//第二类斯特林数

//S(n,m)=1/m!\*sum(k:[0,m],(-1)^k\*C(m,k)\*(m-k)^n)

//n^k=sum(i:[0,k],S(k,i)\*i!\*C(n,i))

#include<stdio.h>

//using namespace std;

#define ll long long

const int mod=1e9+7;

const int MAXN=1e6;

int n,m;

ll jc[MAXN+20]={1};

int inv[MAXN+20]={1,1};

ll mo(ll x,int p){

return x<0?(x+p)%p:x<p?x:x%p;

}

ll speed(ll a,ll b,int p){

ll cur=a,ans=1;

while(b){

if(b&1) ans=ans\*cur%p;

cur=cur\*cur%p;

b>>=1;

}

return ans%p;

}

inline void init(){

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

inv[i]=((mod-mod/i)\*1LL\*inv[mod%i])%mod;

}

for(int i=1;i<=m;i++){

inv[i]=inv[i-1]\*1LL\*inv[i]%mod;

jc[i]=jc[i-1]\*i%mod;

}

}

inline ll C(ll m,ll n){

return jc[m]\*inv[n]%mod\*inv[m-n]%mod;

}

int main(){

ll ans=0;

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

init();

ll f=-1;

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

f=-f;

ans=mo(ans+f\*C(m,i)\*speed(m-i,n,mod),mod);

}

//ans=mo(ans\*njc[m],mod);

printf("%lld\n",(ans+mod)%mod);

return 0;

}

//LIS

#include<bits/stdc++.h>

using namespace std;

int LIS(int num[],int l,int r,bool dec=false,bool equ=false){

if(l>=r) return 0;

vector<int>vdp;

if(dec){

for(int i=l;i<=r;i++){

num[i]=-num[i];

}

}

for(int i=l;i<=r;i++){

vector<int>::iterator iter;

if(!equ) iter=lower\_bound(vdp.begin(),vdp.end(),num[i]);

else iter=upper\_bound(vdp.begin(),vdp.end(),num[i]);

if(iter==vdp.end()) vdp.push\_back(num[i]);

else \*iter=num[i];

}

if(dec){

for(int i=l;i<=r;i++){

num[i]=-num[i];

}

}

return vdp.size();

}

int n,ans;

int a[100020];

int main(){

while(scanf("%d",&a[++n])!=EOF);

n--;

printf("%d\n",LIS(a,1,n,true,true));

printf("%d\n",LIS(a,1,n));

return 0;

}

//质因子分解

int c[MAXN+20];

int tmp[MAXN+20];

int main(){

prime();

/\*

分解质因数

n:要分解的数字

cnt:sqrt(n)以内素数的数量

pri[]:素数表，下标从1开始

c[]:存分解结果

tmp:最后可能剩下的素数

\*/

int in;

scanf("%d",&in);

for(int k=2;k<=in;k++){

int n=k;

for(int i=1;i<=cnt&&pri[i]\*pri[i]<=n;i++){

while(n%pri[i]==0){

c[i]++;

n/=pri[i];

}

if(n==1) break;

}

if(n>1) tmp[n]++;

}

for(int i=1;i<=cnt;i++){

int out=c[i]+tmp[pri[i]];

if(out>0){

printf("%d %d\n",pri[i],out);

}

}

return 0;

}

//模意义下的高斯消元

#include<cstdio>

#define maxn 110

#define r register

using namespace std;

typedef long long ll;

int n,p,maxi;

ll tmp,ans[maxn],a[maxn][maxn];

int read()

{

r char ch=getchar();r int in=0;

while(ch>'9'||ch<'0') ch=getchar();

while(ch>='0'&&ch<='9') in=(in<<3)+(in<<1)+ch-'0',ch=getchar();

return in;

}

ll ksm(r ll x,r int y)

{

if(!y) return 1;

r ll ret=ksm(x,y>>1);

if(y&1) return ret\*ret%p\*x%p;

return ret\*ret%p;

}

int main()

{

n=read(),p=read();

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

for(r int j=1;j<=n+1;j++)

a[i][j]=read();

