# Problem A. 进制转换

## 题目描述

输入数据 0-2 的 32 次方范围,化成二进制,然后逆序这个二进制序列,转换成十进制。

* 测试输入

2

* 输出

1

* 测试输入

10

* 输出

5

## AC代码

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int a[LEN];  
  
string getBinarySequence(int num){  
 string ans="";  
 while(num){  
 if(num&1){  
 ans="1"+ans;  
 }else{  
 ans="0"+ans;  
 }  
 num>>=1;  
 }  
 return ans;  
}  
  
int getTrueValue(int r,string seq){//radix sequence  
 int n=seq.size();  
 int base=1;  
 int ans=0;  
 for(int i=n-1;i>=0;i--){  
 int v=seq[i]-'0';  
 ans+=v\*base;  
 base\*=r;  
 }  
 return ans;  
}  
  
  
int main()  
{  
 freopen("./in","r",stdin);  
 int num;  
 scanf("%d",&num);  
 string s=getBinarySequence(num);  
 bug(num)  
 bug(s)  
 reverse(s.begin(),s.end());  
 int ans=getTrueValue(2,s);  
 printf("%d",ans);  
 return 0;  
}

# Problem B. 输出图象数字

## 题目描述

输出图象数字

## 输入:

0-9 任意一个数字

## 输出:

输出对应的 5\*3 的图像数字。

# Problem C. 发财数

https://blog.csdn.net/qq\_37640597/article/details/79772948

## 题目描述

一个大于等于2的整数，如果可以分解为8个或8个以上的素数相乘，则称其为发财数，让你输出第n个发财数（n最大到1w）

## 样例输入：

1  
1

## 样例输出：

256

# Problem D. 最长平衡串

http://www.bubuko.com/infodetail-2273733.html

## 题目描述

给定只含01的字符串，找出最长平衡子串的长度（平衡串：包含0和1的个数相同），串长最大十万

## 样例输入：

01001

## 样例输出：

1001

## 代码

因为没有OJ可以提交，所以不确定是否正确，请评论区指点

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
char s[10010];  
int a[10010];  
  
map<int,int> sum2i; //把当前前缀和转换为最早的下标  
  
int main()  
{  
 freopen("in","r",stdin);  
 gets(s);  
 string str(s);  
 int N=strlen(s);  
 int sum=0;  
 int maxL=0;  
 string bestStr="";  
 FF(i,N){  
 //完成从字符到数字的转换  
 if(s[i]=='0'){  
 a[i]=-1;  
 }else{  
 a[i]=1;  
 }  
 //把当前前缀和转换为最早的下标  
 sum+=a[i];  
 if(sum2i.count(sum)==0){  
 sum2i[sum]=i;  
 }  
 //更新最优解  
 if(sum==0){  
 int L=i+1;  
 if(L>maxL){  
 bestStr=str.substr(0,L);  
 maxL=i;  
 }  
 }else{  
 if(sum2i.count(sum)){ //可以在草稿纸上手推一下，找到当前和的最早出现下标，就是最佳字串的前一个下标  
 int j=sum2i[sum];   
 int L=i-j;  
 if(L>maxL){  
 maxL=L;  
 bestStr=str.substr(j+1,L);  
 }  
 }  
 }  
 }  
 printf("%s\n",bestStr.c\_str());  
 bug(maxL)  
 return 0;  
}