**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 指导教师 李展 实验地点 N116

实验项目编号 001 实验项目类型 综合型

学生姓名 张瑞鹏 学号 2020101124

学院 信息科学技术学院 系 计算机 专业 计算机科学与技术

实验时间 2022 年 月 日 午～ 月 日 下 午

1. **问题描述**

**Gray码是一个长度为2^n的序列。序列中无相同的元素，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。用分治策略设计一个算法对任意的n构造相应的Gray码。**

1. **算法思路**（用文字简单说明）

运用分治的思想，既然要求是每一位相邻的格雷码都只相差一个数，显然，如果利用（n-1）的构造结果，然后再全部加上‘1’或者‘0’显然也符合描述的两两之间只相差1个数字

第一步：

先将之前构造好的格雷码按顺序排列，再最后一个格雷码之后，倒叙排列，这样格编码的数量翻了一倍。

第二部：

再在上述编码的基础之上，前半部分每个编码在最前面（或者最后面）添加一个‘0’，后半部分全部添加一个‘1’，这样格雷码就构造完成了。

算法递归的出口（或者循环的结束）

对于n=1的格雷码直接定义为{0，1}；

例：（构造三位的格雷码）  
如要求n=2的格雷码，需要用n=1的结果，在前面添‘0’即{00，01}以及倒叙后添‘1’{11，10}，然后合并{00，01，11，10}.构造3位的格雷码可以看成将已经构造好的2位的格雷码前面按顺序全部加上0，接上按倒转顺序，前面全部街上1，便形成3位的格雷码{000，001，011，010，110，111，101，100}