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

{

if(!a[i][i])//主元不能为0

{

maxi=0;

for(r int j=i+1;j<=n&&!maxi;j++)

if(a[j][i]) maxi=j;

if(!maxi) continue;//如果一整列都为0，不需要消元

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

tmp=a[maxi][j],a[maxi][j]=a[i][j],a[i][j]=tmp;

}

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

{

tmp=a[j][i];

if(!tmp) continue;//已经为0，不需要消元

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

a[j][k]=((a[j][k]\*a[i][i]-a[i][k]\*tmp)%p+p)%p;

}

}

for(r int i=n;i;i--)

{

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

a[i][n+1]=((a[i][n+1]-ans[j]\*a[i][j])%p+p)%p;

ans[i]=a[i][n+1]\*ksm(a[i][i],p-2)%p;

}

for(r int i=1;i<=n;i++) printf("%lld ",ans[i]);

return 0;

}

//区间第k小 尺取

#include<iostream>

#include<algorithm>

using namespace std;

const int N = 1e5 + 5;

int a[N];

int n, k;

//尺取法求数组某一元素在所有子区间第k小元素中的最大位置（可能该元素有多个，返回排行最靠后的位置）

/\*维护两个指针：l和r，从初始位置开始，r一直向右移动直到满足条件或r的位置超出了数组的范围，进行相应的操作；

l向右移动一个元素，再进行上述过程\*/

long long max\_position(int x)

{

long long result = 0; //result用来记录所有子区间第k小元素中x的位置

int l = 0, r = -1, num = 0;

while (r < n)

{

if (num < k) //找到一个区间，使得此区间的元素小于等于x的个数为k，即x>=此区间的第k小元素

{

if (a[r + 1] <= x)num++;

r++;

}

else

{

// 如果此时的区间[l,r]第k小元素是x或小于x，那么区间[l,r+1],[l,r+2]...[l,n-1]的第k小元素也是x或小于x，所以x在所有第k小元素中的最大位次为n-r

//证明过程可以分情况讨论：1.a[r+1]<x,则区间[l,r+1]的第k小元素小于x；2.a[r+1]>=x,则区间[l,r+1]的第k小元素小于等于x（等于原区间的第k小元素）

result += n - r;

if (a[l] <= x)num--;

l++;

}

}

return result;

}

int main()

{

cin >> n >> k;

int\*b=new int[n];

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

{

cin >> a[i];

b[i] = a[i];

}

sort(b, b + n);

int len = unique(b, b + n) - b; //去掉重复元素

int l = 0, r = len - 1;

int ans = 0;

while (l <= r)

{

int mid = (l + r) / 2;

long long ret = max\_position(b[mid]);

//有两种情况：b[mid]就是题解或b[mid]大于题解（b[mid]等于题解也可能出现ret>k的情况，因为b[mid]可能在第k小中有多个，返回的是最大的位置）

//所以要找返回值大于等于k的最小元素

if (ret >= k)

{

ans = b[mid];

r = mid - 1;

}

else l = mid + 1;

}

cout << ans;

return 0;

}

//Tarjan求有向图强连通分量

#include<bits/stdc++.h>

using namespace std;

const int MAXN=10020;

const int MAXM=100020;

int n,m,idx,cnt,g[MAXN];

int dfn[MAXN],low[MAXN];

bool instack[MAXN];

struct E{

int u,v,nex;

}e[MAXM];

stack<int>s;

vector<int>belong[MAXN];

void Tarjan(int u){

dfn[u]=low[u]=++idx;

s.push(u);

instack[u]=true;

for(int i=g[u];i>0;i=e[i].nex){

int v=e[i].v;

if(!dfn[v]){

Tarjan(v);

low[u]=min(low[u],low[v]);

}

else if(instack[v]){

low[u]=min(low[u],dfn[v]);

}

}

if(dfn[u]==low[u]){

++cnt;

int cur;

do{

cur=s.top();s.pop();

instack[cur]=false;

belong[cnt].push\_back(cur);

}while(cur!=u);

}

}

int main(){

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

int x,y;

for(int i=1;i<=m;i++){

scanf("%d%d",&x,&y);

e[i]=(E){x,y,g[x]};g[x]=i;

}

Tarjan(1);

return 0;

