Matemática

(Matemática en la programación competitiva)

Eduardo Carranza Velez

Universidad Nacional de Córdoba - Facultad de Matemática, Astronomía y Física

Training Camp 2025

Gracias Sponsors!

Organizador

Diamond Plus

Gracias Sponsors!

Platino

FOLDER IT

INTERNATIONAL
SOFTWARE COMPANY

Gold

NeuralSoft

Oro

JS JERÁRQUICOS

Gracias Sponsors!

Aliado

Outline

Outline

De qué es esta charla?

La palabra matemática en programación competitiva se suele usar para dos cosas, y vale la pena distinguirlas bien:

- Razonamiento matemático
- Problemas de matemática

De qué es esta charla?

- La primera? Se adquiere inevitablemente con la práctica de resolver muchos problemas.
- La segunda? De esto se va a tratar la clase de hoy, además de mostrar como hacer en código estas cosas.

Outline

Evitar usar floats/doubles

Evitar usar doubles, usar siempre enteros en lo posible.

En vez de

```
if(a/c >= b/d)...
```

Usar

```
if(a*d >= b*c)...
```

Si quiero hacer la división techo en enteros, en vez de usar `ceil(a/b)` podemos hacer `(a+b-1)/b`

Si queremos ver si un número i es menor o igual a la raíz de n, en lugar de hacer `i<=sqrt(n)`, podemos hacer `i*i<=n`

Raíz piso de un número

Para obtener la raíz piso de un número (la raíz redondeada para abajo), podemos hacer lo siguiente:

```
ll raiz_piso(ll n){
    ll r=max(sqrt(n)-3,0);
    while(r*r<=n)r++;
    return r-1;
}
```

En lugar de usar `ll(sqrt(n))` que pierde precisión y puede dar un resultado incorrecto.

Outline

Divisibilidad

Definition

Vamos a decir que $a \mid b$ (a divide a b) si existe un entero k tal que $a \cdot k = b$.

Equivalentemente, b es un múltiplo de a.
Si $a \neq 0$ esto es decir que la división b/a tiene resultado entero.

Por ejemplo $2 \mid 4$ y $5 \mid 15$ pero $4 \nmid 6$.

Notar que cualquier número divide a 0, es decir 0 es múltiplo de cualquier número.

Divisores y números primos

Definition

Vamos a decir que d es un divisor de n si $d \mid n$.

Los divisores de 10 son 1, 2, 5, 10.

Definition

Vamos a decir que p es un número primo si tiene exactamante dos divisores distintos, 1 y p.

5 y 7 son números primos, pero 6 y 1 no.

Theorem

Todo número entero se puede escribir como un único producto de primos, y esto se llama la factorización en primos de ese número.

La factorización en primos de 10 es $2 \cdot 5$, y la de 12 es $2^2 \cdot 3$.

Cómo hago estas tres cosas en código?

```
bool es_primo(n); // decidir si n es primo

vector<ll> divisores(n); // devolver todos los divisores

vector<pair<ll,ll>> factorizar(n);
// devolver la factorización en primos de n
```

Cómo hago estas tres cosas en código?

Con una simple observación, toda la información que necesitamos se puede obtener iterando hasta la raíz del número, obteniendo una complejidad de $O(\sqrt{n})$ en todos los casos, en vez de $O(n)$, que es mucho mejor.

Para decidir si un número es primo o no, basta con ver si algún número $\leq \sqrt{n}$ lo divide:

```
bool es_primo(n){
    for(ll i=2; i*i<=n; i++){
        if(n%i==0) return false;
    }
    return true;
}
```

Cómo implemento la función divisores?

- Si d es divisor de n, entonces n/d también lo es.
- Entonces si encontramos d también encontramos n/d.
- Alguno de d o n/d tiene que ser $\leq \sqrt{n}$.

Y ese es nuestro algoritmo.

Caso borde:

- Cuando $d = \sqrt{n}$ podemos estar agregando divisores repetidos.

Ejercicio:

- Implementar divisores(n).

