Gray Code

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.

Solution 1

```
public List<Integer> grayCode(int n) {
    List<Integer> result = new LinkedList<>();
    for (int i = 0; i < 1<<n; i++) result.add(i ^ i>>1);
    return result;
}
```

The idea is simple. G(i) = i^ (i/2).

written by jinrf original link here

Solution 2

I have a solution here which takes O(1) on space and no recursion used. Is this the best possible solution? (I combined the base cases in the loop as mike3 does. Thanks mike3!)

```
vector<int> grayCode(int n)
{
    vector<int> result(1, 0);
    for (int i = 0; i < n; i++) {
        int curCount = result.size();
        // push back all element in result in reverse order
        while (curCount) {
            curCount--;
            int curNum = result[curCount];
            curNum += (1<<i);
            result.push_back(curNum);
        }
    }
    return result;
}
```

written by scorpionsky original link here

Solution 3

My idea is to generate the sequence iteratively. For example, when n=3, we can get the result based on n=2. 00,01,11,10 -> (000,001,011,010) (110,111,101,100). The middle two numbers only differ at their highest bit, while the rest numbers of part two are exactly symmetric of part one. It is easy to see its correctness. Code is simple:

```
public List<Integer> grayCode(int n) {
    List<Integer> rs=new ArrayList<Integer>();
    rs.add(0);
    for(int i=0;i<n;i++){
        int size=rs.size();
        for(int k=size-1;k>=0;k--)
            rs.add(rs.get(k) | 1<<i);
    }
    return rs;
}
```

written by yuyibestman original link here