}

//Tarjan缩点

#include<bits/stdc++.h>

#define io\_opt ios::sync\_with\_stdio(false);cin.tie(0);cout.tie(0)

using namespace std;

const int MAXN=500020;

const int MAXM=500020;

int n,m,idx,cnt,mm;

int g[MAXN],dfn[MAXN],low[MAXN],p2p[MAXN];

bool instack[MAXN];

stack<int>s;

vector<int>belong[MAXN];

struct E{

int u,v,nex;

bool operator<(const E x)const{

if(u==x.u) return v<x.v;

return u<x.u;

}

bool operator==(const E x)const{

return u==x.u&&v==x.v&&nex==x.nex;

}

}e[MAXM];

void Tarjan(int u){

dfn[u]=low[u]=++idx;

s.push(u);

instack[u]=true;

for(int i=g[u];i>0;i=e[i].nex){

int v=e[i].v;

if(!dfn[v]){

Tarjan(v);

low[u]=min(low[u],low[v]);

}

else if(instack[v]){

low[u]=min(low[u],dfn[v]);

}

}

if(dfn[u]==low[u]){

++cnt;

int cur;

do{

cur=s.top();s.pop();

instack[cur]=false;

belong[cnt].push\_back(cur);

p2p[cur]=cnt;

}while(cur!=u);

}

}

int main(){

io\_opt;

cin>>n>>m;

int x,y;

for(int i=1;i<=m;i++){

cin>>x>>y;

e[i]=(E){x,y,g[x]};g[x]=i;

}

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

if(!dfn[i]) Tarjan(i);

}

for(int i=1;i<=m;i++){

int u=e[i].u,v=e[i].v;

if(!p2p[u]||!p2p[v]||p2p[u]==p2p[v]){

e[i].u=e[i].v=1000000;

e[i].nex=1000000;

}

else{

e[i].u=p2p[u];

e[i].v=p2p[v];

e[i].nex=0;

}

}

sort(e+1,e+1+m);

mm=unique(e+1,e+1+m)-(e+1);

while(e[mm].nex==1000000) mm--;

memset(g,0,sizeof(g));

for(int i=1;i<=mm;i++){

e[i].nex=g[e[i].u];

g[e[i].u]=i;

}

return 0;

}

//接近分组

int main(){

//cut n to m

cin>>n>>m;

for(int i=1;i<=m;i++){

if(i!=1) printf(" ");

cout<<ceil((n-i+1)/(db)m);

}

cout<<endl;

return 0;

}

//优读

int inline read(){

int num=0;

char c;

bool plus=true;

while((c=getchar())==' '||c=='\n'||c=='\r');

if(c=='-') plus=false;

else num=c-'0';

while(isdigit(c=getchar()))

num=num\*10+c-'0';

return num\*(plus?1:-1);

}

//埃筛

const int MAXN=1e7;

bool ipr[MAXN+20];

int cnt,pri[MAXN/5];

void prime(){//埃式筛法

int N=sqrt(MAXN)+0.5,mul;

memset(ipr,true,sizeof(ipr));

ipr[1]=false;

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

if(ipr[i]==true){

i==2?mul=1:mul=2;

for(int j=i\*i;j<=MAXN;j+=i\*mul){

ipr[j]=false;

}

}

}

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

if(ipr[i]==true){

pri[++cnt]=i;

}

}

}

//可靠快速幂

ll lowspeed(ll a,ll b,ll p){

ll cur=a,ans=0;

while(b){

if(b&1) ans=(ans+cur)%p;

cur=(cur+cur)%p;

b>>=1;

}

return ans%p;

}

ll speed(ll a,ll b,ll p){

ll cur=a,ans=1;

while(b){

if(b&1) ans=lowspeed(ans,cur,p)%p;

cur=lowspeed(cur,cur,p)%p;

b>>=1;

}

return ans%p;

}

//crt

//中国剩余定理模板

typedef long long ll;

ll china(ll a[],ll b[],int n)//a[]为除数，b[]为余数

{

ll M=1,y,x=0;

for(int i=0;i<n;++i) //算出它们累乘的结果

M\*=a[i];

