89. Gray Code

Add to List

Description (?tab=Description) | Submission (?tab=Submission) | Solutions (?tab=Solutions)

Total Accepted: **81504** Total Submissions: **205144** Difficulty: **Medium** Contributors: **Admin**

Notes

The gray code is a binary numeral system where two successive values differ in only one bit.

Given a non-negative integer *n* representing the total number of bits in the code, print the sequence of gray code. A gray code sequence must begin with 0.

For example, given *n* = 2, return `[0,1,3,2]`. Its gray code sequence is:

```
00 - 0
01 - 1
11 - 3
10 - 2
```

Note:

For a given *n*, a gray code sequence is not uniquely defined.

For example, `[0,2,3,1]` is also a valid gray code sequence according to the above definition.

For now, the judge is able to judge based on one instance of gray code sequence. Sorry about that.

Hide Company Tags | Amazon (/company/amazon/)

Hide Tags | Backtracking (/tag/backtracking/)

Have you met this question in a real interview? Yes No

Discuss (https://discuss.leetcode.com/category/97) | Pick One (/problems/random-one-question/) | Editorial Solution

C++

```cpp
class Solution {
public:

    bool dfs(int num, int n, int total, vector<int>& path, set<int>& visited) {
        if(path.size() == total) {
            return true;
        }
        for(int i = 0; i<n; ++i) {
            int newnum;
            if(num & (1<<i)) {
                newnum = num ^ (1<<i);
            } else {
                newnum = num | (1<<i);
            }
            if(visited.find(newnum) == visited.end()) {
                visited.insert(newnum);
                path.push_back(newnum);
                if(dfs(newnum, n, total, path, visited))
                    return true;
                path.pop_back();
                visited.erase(newnum);
            }
        }
        return false;
    }

    vector<int> grayCode(int n) {
        vector<int> path;
        set<int> visited;
        int total = pow(2, n);
        path.push_back(0);
        visited.insert(0);
        dfs(0, n, total, path, visited);
        return path;
    }
```

Custom Testcase

Contribute Testcase

Run Code | Submit Solution

Notes