

**数学与信息学院学生实验报告**

**实验课程名称：** 算法分析与设计基础 **教师： \_\_**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **实验一 递归算法设计与应用** | | | **实验成绩** |  |
| **学生姓名** |  | **学 号** | **100** | **年级专业班级** |  |
| **小组成员** | **无** | | | **实验日期** | **2019年3 月** |

# 1. 实验目的和要求

## 1.1 实验目的

① 加深对**递归算法**的理解，并针对具体问题设计算法；

② 分析算法的复杂性，寻找比较高效的算法，并实现。

③ 分析**格雷码**问题，并设计递归算法求解之。

## 1.2 实验软硬件环境

① 操作系统

② 编译环境

## 1.3 实验要求

① 问题描述

② 具体要求

③ 测试数据和结果

# 2. 实验记录

一开始我想到的方法是：把一个格雷码拿过来，每次只变一位，这样可以保证相邻的格雷码一定只有一位不一样，但是这样会出现重复，所以把格雷码转化为十进制，这样就可以用一个vis数组标记，看看这个变化后的格雷码以前是否出现过。

代码：

#include <bits/stdc++.h>

using namespace std;

const int maxn = 4e8;

int vis[maxn];

int a[100],p[100],num;

void f(int n)

{

int temp = 0,temp2;

for(int i = 0;i < n;i++)

{

temp += a[i] \* p[i];

}

int flag = 0;

for(int i = 0;i < n;i++)

{

if(a[i])

{

temp2 = temp - a[i] \* p[i];

}

else

{

temp2 = temp + (!a[i])\* p[i];

}

if(!vis[temp2])

{

flag = 1;

a[i] = !a[i];

vis[temp2] = 1;

break;

}

}

if(flag)

{

printf("%d",a[n - 1]);

for(int i = n - 2;i >= 0;i--)

{

printf(" %d",a[i]);

}

puts("");

f(n);

}

return ;

}

int main()

{

int m,n;

p[0] = 1;

for(int i = 1;i <= 30;i++)

{

p[i] = p[i - 1] \* 2;

}

scanf("%d",&m);

while(m--)

{

num = 1;

scanf("%d",&n);

memset(vis,0,sizeof(vis));

memset(a,0,sizeof(a));

printf("0");

for(int i = 1;i < n;i++)

{

printf(" 0");

}

puts("");

vis[0] = 1;

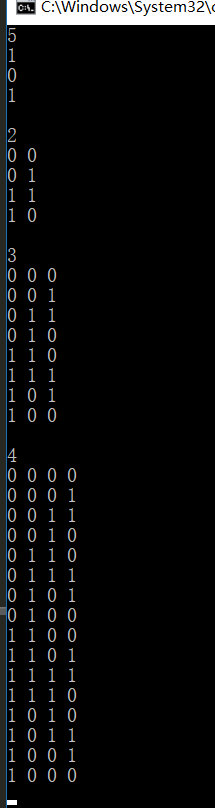
f(n);

}

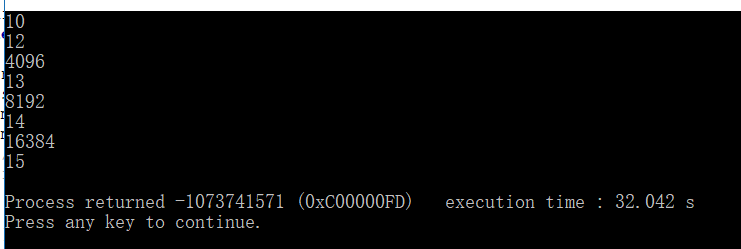
return 0;

}

运行结果：



然后发现这个方法不好，因为n的大小会受vis数组大小的限制，vis可以开到4e8，开到5e8就炸了。再然后发现我想多了，空间还用不到4e8（n还不需要到28）呢，栈就先炸了，这个代码，n最大可以为14，若n为15就会爆栈。



于是改为迭代

因为把所有格雷码都输出的话，就太多了，所以跑完后只输出格雷码的个数

代码：

#include <bits/stdc++.h>

using namespace std;

const int maxn = 4e8;

int vis[maxn];

int a[100],p[100],num,flag;

void f(int n)

{

//freopen("C:\\Users\\Administrator\\Desktop\\out.txt","w",stdout);

int temp = 0,temp2;

for(int i = 0;i < n;i++)