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

{

ll w=M/a[i];

ll tx=0;

int t=exgcd(w,a[i],tx,y); //计算逆元

x=(x+w\*(b[i]/t)\*x)%M;

}

return (x+M)%M;

}

//excrt 自为风月马前卒

#include<iostream>

#include<cstdio>

#define LL long long

using namespace std;

const LL MAXN = 1e6 + 10;

LL K, C[MAXN], M[MAXN], x, y;

LL gcd(LL a, LL b) {

return b == 0 ? a : gcd(b, a % b);

}

LL exgcd(LL a, LL b, LL &x, LL &y) {

if (b == 0) {x = 1, y = 0; return a;}

LL r = exgcd(b, a % b, x, y), tmp;

tmp = x; x = y; y = tmp - (a / b) \* y;

return r;

}

LL inv(LL a, LL b) {

LL r = exgcd(a, b, x, y);

while (x < 0) x += b;

return x;

}

int main() {

#ifdef WIN32

freopen("a.in", "r", stdin);

#else

#endif

while (~scanf("%lld", &K)) {

for (LL i = 1; i <= K; i++) scanf("%lld%lld", &M[i], &C[i]);

bool flag = 1;

for (LL i = 2; i <= K; i++) {

LL M1 = M[i - 1], M2 = M[i], C2 = C[i], C1 = C[i - 1], T = gcd(M1, M2);

if ((C2 - C1) % T != 0) {flag = 0; break;}

M[i] = (M1 \* M2) / T;

C[i] = ( inv( M1 / T , M2 / T ) \* (C2 - C1) / T ) % (M2 / T) \* M1 + C1;

C[i] = (C[i] % M[i] + M[i]) % M[i];

}

printf("%lld\n", flag ? C[K] : -1);

}

return 0;

}

//excrt2

#include<iostream>

#include<string>

#include<cstdio>

typedef long long ll;

using namespace std;

const int maxn=100000+5;

int n;

ll ai[maxn],bi[maxn];

ll exgcd(ll a,ll b,ll &x,ll &y)

{

if(b==0){ x=1, y=0; return a;}

ll gcd=exgcd(b,a%b,x,y);

ll tp=x;

x=y, y=tp-a/b\*y;

return gcd;

}

ll mult(ll a,ll b,ll mod){

long long res=0;

while(b>0){

if(b&1) res=(res+a)%mod;

a=(a+a)%mod;

b>>=1;

}

return res;

}

ll excrt(){

ll x,y;

ll ans=bi[1],M=ai[1];

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

ll a=M,b=ai[i],c=(bi[i]-ans%ai[i]+ai[i])%ai[i];

ll gcd=exgcd(a,b,x,y);

x=mult(x,c/gcd,b/gcd);

ans+=x\*M;

M\*=b/gcd;

ans=(ans%M+M)%M;

}

return (ans%M+M)%M;

}

int main(){

cin>>n;

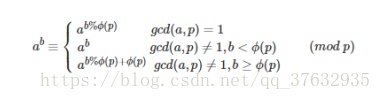
for(int i=1;i<=n;++i)

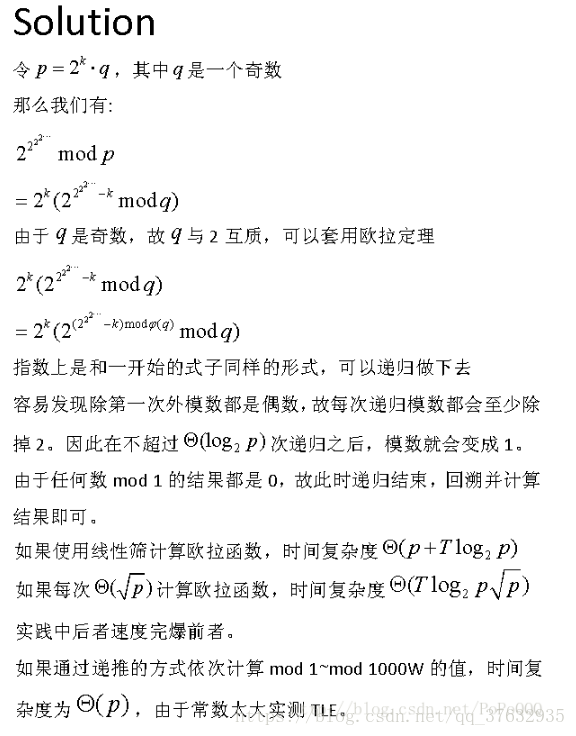
cin>>ai[i]>>bi[i];

cout<<excrt();

