GFG Student Chapter – BTI Bangalore.

DSA BOOTCAMP DAY 10:

Ques.> Gray Code

Given a number N, generate bit patterns from 0 to 2^N-1 such that successive patterns differ by one bit. A Gray Code sequence must begin with 0.

Gray Code: Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit). Gray codes come in advantage especially in the normal sequence of binary numbers generated by the hardware that may cause an error or ambiguity during the transition from one number to next since there is one bit value change. Gray code does not depend on positional values of a digit. It is also called as "Reflected binary code".

There are several ways to solve this problem. Some are mentioned here:

METHOD 1

Approach:

The above sequences are Gray Codes of different widths. Following is an interesting pattern in Gray Codes.

n-bit Gray Codes can be generated from a list of (n-1)-bit Gray codes using the following steps:

- ➢ Let the list of (n-1)-bit Gray codes be L1. Create another list L2 which is the reverse of L1.
- ➢ Modify the list L1 by prefixing a '0' in all codes of L1.
- ➢ Modify the list L2 by prefixing a '1' in all codes of L2.
- ➢ Concatenate L1 and L2. The concatenated list is required list of n-bit Gray codes
- ➢ For example, following are steps for generating the 3-bit Gray code list from the list of 2-bit Gray code list.
- ➢ L1 = {00, 01, 11, 10} (List of 2-bit Gray Codes)

- ➢ L2 = {10, 11, 01, 00} (Reverse of L1)
- ➢ Prefix all entries of L1 with ‘0’, L1 becomes {000, 001, 011, 010}
- ➢ Prefix all entries of L2 with ‘1’, L2 becomes {110, 111, 101, 100}
- ➢ Concatenate L1 and L2, we get {000, 001, 011, 010, 110, 111, 101, 100}
- ➢ To generate n-bit Gray codes, we start from a list of 1 bit Gray codes. The list of 1 bit Gray code is {0, 1}. We repeat the above steps to generate 2 bit Gray codes from 1 bit Gray codes, then 3-bit Gray codes from 2-bit Gray codes till the number of bits becomes equal to n.

The Time complexity of this approach is O(2^N) which means the algorithm’s growth doubles with each addition to the input data set.

Meanwhile the auxiliary space is O(2^N). Auxiliary space refers to temporary space required by an algorithm to be used.

METHOD 2 (Recursive Approach)

Approach:

We use recursive method to solve given problem. Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem. Recursion solves such recursive problems by using functions that call themselves from within their own code. The idea is to recursively append the bit 0 and 1 each time until the number of bits is not equal to N.

Base condition: The base condition for this problem will be when the value of N=0 or 1.

i.e.

```
If (N == 0)
    return {“0”}
```

```
if (N == 1)
    return {“0”, “1”}
```

Recursive Condition: Otherwise, for any value greater than 1, recursively generate the gray codes of the N – 1 bits and then for each of the Gray code generated add the prefix 0 and 1.

The Time complexity of this approach is $O(2^N)$ which means the algorithm’s growth doubles with each addition to the input data set.

Meanwhile the auxiliary space is $O(2^N)$. Auxiliary space refers to temporary space required by an algorithm to be used.

METHOD 3 (Using Bitset)

Approach:

A bitset is an array of bool but each Boolean value is not stored separately instead bitset optimizes the space such that each bool takes 1 bit space only, so space taken by bitset bs is less than that of bool bs[N] and vector bs(N). However, a limitation of bitset is, N must be known at compile time, i.e., a constant (this limitation is not there with vector and dynamic array).

We should first find binary no from 1 to n and then convert it into string and then print it using the substring function of string.

The Time Complexity of this algorithm is $O(2^N)$ which means the algorithm’s growth doubles with each addition to the input data set.

Meanwhile the auxiliary space is O(N). Auxiliary space refers to temporary space required by an algorithm to be used.