1. **算法实施步骤和流程**（伪代码/流程图等方式描述）

**递归写法：**

**char\*\* Graycode(int n)//返回一个二维数组数表**

**{**

**int i,j;//需要用到的辅助变量**

**if (n == 1){//如果格雷码是一位的**

**char\*\* head = new char\*[2];//(char\*\*)malloc(2 \* sizeof(char\*));**

**head[0] = new char[2];**

**head[1] = new char[2];**

**head[0][0] = '0';head[0][1] = '\0';**

**head[1][0] = '1';head[1][1] = '\0';**

**return head;**

**}**

**else**

**{**

**int result = pow(2, n - 1);**

**char\*\* temp = Graycode(n - 1);//分治思想，调用之前的结果**

**char\*\* head = new char\*[2\*result];**

**for (i = 0; i < result; i++) {**

**head[i] = new char[n+1];//字符串结束符**

**head[i][0] = '0';head[i][1] = '\0';**

**strcat(head[i], temp[i]);**

**head[i][n] = '\0';**

**}**

**for (i = result,j = result-1; i < 2 \* result; i++,j--) {**

**head[i] = new char[n+1];**

**head[i][0] = '1';head[i][1] = '\0';**

**strcat(head[i], temp[j]);**

**head[i][n] = '\0';**

**}**

**delete []temp;**

**return head;**

**}**

**}**

**非递归写法：**

**char\*\* Graycode(int n)//返回一个二维数组数表**

**{**

**int i,j,k;//需要用到的辅助变量**

**//如果格雷码是一位的**

**char\*\* head = (char\*\*)malloc(pow(2, n)\*sizeof(char\*));//(char\*\*)malloc(2 \* sizeof(char\*));**

**for(i = 0;i<n;i++){**

**head[i] = (char\*)malloc((n+10)\*sizeof(char));**

**}**

**head[0][0] = '0';head[0][1] = '\0';**

**head[1][0] = '1';head[1][1] = '\0';**

**if (n==1) return head;**

**char\* temp = (char\*)malloc((n+1)\*sizeof(char));//分配辅助空间**

**for(k = 2;k<=n;i++)**

**{**

**int result = pow(2, k - 1);**

**for (i = result,j = result-1; i < 2 \* result; i++,j--) {**

**strcpy(temp,"");strcpy(temp,"1");temp[1] = '\0';**

**strcat(temp,head[j]);temp[k]='\0';**

**strcpy(head[i],"");strcpy(head[i],temp);**

**}**

**for (i = 0; i < result; i++) {**

**strcpy(temp,"");strcpy(temp,"0");temp[1] = '\0';**

**strcat(temp,head[i]);strcpy(head[i],temp);**

**}**

**}**

**return head;**

**}**

1. **源代码**（通过了编译运行的正确程序）