}

//欧拉降幂





#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

ll ph(ll x)

{

ll res=x,a=x;

for(ll i=2;i\*i<=x;i++)

{

if(a%i==0)

{

res=res/i\*(i-1);

while(a%i==0) a/=i;

}

}

if(a>1) res=res/a\*(a-1);

return res;

}

ll quick\_pow(ll a,ll b,ll mod)

{

ll ans=1;

while(b)

{

if(b&1) ans=(ans\*a)%mod;

a=(a\*a)%mod;

b>>=1;

}

return ans;

}

ll f(ll p)

{

if(p==1) return 0;

ll k=ph(p);

return quick\_pow(2,f(k)+k,p);

}

int main()

{

int T;

scanf("%d",&T);

while(T--)

{

ll p;scanf("%lld",&p);

printf("%lld\n",f(p));

}

return 0;

}

//欧拉函数线性筛

void euler(int n)

{

phi[1]=1;//1要特判

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

{

if (flag[i]==0)//这代表i是质数

{

prime[++num]=i;

phi[i]=i-1;

}

for (int j=1;j<=num&&prime[j]\*i<=n;j++)//经典的欧拉筛写法

{

flag[i\*prime[j]]=1;//先把这个合数标记掉

if (i%prime[j]==0)

{

phi[i\*prime[j]]=phi[i]\*prime[j];//若prime[j]是i的质因子，则根据计算公式，i已经包括i\*prime[j]的所有质因子

break;//经典欧拉筛的核心语句，这样能保证每个数只会被自己最小的因子筛掉一次

}

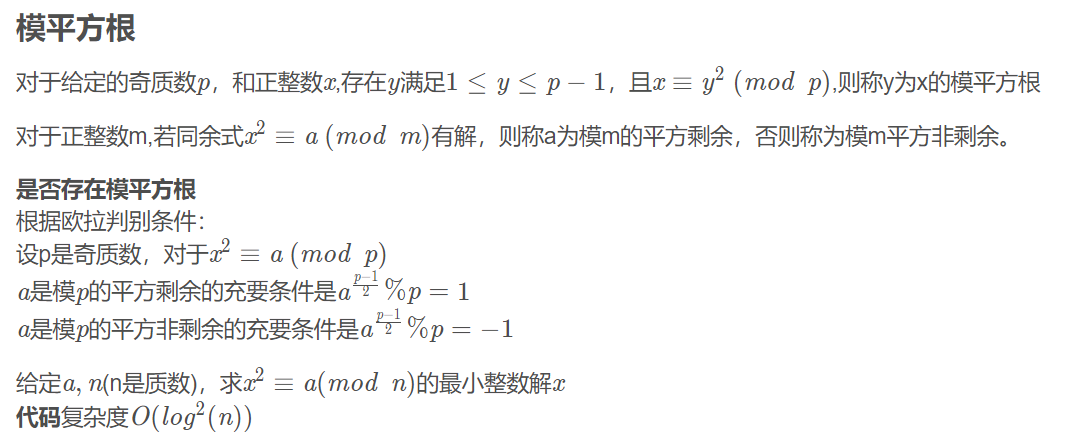
else phi[i\*prime[j]]=phi[i]\*phi[prime[j]];//利用了欧拉函数是个积性函数的性质

}

}

}

//平方剩余



#include <bits/stdc++.h>

using namespace std;

#define me(x,y) memset(x,y,sizeof x)

#define MIN(x,y) (x) < (y) ? (x) : (y)

#define MAX(x,y) (x) > (y) ? (x) : (y)

#define SGN(x) ((x)>0?1:((x)<0?-1:0))

