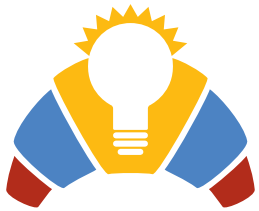
Matematica

Inicial

Joaquín Inama

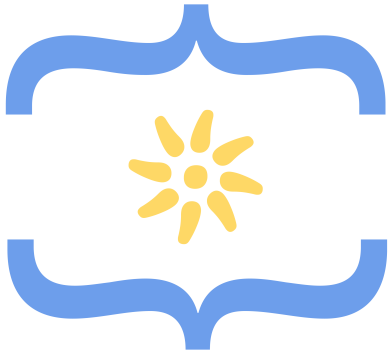
Universidad Nacional de La Plata – Facultad de Ciencias Exactas

Training Camp 2026



Gracias Sponsors!

Organiza



Facultad de INFORMÁTICA

UNIVERSIDAD NACIONAL DE LA PLATA

Diamond Sponsor

Outline

Outline

Divisibilidad

Definition (Divisibilidad)

Un número entero a es divisible por otro número entero b si existe un número entero k tal que $a = b \cdot k$.
En este caso, decimos que b es un divisor de a, y escribimos $b|a$.

Definition (Resto)

$a\%b$ es el resto de a en la división por b.
Por ejemplo $67\%10 = 7$ y $30\%4 = 2$.
$a\%b == 0$ es true si a es divisible por b.

Teorema Fundamental de la Aritmética

Definition (Número primo)

Un número primo es un número entero mayor que 1 que tiene exactamente dos divisores positivos distintos: 1 y él mismo. Por ejemplo, los primeros números primos son: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...

Theorem (Teorema Fundamental de la Aritmética)

Todo número entero mayor que 1 puede ser expresado de manera única (excepto por el orden de los factores) como un producto de números primos.

$$N = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$$

- Ejemplo: $60 = 2^2 \cdot 3^1 \cdot 5^1$
- Ejemplo: $84 = 2^2 \cdot 3^1 \cdot 7^1$

Fórmulas útiles

Conociendo la factorización en primos de un número N, podemos calcular la cantidad de divisores y la suma de divisores de N usando las siguientes fórmulas:

- Cantidad de divisores de N:

$$d(N) = (\alpha_1 + 1)(\alpha_2 + 1) \cdots (\alpha_k + 1)$$

- Suma de divisores de N:

$$\sigma(N) = \frac{p_1^{\alpha_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{\alpha_2+1} - 1}{p_2 - 1} \cdot \ldots \cdot \frac{p_k^{\alpha_k+1} - 1}{p_k - 1}$$

Máximo Comúm Divisor (MCD) y Mínimo Común Múltiplo (MCM)

Definition (Máximo Común Divisor (MCD))

El MCD de dos números enteros a y b, denotado como $\gcd(a, b)$, es el mayor número entero que divide a ambos.

Definition (Mínimo Común Múltiplo (MCM))

El MCM de dos números enteros a y b, denotado como $\text{lcm}(a, b)$, es el menor número entero positivo que es múltiplo de ambos.

Example

Por ejemplo, para $a = 12$ y $b = 18$, tenemos:

$$\gcd(12, 18) = 6, \quad \text{lcm}(12, 18) = 36$$

Máximo Comúm Divisor (MCD) y Mínimo Común Múltiplo (MCM)

Código C++

```
mcd = gcd(a,b); // O(log n) C++17
mcm = a/gcd(a,b)*b; // Relación entre MCD y MCM
```

Test de primalidad

Definition (Propiedad)

Los divisores de N vienen de a pares. Si d es un divisor de N, entonces $\frac{N}{d}$ también es un divisor de N.

Example

Por ejemplo, los divisores de 60 son: $1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60$.

- Para determinar si un número N es primo, podemos probar si es divisible por algún número menor o igual a $\sqrt{N}$.
- Ejemplo: Para verificar si 60 es primo, probamos con los números enteros menores o iguales a $\sqrt{60} \approx 7.75$.

Criba de Eratóstenes

La Criba de Eratóstenes es un algoritmo eficiente para encontrar todos los números primos hasta un número entero dado N.

El algoritmo funciona de la siguiente manera:

- Crear una lista de números enteros desde 2 hasta N.
- Comenzando con el primer número primo $p = 2$
- Marcar todos los múltiplos de p (excepto p mismo) como no primos.
- Encontrar el siguiente número no marcado en la lista y repetir el proceso hasta que todos los números hayan sido procesados.
- Los números que permanecen sin marcar en la lista son los números primos hasta N.