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**#include<math.h>**

**#include<iostream>**

**using namespace std;**

**char\*\* Graycode(int n)//返回一个二维数组数表**

**{**

**int i,j;//需要用到的辅助变量**

**if (n == 1){//如果格雷码是一位的**

**char\*\* head = new char\*[2];//(char\*\*)malloc(2 \* sizeof(char\*));**

**head[0] = new char[2];**

**head[1] = new char[2];**

**head[0][0] = '0';head[0][1] = '\0';**

**head[1][0] = '1';head[1][1] = '\0';**

**return head;**

**}**

**else**

**{**

**int result = pow(2, n - 1);**

**char\*\* temp = Graycode(n - 1);//分治思想，调用之前的结果**

**char\*\* head = new char\*[2\*result];**

**for (i = 0; i < result; i++) {**

**head[i] = new char[n+1];//字符串结束符**

**head[i][0] = '0';head[i][1] = '\0';**

**strcat(head[i], temp[i]);**

**head[i][n] = '\0';**

**}**

**for (i = result,j = result-1; i < 2 \* result; i++,j--) {**

**head[i] = new char[n+1];**

**head[i][0] = '1';head[i][1] = '\0';**

**strcat(head[i], temp[j]);**

**head[i][n] = '\0';**

**}**

**delete []temp;**

**return head;**

**}**

**}**

**int main(void)**

**{**

**int n;**

**printf("请输入n的大小：");**

**scanf("%d", &n);**

**printf("\n");**

**char\*\* result = Graycode(n);**

**int sq = pow(2, n);//计算格雷码位数**

**for (int i; i < sq; i++){**

**printf("%s\n", result[i]);**

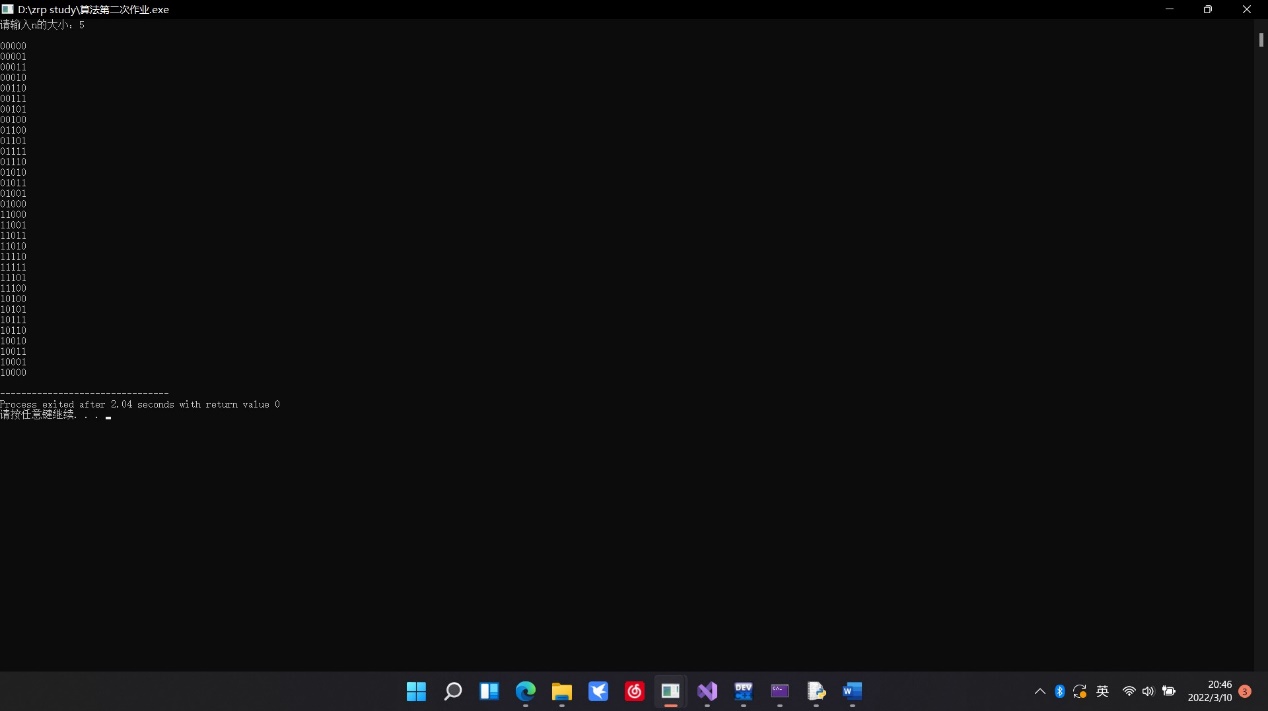
**}**

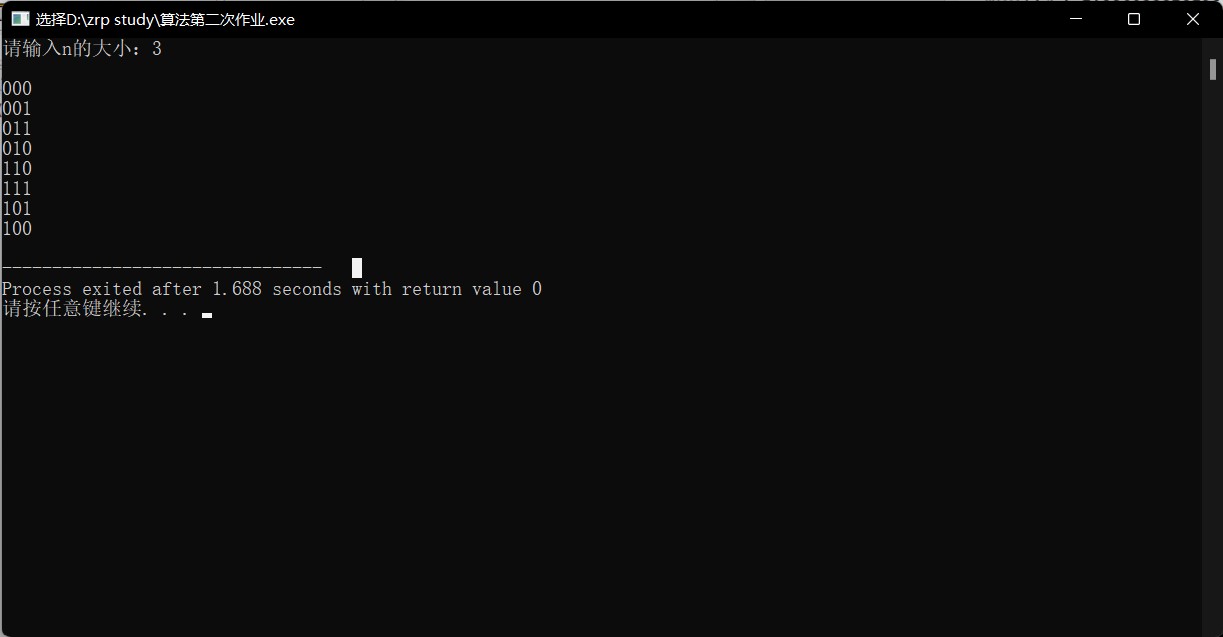
**return 0;**

**}**

**源代码只贴了递归版本，非递归版本仅是一个函数不同，在（三）中的代码即是非递归版本**

1. **测试结果**（至少有两个以上算例及程序运行结果，截图贴进实验报告）

****

****

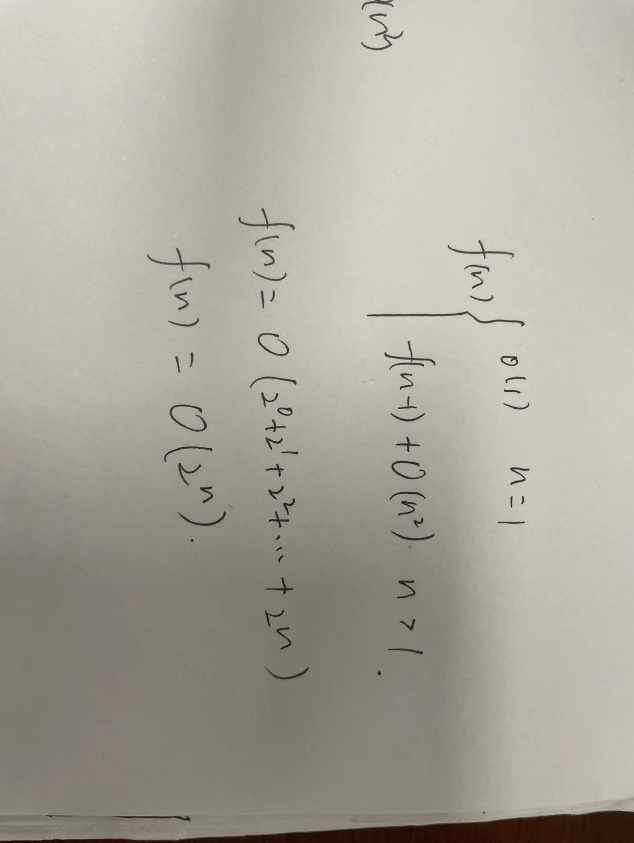
**（六）实验总结**（至少三句话，可以写复杂度分析、遇到问题、可能的改进措施、心得体会等）

1.时间复杂度为O（2^n）,是一个效率非常低的算法

2.尝试写过非递归，用循环代替递归，发现这道题目在算法复杂度而言提升不大。

3.递归的进行分治算法的程序，可读性比较强，便于理解。但是效率很低。

复杂度计算如下：



**暨南大学本科实验报告专用纸(附页)**