#define ABS(x) ((x)>0?(x):-(x))

typedef long long ll;

typedef unsigned long long ull;

const int maxn = 1e5+10;

const ll INF = 0x3f3f3f3f;

const int MOD = 1e9+7;

const int eps = 1e-8;

ll qpow(ll a,ll b,ll p){

ll ans=1;

while(b){

if(b&1) ans=ans\*a%p;

a=a\*a%p;

b>>=1;

}

return ans;

}

int modsqr(int a,int n){

int b,k,i,x;

if(n==2) return a%n;

if(qpow(a,(n-1)/2,n) == 1){

if(n%4 == 3){

x=qpow(a,(n+1)/4,n);

}

else{

for(b=1; qpow(b,(n-1)/2,n) == 1; b++);

i = (n-1)/2;

k=0;

while(i%2==0){

i /= 2,k /= 2;

if((qpow(a,i,n)\*qpow(b,k,n)+1)%n == 0) k += (n-1)/2;

}

x = qpow(a,(i+1)/2,n)\*qpow(b,k/2,n)%n;

}

if(x\*2 > n) x = n-x;

return x;

}

return -1;

}

int main(){

int a,n;

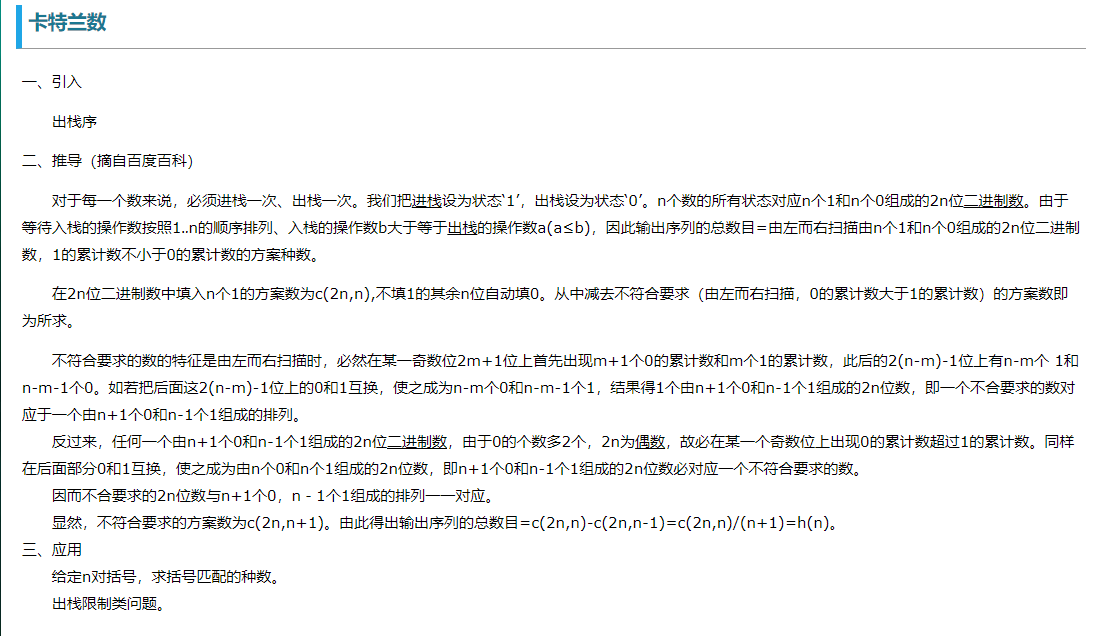
while(cin>>a>>n){

cout<<modsqr(a,n)<<endl;

}

return 0;

}





**代码**

作用：输出有多少次匹配，可重叠。

由于通常的输入从0开始，这个代码的next数组从0到n-1，每一位比正常的next数组少1。

回到开头的问题，如果想求的匹配不重叠，后面每当匹配到，last=0即可。

#include<bits/stdc++.h>

using namespace std;

int n,last;

char t[1000020],p[1002000];

int nex[1002000];

int main(){

scanf("%d",&n);

while(n--){

int ans=0;

scanf("%s%s",p,t);

int pl=strlen(p),tl=strlen(t);

nex[0]=last=-1;

for(int i=1;i<pl;i++){

while(last>-1&&p[i]!=p[last+1]){

last=nex[last];

}

if(p[i]==p[last+1]){

last++;

}

nex[i]=last;