{

temp += a[i] \* p[i];

}

flag = 0;

for(int i = 0;i < n;i++)

{

if(a[i])

{

temp2 = temp - a[i] \* p[i];

}

else

{

temp2 = temp + (!a[i])\* p[i];

}

if(!vis[temp2])

{

flag = 1;

a[i] = !a[i];

vis[temp2] = 1;

return ;

}

}

return ;

}

int main()

{

int m,n;

p[0] = 1;

for(int i = 1;i <= 30;i++)

{

p[i] = p[i - 1] \* 2;

}

scanf("%d",&m);

while(m--)

{

num = 0;

scanf("%d",&n);

memset(vis,0,sizeof(vis));

memset(a,0,sizeof(a));

vis[0] = 1;

flag = 1;

while(true)

{

if(flag)

{

num++;

f(n);

}

else

break;

}

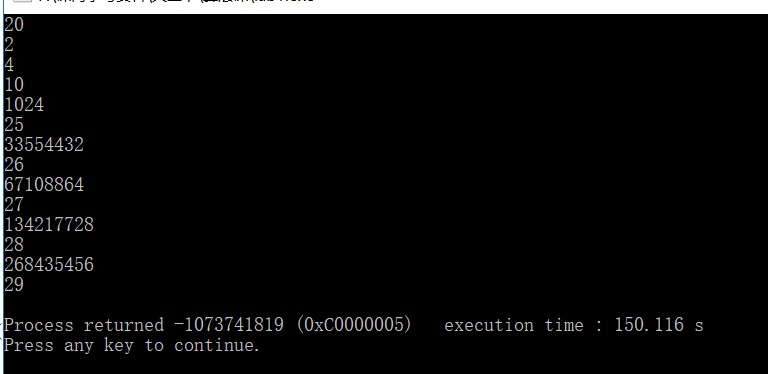
printf("%d\n",num);

}

return 0;

}

此时受空间限制，n最大为28，n为29时，数组溢出。



后来观察格雷码发现，有另一种方法会简单些，初始字符串为全0，输出，把第n位翻转（0变1,1变0），输出，再把第n-1位翻转，输出，然后回去翻转第n位，输出，然后翻转第n-2位，输出，回去翻转第n位，输出，再回去翻转第n-1位，这时，就和前面翻转第n-1位时一样，接下来以此类推。所以也可以写成递归。

代码：

#include <bits/stdc++.h>

using namespace std;

const int maxn = 4e8;

int n;

char s[100];

void dfs(int now)

{

//freopen("C:\\Users\\Administrator\\Desktop\\out.txt","w",stdout);

if(now == n)

{

for(int i = 0;i < n - 1;i++)

{

printf("%c ",s[i]);

}

printf("%c\n",s[n - 1]);

return ;

}

dfs(now + 1);

s[now] = !(s[now] - '0') + '0';

dfs(now + 1);

return ;

}

int main()

{

int m;

scanf("%d",&m);

while(m--)

{

scanf("%d",&n);

for(int i = 0;i < n;i++)

{

s[i] = '0';

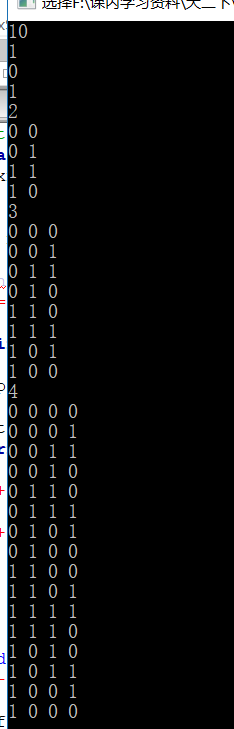
}

dfs(0);

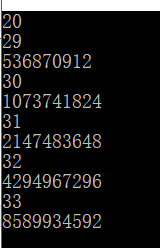
}

return 0;

}



这个方法的好处在于：它对栈空间的占用很小，每次只有n个函数被挂起，测的时候，当n为33时，可以跑得出来，但是要很久，所以我也不知道n最大可以是多少。



最后，上网一查，发现还有种不错的方法，比如要求位数为n的，就在n-1的基础上加1和0，然后注意一下加的顺序就好了。

总结：

方法挺多的，认真想一想就可以优化空间复杂度或者时间复杂度，时空复杂度一起大幅度优化的时候很少。