**并查集**

#include<bits/stdc++.h>

const int MAXN=10000;

int height[MAXN],f[MAXN];

void init\_set(){

for(int i=0;i<MAXN;i++){

f[i]=i;

height[i]=0;

}

}

int find\_set(int x){

if(x!=f[x]) f[x]=find\_set(f[x]);

return f[x];

}

void union\_set(int a,int b){

a=find\_set(a);

b=find\_set(b);

if(height[a]==height[b]){

height[a]++;

f[b]=a;

}else{

if(height[a]<height[b]){

f[a]=b;

}else{

f[b]=a;

}

}

}

using namespace std;

int main(){

int N,M,a,b;

int j=1;

while(cin>>N>>M && (N!=M) ){

init\_set();

for(int i=0;i<M;i++){

cin>>a>>b;

union\_set(a,b);

}

int ans=0;

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

if(f[i]==i) ans++;

cout<<"Case "<<j++<<": "<<ans<<endl;

}

return 0;

}

**KMP**

#include<bits/stdc++.h>

using namespace std;

int Next[100];

void GetNext(string ch,int \*next){

int len=ch.size();

next[0]=-1;

int i=0,j=-1;

while(i<len){

if(!(~j)||ch[i]==ch[j]) next[++i]=++j;

else j=next[j];

}

}

int KMP(string s,string p){

int last=0;

int slen=s.size(),plen=p.size();

GetNext(p,Next);

int j=0,i=0;

while(i<slen){

if(!(~j)||s[i]==p[j]){

i++;

j++;

}else

j=Next[j];

if(j>=plen) {

j=Next[j];

return i-plen;

}

}

return -1;

}

int main(){

string s,subs;

cin>>s;

cin>>subs;

cout<<KMP(s,subs);

return 0;

}

**快速幂**

#include<bits/stdc++.h>

#define ll long long

using namespace std;

ll fastpow(ll a,ll n,ll MOD){

ll base=a,res=1;

while(n){

if(n&1){

res=(res\*base)%MOD;

}

base=(base\*base)%MOD;

n>>=1;

}

return res;

}

int main(){

ll a,n;

cin>>a>>n;

ll ans=fastpow(a,n,1000);

cout<<ans;

vector<int> x;

return 0;

}

**质数**

#include<bits/stdc++.h>

#define ll long long

using namespace std;

const int M=1e7;

int prime[M+1];

bool visit[M+1];

int E\_sieve(int n){

int k=0;

for(int i=0;i<n;i++) visit[i]=0;

for(int i=2;i<=n;i++){

if(!visit[i]){

for(int j=i\*i;j<=n;j+=i) visit[j]=1;

}

}

for(int i=2;i<=n;i++){

if(!visit[i]){

prime[k++]=i;

}

}

return k;

}

int main(){

int n,k;

cin>>n;

k=E\_sieve(n);

cout<<k<<endl<<endl;

for(int i=0;i<k;i++){

cout<<prime[i]<<" ";

}

return 0;

}