Experiment 2.2

Student Name: Yash Gupta **UID: 20BCS5009**
Branch: BE-CSE **Section/Group:20BCS_DM-716 B**
Semester: 6 **Date of Performance: 03/04/23**
Subject Name: CC LAB **Subject Code: 20CSP_351**

1. Aim:

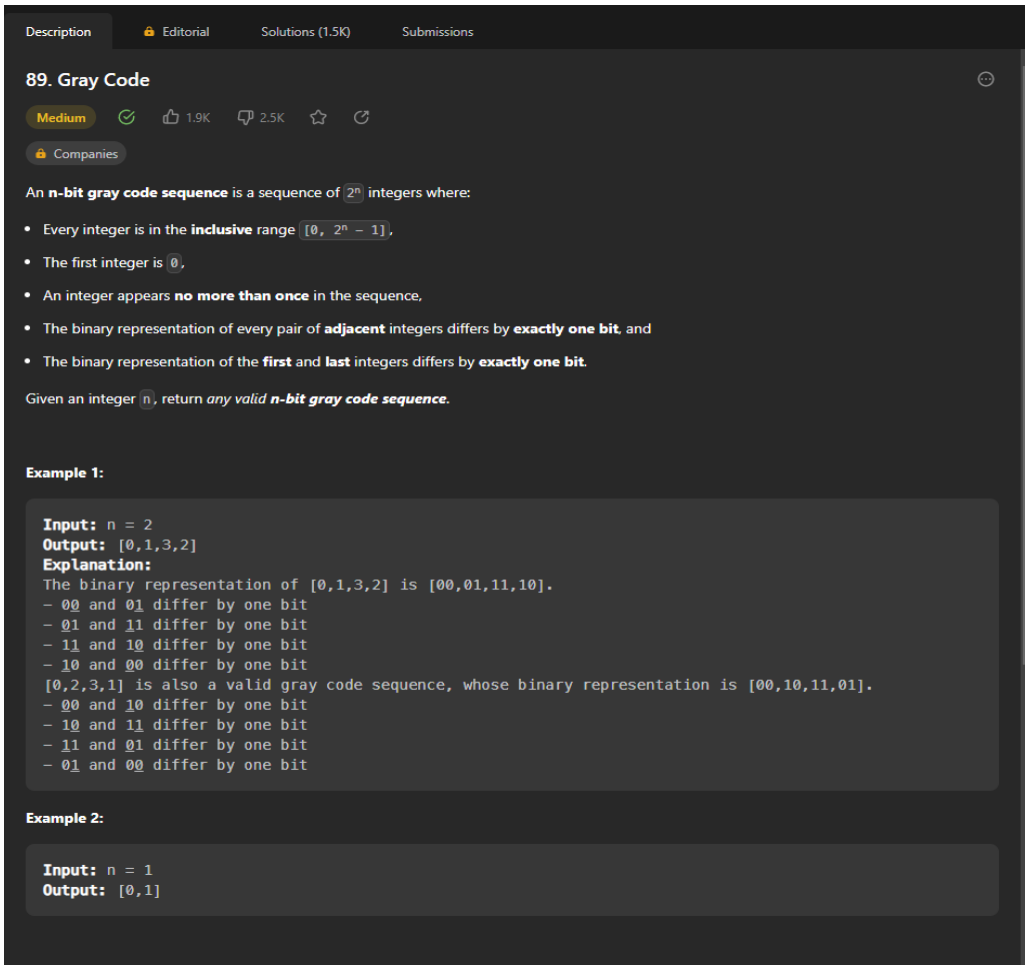
To implement the concept of Graphs.

2. Objective:

- The objective is to build problem solving capability and to learn the basic concepts of data structures.
- Understand the problem and find out better approach to solve particular problem

3. LeetCode code and output:

• Grey Code



Description | Editorial | Solutions (1.5K) | Submissions

89. Gray Code

Medium 1.9K 2.5K

Companies

An **n-bit gray code sequence** is a sequence of 2^n integers where:

- Every integer is in the **inclusive** range $[0, 2^n - 1]$,
- The first integer is 0,
- An integer appears **no more than once** in the sequence,
- The binary representation of every pair of **adjacent** integers differs by **exactly one bit**, and
- The binary representation of the **first** and **last** integers differs by **exactly one bit**.

Given an integer n, return *any valid* ***n-bit gray code sequence***.

