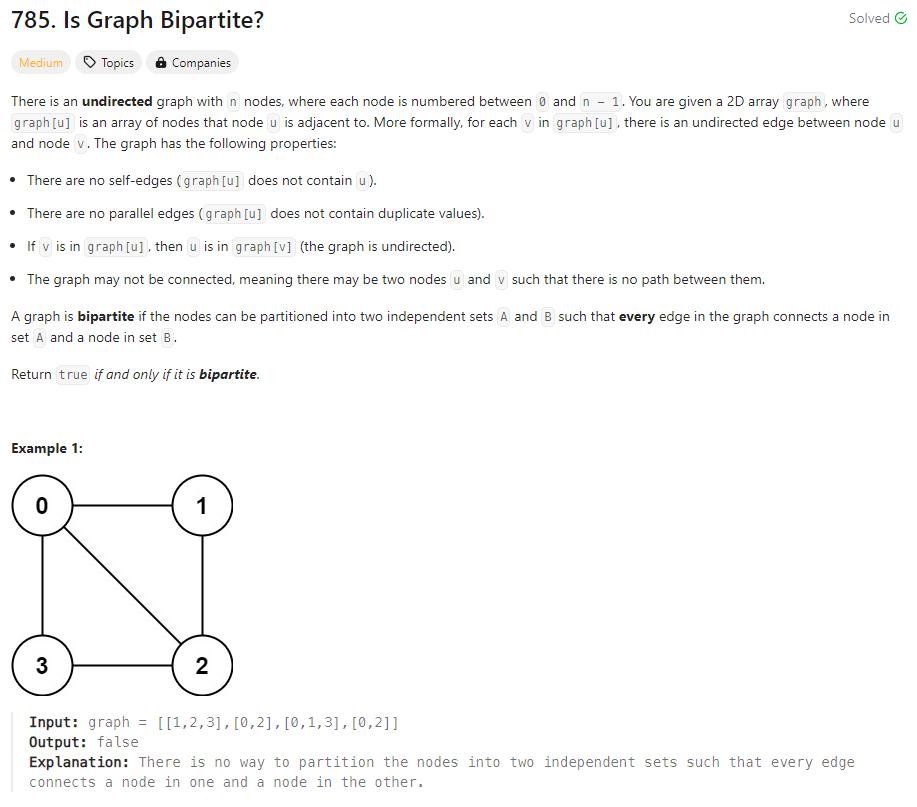
}

return true;

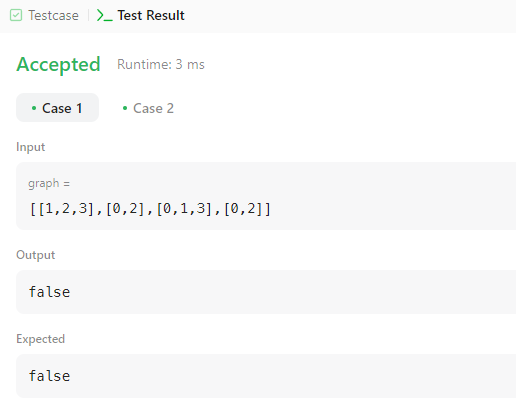
}

};

**Description:**

****

**Output:**



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

class Solution {

public:

vector<int> grayCode(int n) {

vector<int> ans{0};

for (int i = 0; i < n; ++i)

for (int j = ans.size() - 1; j >= 0; --j)

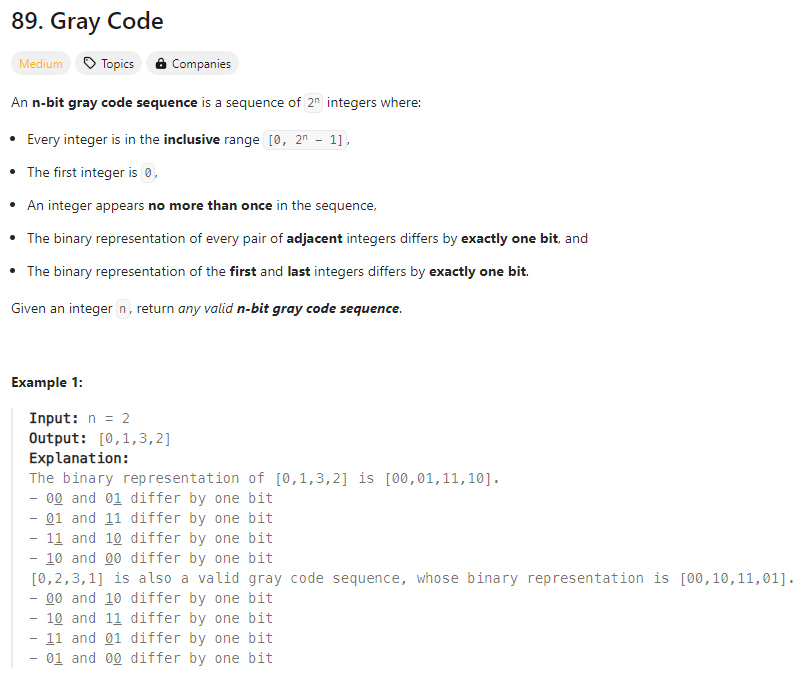
ans.push\_back(ans[j] | 1 << i);

return ans;

}

};

**Description:**



**Output:**

****