Factorización en primos

```
vector<pair<ll,ll>> factorizar(n){
    vector<pair<ll,ll>> res;
    for(ll p=2; p*p <= n; p++){
        ll k=0;
        while(n%p==0){
            k++;
            n/=p;
        }
        if(k>0)res.push_back({p,k});
    }
    if(n>1)res.push_back({n,1});
    return res;
}
```

GCD y LCM

- GCD = "Greatest common divisor"
- $gcd(4, 6) = 2$, $gcd(8, 15) = 1$, $gcd(3, 9) = 3$
- Cuando el GCD de dos números es 1 se dice que estos dos números son coprimos, por ejemplo 8 y 15 son coprimos, pero 10 y 8 no.
- Notar que 1 es coprimo con cualquier número.

LCM

- LCM = "Lowest common multiple"
- $lcm(10, 15) = 30$, $lcm(2, 3) = 6$, $lcm(7, 14) = 14$
- Notar que $lcm(1, x) = x$.
- Notar también que $lcm(a, b) = a \cdot b \ / \ gcd(a, b)$

Example

$lcm(10, 15) = 10 \cdot 15 \ / \ gcd(10, 15)$
$lcm(10, 15) = 150 \ / \ 5 = 30$

Código de LCM y GCD

A partir de C++17, ambas operaciones, gcd y lcm vienen implementadas en la stl.
Escribiendo simplemente `gcd(a,b)` nos dará el gcd de a y b, y lo mismo para lcm.

Antes de C++17:
Podemos usar `__gcd`.
Para lcm podemos usar que `lcm(a,b) = a*b / __gcd(a,b)`

IMPORTANTE: La complejidad de llamar tanto a gcd(a,b) como a lcm(a,b) es O(log(a)),

Criba de eratóstenes

```
int cr[MAXV];
void init_sieve(){ // O(MAXV loglog(MAXV))
 memset(cr,-1,sizeof(cr));
 fore(i,2,MAXV)if(cr[i]<0){
        for(ll j=1ll*i*i;j<MAXV;j+=i)cr[j]=i;
    }
}
map<int,int> fact(int n){ // O(log(n))
 map<int,int> res;
 while(cr[n]>=0)res[cr[n]]++,n/=cr[n];
 if(n>1)res[n]++;
 return res;
}
```

Aritmética Modular

Definition

Decimos que a tiene resto r módulo m si al hacer la división entera a/m, nos queda con resto r.

Por ejemplo `14%3 = 2`

Definition

Decimos que $a \equiv b \pmod{m}$ si a y b tienen el mismo resto módulo m.

Por ejemplo $7 \equiv 12 \pmod{5}$

- Sumar: `(a+b)%MOD`
- En C++ ocurre lo siguiente: `-8%5 = -3`
- Restar: `((a-b)%MOD + MOD) % MOD`
- Multiplicar: `a*b%MOD`

Cómo dividir?

Por el pequeño teorema de Fermat sabemos que

Theorem

$b^{p-1} \equiv 1 \pmod{p}$ *con* p *primo.*

- $b^{p-2} \cdot b \equiv 1 \pmod{p}$
- b^{p-2} es el inverso de b módulo p.
- Dividir: `a * fpow(b,MOD-2) % MOD`

`fpow` es exponenciación binaria (próxima slide)

Exponenciación binaria

```
ll exp_binaria(ll a, ll b){
    ll res=1;
    while(b>0){
        if(b%2==1) res = res*a % MOD;
        a = a*a%MOD;
        b/=2;
    }
    return res;
}
```

Funciones modulares

```
int add(int a, int b){return (a+b)%MOD;}
int sub(int a, int b){return ((a-b)%MOD+MOD)%MOD;}
int mul(ll a, ll b){return a*b%MOD;}
int fpow(int a, ll b){
    if(b<0)return 0;
    int res=1;
    while(b){
        if(b&1)res=mul(res,a);
        a=mul(a,a); b/=2;
    }
    return res;
}
```

Mejor constante para add y sub

```
int add(int a, int b){
    a+=b;
    if(a>=MOD)a-=MOD;
    return a;
}

int sub(int a, int b){
    a-=b;
    if(a<0)a+=MOD;
    return a;
}
```

Resumen (y copy paste)

```
int add(int a, int b){a+=b;if(a>=MOD)a-=MOD;return a;}
int sub(int a, int b){a-=b;if(a<0)a+=MOD;return a;}
int mul(ll a, ll b){return a*b%MOD;}
int fpow(int a, ll b){
    if(b<0)return 0;
    int res=1;
    while(b){if(b%2)res=mul(res,a); b/=2; a=mul(a,a);}
    return res;
}
```

Outline

Combinatoria

Theorem

Principio de multiplicación: Si tengo 5 *remeras de colores diferentes y* 3 *pantalones de colores diferentes, tengo* $3 \times 5 = 15$ *combinaciones para vestirme*

Si tengo una carrera de 5 jugadores, de cuantas formas puede quedar la tabla de posiciones?
$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ formas posibles

Definition

Una permutación es un array de tamaño n con elementos distintos del 1 al n.