}

last=-1;

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

while(last>-1&&t[i]!=p[last+1]){

last=nex[last];

}

if(t[i]==p[last+1]) last++;

if(last+1==pl){

ans++;

last=nex[last];

//printf("%d\n",i-pl+2);

}

}

/\*for(int i=0;i<pl;i++){

//if(i!=0) printf(" ");

//printf("%d",nex[i]);

}\*/

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

}

return 0;

}



# 例题

[hdu2328](http://acm.hdu.edu.cn/showproblem.php?pid=2328)

给一堆字符串，求最长公共字串。

找一个最短的串，暴力求出每一个后缀，和所有串匹配，找到每个extend里最大的，取总体最小，是一个答案，找到所有答案里长度最长的字典序最小的，就是答案。

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<cmath>

#include<string>

#include<iostream>

#define ll long long

#define db double

#define ioss ios::sync\_with\_stdio(false);cin.tie(0);cout.tie(0)

using namespace std;

int n,cnt;

ll ext[220],nex[220];

string skr[4020];

string ans[4020];

void getNext(string &strp,ll nextt[]){

ll pl=strp.size();

ll fir=0,las=0;

nextt[0]=pl;

for(ll i=1;i<pl;i++) {

nextt[i] = min(las - i + 1, nextt[i - fir]);

if (nextt[i] < 0) nextt[i] = 0;

while (i+nextt[i]<pl && strp[nextt[i]] == strp[i + nextt[i]]) {

nextt[i]++;

}

if (i + nextt[i] - 1 > las) {

las = i + nextt[i] - 1;

fir = i;

}

}

}

void exKMP(string &strp,string &strs,ll nextt[],ll extt[]){

//cout<<"start exKMP:"<<endl;

getNext(strp,nextt);

ll pl=strp.size(),sl=strs.size();

ll fir=0,las=-1,mnl=min(sl,pl);

//cout<<strp<<endl<<strs<<endl;

while(las<mnl-1&&strp[las+1]==strs[las+1]){

las++;

//cout<<"init++"<<endl;

}

extt[0]=las+1;

for(ll i=1;i<sl;i++){

extt[i]=min(las-i+1,nextt[i-fir]);

if(extt[i]<0) extt[i]=0;

while(extt[i]<pl && i+extt[i]<sl && strp[extt[i]]==strs[i+extt[i]]){

extt[i]++;

}

if(i+extt[i]-1>las){

las=i+extt[i]-1;

fir=i;

}

}

}

int main() {

//ioss;

//freopen("1.in","r",stdin);

//freopen("2.out","w",stdout);

while(scanf("%d",&n)==1&&n){

cnt=0;

int mnlen=300,mnlenx;

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

cin >> skr[i];

if (skr[i].size() < mnlen) {

mnlen = skr[i].size();

mnlenx = i;

}

}

for(int i=0;i<skr[mnlenx].size();i++){

ll mn=1e10;

string cur=skr[mnlenx].substr(i);

//out<<i+1<<": cur= "<<cur<<endl;

for(int j=1;j<=n;j++){

ll mx=0;

exKMP(cur,skr[j],nex,ext);

/\*cout<<"nex: ";

for(int k=0;k<cur.size();k++){

cout<<nex[k]<<' ';

}

cout<<endl;

cout<<"ext: ";\*/

for(int k=0;k<skr[j].size();k++){

//cout<<ext[k]<<' ';

mx=max(mx,ext[k]);

}

//cout<<endl;

mn=min(mn,mx);

//cout<<"mn = "<<mn<<endl;

}

if(mn>0){

if(cnt==0||(mn==ans[1].size())){

ans[++cnt]=cur.substr(0,mn);

}

else if(mn>ans[1].size()){

cnt=0;

ans[++cnt]=cur.substr(0,mn);

}

}

}

if(cnt){

sort(ans+1,ans+1+cnt);

cout<<ans[1]<<endl;

}

else cout<<"IDENTITY LOST"<<endl;

}

return 0;

}

//高斯消元

#include<cstdio>

#include<algorithm>

#include<cmath>

using namespace std;

const double eps=1e-6;

int n;

double a[110][110],b[110];

bool flag;

int main()

