暨南大学本科实验报告专用纸

课程名称 算法分析与设计实验 成绩评定

实验项目名称 格雷码的构造 指导教师 李展

实验项目编号 实验二 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 2018051948

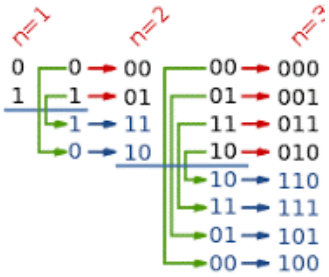
学院 信息科学技术 系 计算机科学 专业 网络工程

实验时间 2020 年 3 月 26 日 下午 ~ 3 月 26 日 下 午

一、问题描述

Gray 码是一个长度为2^n的序列。序列中无相同元素，每个元素都是长度为 n 位的（0，1）串，相邻元素恰好只有一位不同。利用分治法设计一个算法对任意的 n 构造相应的格雷码。

二、算法思路



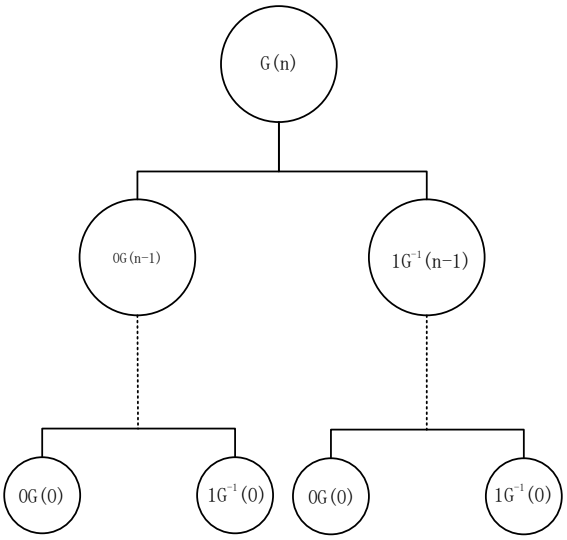
读取一个位数 n，计算出一个值为2^n的参数。格雷码的构造如右图示，每次构造都是将低位对称反转后，前一半的序列的最高位填 0，后一半序列最高位填 1。利用这个性质的倒推到得出本算法：

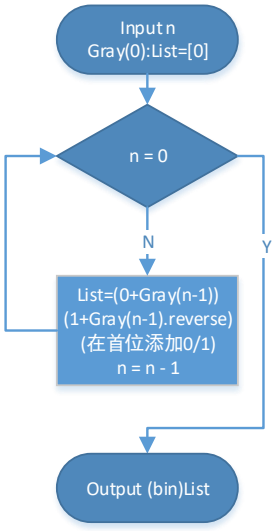
（1） 设置一个 n = 0 时为[0]的初始列表。

（2） 我们可以发现高位填充 1 这个操作可以使用移位运算符来实现，而相对于十进制数值大小而言，高位补 0，数值本身没有变化。

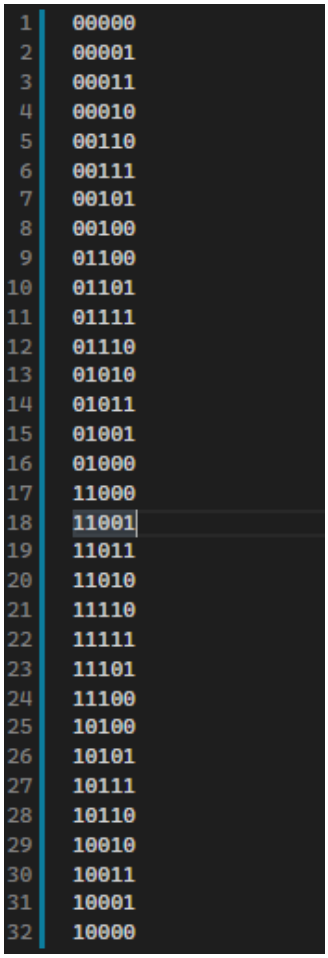
（3） 通过分治法，我们相当于将 G(n)的这一问题分解为 0G(n-1) 和 $1G^{-1}(n-1)$，最终相当于解决 0G(0)和 $1G^{-1}(n)$问题

三、算法实施流程





四、测试结果



五、实验总结

本算法运用了一个递归分治，因为每个小分块都有规律：N = n 时前一半格雷码从十进制数值上而言是没有变化的，可以直接复制，后一半相当于在二进制首位增加 1，时间复杂度为 O(n)；接着递归一次，得到该算法的时间复杂度：

$$T(n) = \begin{cases} O(1) & n = 1 \\ 2T(n-1) + O(n) & n > 1 \end{cases}$$

对该公式进行递归套用，最后算出，$T(n) = O(2^n)$。

可见，该算法的时间复杂度相对使用二维列表而言复杂度大幅降低，可能不是时间复杂度最优的算法，也没有那么好理解，需要有到十进制与二进制转换的敏感度与移位运算符的运用，但是从题目要求、时间与空间复杂度平衡性而言，它是一个极其优秀的算法。

同时，如果不需要分治思想的话，我们可以利用格雷码的电路的生成原理 G(i)=i^(i>>1)，使用位运算来达到空间与时间复杂度的最优。

最优算法的代码如下：

```
def calCode(n:int) -> list:
    return [i^(i>>1) for i in range(2**n)]
```

六、源代码

```
import os
PATH = '.\\'

class Gray():
    def calCode(self, n: int) -> list:  # 核心算法
        return [0] if n == 0 else self.calCode(n - 1) + [x + (1 << (n - 1)) for x in reversed(self.calCode(n - 1))]

n = eval(input("请输入格雷码位数:\n"))
ans = Gray()
result = []
b = ans.calCode(n)
for i in b:
    result.append(bin(i)[2:].zfill(n))  #二进制保存
print(result)
## 文件输出
f = open(PATH + '2.GrayCode_result', 'w', encoding='UTF-8')
for i in range(2 ** n):
    f.write(result[i])
    f.write('\n')
f.close()
```