Por ejemplo, 5 4 1 3 2 es una permutación de tamaño 5.

- Cuantas permutaciones hay de tamaño n?

Factorial con repetición

Example

Dada una palabra, cuantas palabras distintas existen reordenando las letras de esta palabra?

`mama:`

`aamm`
`amam`
`amma`
`maam`
`mama`
`mmaa`

6 palabras distintas

Factorial con repetición

Example

Dada una palabra, cuantas palabras distintas existen reordenando las letras de esta palabra?

`nan:`

`anN`
`aNn`
`naN`
`Nan`
`nNa`
`Nna`

$3! = 6$

Factorial con repetición

Example

Dada una palabra, cuantas palabras distintas existen reordenando las letras de esta palabra?

`nan:`

`anN`
`aNn`

`naN`
`Nan`

`nNa`
`Nna`

$\frac{3!}{2!} = 3$ palabras distintas

Factorial con repetición

Example

Dada una palabra, cuantas palabras distintas existen reordenando las letras de esta palabra?

Si tengo 3 veces la letra a, 4 veces la letra b y 2 veces la letra c, tengo $\frac{(3+4+2)!}{3!\ 4!\ 2!} = \frac{9!}{3!\ 4!\ 2!}$ palabras distintas.

En general, si tengo a_i veces la i-ésima letra, tengo $\frac{S!}{a_1!a_2!...a_n!}$ combinaciones, donde S es el largo de la palabra, y n la cantidad de letras distintas.

Números combinatorios

Example

Si hay 10 personas en aula, cuantos grupos distintos de 4 personas se pueden formar?

```
1 2 3 4 5 6 7 8 9 10
0 1 1 0 1 0 0 1 0  0
```

Corresponde al grupo

```
{2, 3, 5, 8}
```

Entonces tengo 6 ceros y 4 unos, osea $\frac{(4+6)!}{4!\ 6!} = \frac{10!}{4!\ 6!}$ formas de reordenar los unos y ceros, y por lo tanto grupos de 4 personas.

Estos se llaman números combinatorios, donde $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ denota la cantidad de formas de elegir k elementos de un grupo de n elementos.

Como computar los números combinatorios

- En $O(n^2)$:

Uso la recursión $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ Con $\binom{0}{0} = 1$ y $\binom{n}{k} = 0$ para $n < 0$, $k < 0$ o $n < k$.

- En $O(n)$ precomputo y $O(1)$ query:

Precomputo los factoriales hasta `MAXN`.
También precomputo los inversos de los factoriales en otro arreglo:
Calculo el inverso de `MAXN-1`! (inverso modular), hago un for para atrás y uso que $\frac{1}{n!} = \frac{1}{(n+1)!} \cdot (n+1)$.
Y luego uso la fórmula $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ en $O(1)$ dado que ya tengo calculados $n!$, $\frac{1}{k!}$ y $\frac{1}{(n-k)!}$.

Código de combinatorios $O(1)$ query

```
ll fc[MAXF],fci[MAXF];
void factoriales(){
fc[0]=1;
fore(i,1,MAXF)fc[i]=mul(fc[i-1],i);
fci[MAXF-1]=fpow(fc[MAXF-1],MOD-2);
for(ll i=MAXF-2;i>=0;i--)fci[i]=mul(fci[i+1],(i+1));
}

ll comb(ll n, ll k){ // must call factoriales before
if(n<0||k<0||k>n)return 0;
return mul(mul(fc[n],fci[k]),fci[n-k]);
}
```

Consultas

Pueden consultarme durante esta semana, o al telegram, o me pueden enviar un mail a:

- eduardocarranzavelez@gmail.com