Daily Coding Problem #148

Problem

This problem was asked by Apple.

Gray code is a binary code where each successive value differ in only one bit, as well as when wrapping around. Gray code is common in hardware so that we don't see temporary spurious values during transitions.

Given a number of bits n, generate a possible gray code for it.

For example, for n = 2, one gray code would be `[00, 01, 11, 10]`.

Solution

We can solve this problem recursively—given that we have a gray code for `n - 1`, how can we extend that solution to work with n bits?

Let's take the base case, the solution for `n = 1`: `[0, 1]`.

Now, for `n = 2`, we obviously need to extend the solution by adding an extra bit. Notice that if we simply duplicate the solution twice, we get close to a solution:

```
[00, 01, 10, 11]
```

The only parts that don't work here are at the edges of where we're duplicating. Instead, if we duplicated the solution, prepend `0` to one copy, and prepend `1` to the other and reverse it, we get one where the edges match up:

```
[00, 01, 11, 10]
```

This works because between the copies, the only differences are at the most significant bit. We want to exploit this symmetry and flip the second copy so that at the edges, the only thing that changes is that significant bit.

```
def gray_code(n):
    if n == 0:
        return []
    elif n == 1:
        return [0, 1]

    prev_gray_code = gray_code(n - 1)

    result = []
    for code in prev_gray_code:
        result.append(code)

    for code in reversed(prev_gray_code):
        result.append((1 << n - 1) + code)

    return result
```

This takes O(2^n) time, since we're generating twice as many results for each recursive call and each recursive call reduces the input by a constant amount.