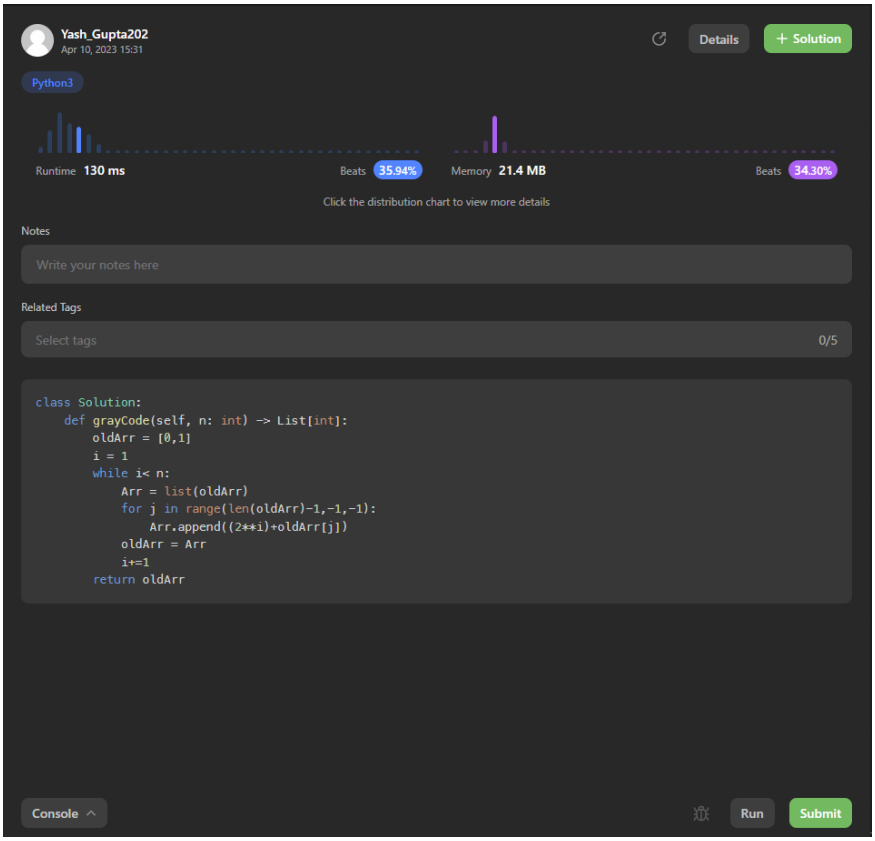
Example 1:

```
Input: n = 2
Output: [0,1,3,2]
Explanation:
The binary representation of [0,1,3,2] is [00,01,11,10].
- 00 and 01 differ by one bit
- 01 and 11 differ by one bit
- 11 and 10 differ by one bit
- 10 and 00 differ by one bit
[0,2,3,1] is also a valid gray code sequence, whose binary representation is [00,10,11,01].
- 00 and 10 differ by one bit
- 10 and 11 differ by one bit
- 11 and 01 differ by one bit
- 01 and 00 differ by one bit
```

Example 2:

```
Input: n = 1
Output: [0,1]
```

```
class Solution:
    def grayCode(self, n: int) -> List[int]:
        oldArr = [0,1]
        i = 1
        while i< n:
            Arr = list(oldArr)
            for j in range(len(oldArr)-1,-1,-1):
                Arr.append((2**i)+oldArr[j])
            oldArr = Arr
            i+=1
        return oldArr
```



- **Symmetric Tree**

```
class Solution:
    def groupThePeople(self, groupSizes: List[int]) -> List[List[int]]:
      group_dic = defaultdict(list)
      ans = []
      for idx, size in enumerate(groupSizes):
        group_dic[size].append(idx)
        if(len(group_dic[size]) == size):
          ans.append(group_dic[size])
          del group_dic[size]
      return ans
```

Description | Editorial | Solutions (1.5K) | Submissions

1282. Group the People Given the Group Size They Belong To

Hint

Medium | 1.5K | 526

Companies

There are `n` people that are split into some unknown number of groups. Each person is labeled with a **unique ID** from `0` to `n - 1`.

You are given an integer array `groupSizes`, where `groupSizes[i]` is the size of the group that person `i` is in. For example, if `groupSizes[1] = 3`, then person `1` must be in a group of size `3`.

Return *a list of groups such that each person `i` is in a group of size `groupSizes[i]`*.

Each person should appear in **exactly one group**, and every person must be in a group. If there are multiple answers, **return any of them**. It is **guaranteed** that there will be **at least one** valid solution for the given input.

Example 1:

```
Input: groupSizes = [3,3,3,3,3,1,3]
Output: [[5],[0,1,2],[3,4,6]]
Explanation:
The first group is [5]. The size is 1, and groupSizes[5] = 1.
The second group is [0,1,2]. The size is 3, and groupSizes[0] = groupSizes[1] = groupSizes[2] = 3.
The third group is [3,4,6]. The size is 3, and groupSizes[3] = groupSizes[4] = groupSizes[6] = 3.
Other possible solutions are [[2,1,6],[5],[0,4,3]] and [[5],[0,6,2],[4,3,1]].
```

Example 2:

```
Input: groupSizes = [2,1,3,3,3,2]
Output: [[1],[0,5],[2,3,4]]
```

Constraints:

- `groupSizes.length == n`
- `1 <= n <= 500`
- `1 <= groupSizes[i] <= n`