{

scanf("%d",&n);

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

{

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

scanf("%lf",&a[i][j]);

scanf("%lf",&b[i]);

}

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

{

flag=0;

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

{

if(flag)

break;

if(fabs(a[j][i])>eps)

{

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

swap(a[i][k],a[j][k]);

swap(b[i],b[j]);

flag=1;

}

}

if(!flag)

{

puts("No Solution");

return 0;

}

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

{

if(i==j)

continue;

double rate=a[j][i]/a[i][i];

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

a[j][k]-=a[i][k]\*rate;

b[j]-=b[i]\*rate;

}

}

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

printf("%.2lf\n",1.0\*b[i]/a[i][i]);

return 0;

}

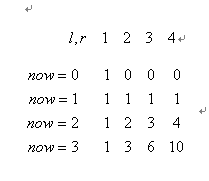
**中文题意：给定一个序列A，长度最长为100000，初始值为0，现在有三种操作：**

1.对区间[l,r]中所有的数都加上一个值。

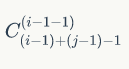
2.对整个序列求一次前缀和。

3.询问[l,r]区间内所有a的和。

 现在对1,0,0,0求3次前缀和得到下图



 可以发现(1,1)对右下角的点的贡献是



接下来我们定义一个变量now记录数组了进行了几次求数组前缀和也就是题目的2号操作。

对于操作1，在[l,r]区间内每个数增加w。

相当于在上次进行2号操作前，在点 L 增加w，在点 R+1 减少w。

例如：在3到5号位置增加1

序号    1 2 3 4 5 6 7 8 9

now     0 0 1 1 1 0 0 0 0     求前缀和后

now-1  0 0 1 0 0 -1 0 0 0   求前缀和前

对于询问的话只要求下次求完前缀和 （位置 R 的值） - （位置 L-1 的值）

对于进行前缀和操作只要将now++即可

具体操作看代码

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

typedef long long ll;

const int maxn = 410000,mod=998244353;

struct Stack

{

ll lie,time,value;

} st[maxn];

ll fac[maxn+10],ifac[maxn+10],top;

ll quick\_pow(ll a,ll b)

{

ll ans=1;

while(b)

{

if(b&1)

ans=1ll\*ans\*a%mod;

a=1ll\*a\*a%mod;

b>>=1;

}

return ans;

}

void init()

{

fac[0]=1;

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

fac[i]=1ll\*fac[i-1]\*i%mod;

ifac[maxn]=quick\_pow(fac[maxn],mod-2);

for(int i=maxn-1; i>=0; i--)

ifac[i]=1ll\*ifac[i+1]\*(i+1)%mod;

}

ll C(ll n,ll m)

{

return 1ll\*fac[n]\*ifac[m]%mod\*ifac[n-m]%mod;

}

inline ll solve(ll x,ll now)

{

if(x==0)return 0;

ll sum = 0;

for(int i=0; i<top; i++) ///计算每个更新对点的贡献值

{

if(st[i].lie>x)continue;

ll lie = st[i].lie;

ll per = st[i].time;

ll value = st[i].value;

ll aa = x-lie + now-per -1;

ll bb = now-per -1;

sum = (sum + value\*C(aa,bb)%mod)%mod;

}

return sum;

}

int main()

{

init(); ///预处理阶乘和逆元将计算组合数的时间复杂度降为O(1)

ll t,n,m,op;

scanf("%lld",&t);

while(t--)

{

scanf("%lld%lld",&n,&m);

ll now = 1,l,r,value;

top = 0;

while(m--)

{

scanf("%lld",&op);

if(op==1)

{

scanf("%lld%lld%lld",&l,&r,&value);

st[top].value = value%mod,st[top].time=now-1,st[top].lie=l;

top++;

st[top].value =(mod-value)%mod,st[top].time=now-1,st[top].lie=r+1;

top++;

///将更新存入数组

}

else if(op==2)

{

now++;

}

else

{

scanf("%lld%lld",&l,&r);

ll ans = solve(r,now+1)-solve(l-1,now+1);

ans = (ans+mod)%mod;

printf("%lld\n",ans);

}

}

}

return 0;

}