Ejemplo: criba hasta 30

2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	

Azul: primo **Rojo: compuesto**

1. Inicialización

Escribimos todos los candidatos desde 2 hasta 30.

Ejemplo: criba hasta 30

2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	

Azul: primo **Rojo: compuesto**

2. Procesamos $p = 2$

Como 2 no está marcado, es primo. Descartamos $2^2, 2^2 + 2, 2^2 + 2 \cdot 2, \ldots$

Ejemplo: criba hasta 30

2	**3**	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	

Azul: primo **Rojo: compuesto**

3. Procesamos $p = 3$

El siguiente candidato no marcado es 3. Descartamos sus múltiplos desde 3^2: $9, 15, 21, 27$.

Ejemplo: criba hasta 30

2	**3**	4	**5**	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	

Azul: primo **Rojo: compuesto**

4. Procesamos $p = 5$

El siguiente candidato es 5 y el único múltiplo nuevo es $5^2 = 25$. Luego $7^2 > 30$, así que podemos detenernos.

Ejemplo: criba hasta 30

2	**3**	4	**5**	6	**7**	8	9	10	**11**
12	**13**	14	15	16	**17**	18	**19**	20	21
22	**23**	24	25	26	27	28	**29**	30	

Azul: primo **Rojo: compuesto**

5. Resultado

Los números no descartados son $2, 3, 5, 7, 11, 13, 17, 19, 23$ y 29.

Implementación (Ejemplo de Código)

Código C++

```
vector<int> min_prime; // min_prime[i] contiene el menor
// primo que divide a i, util para factorizar
// en log(i)

vector<int> criba(int n) { // O(n log log n)
    vector<bool> prime(n+1,true);
    min_prime.resize(n+1,INF);
    vector<int> primos;
    for(int p=2; p*p<=n; p++){
      if(!prime[p]) continue;
      primos.push_back(p), min_prime[p] = p;
        for(int i=p*p; i<=n; i += p) {
            prime[i] = false;
            min_prime[i] = min(min_prime[i],p);
        }
    }
    return primos; // lista de primos hasta n
}
```

Factorizar un número en primos

Código C++

```
vector<pair<int,int>> factorizar(int n) { // O(log n)
    vector<pair<int,int>> fact;
    while(n > 1) {
        int p = min_prime[n], cnt = 0;
        while(n % p == 0) n /= p, cnt++;
        fact.push_back({p, cnt});
    }
    return fact; // lista de factores primos de n
}
```

Hallar los divisores de un número

Código C++

```
vector<int> findDivisors(int n) { // O(sqrt(n))
  vector<pair<int,int>> fact = factorizar(n);
  vector<int> divisors = {1};
  for(auto [p, cnt] : fact) {
      int sz = divisors.size();
      int x = 1;
      for(int j=0; j<cnt; j++) {
        x *= p;
        for(int i=0; i<sz; i++) {
          divisors.push_back(divisors[i] * x);
        }
      }
  }
  return divisors; // lista de divisores de n
}
```

Ejemplo: `findDivisors(60)`

$$60 = 2^2 \cdot 3 \cdot 5 \qquad \texttt{fact} = [(2,2),(3,1),(5,1)]$$

Azul: valores recién agregados

1. Estado inicial

La función comienza con el divisor neutro:

$$\texttt{divisors} = \boxed{1}$$

Ejemplo: `findDivisors(60)`

$$60 = 2^2 \cdot 3 \cdot 5 \qquad \texttt{fact} = [(2,2),(3,1),(5,1)]$$

Azul: valores recién agregados

2. Procesamos $p = 2$: primera potencia

`cnt` $= 2$, `sz` $= 1$ y `x` $= 2$. Agregamos $1 \cdot 2 = 2$.

`divisors` = | 1 | **2** |

Ejemplo: `findDivisors(60)`

$$60 = 2^2 \cdot 3 \cdot 5 \qquad \texttt{fact} = [(2,2),(3,1),(5,1)]$$

Azul: valores recién agregados

3. Procesamos $p = 2$: segunda potencia

Ahora $\texttt{x} = 4$. Como `sz` sigue siendo 1, agregamos solamente $1 \cdot 4 = 4$.

`divisors` = | 1 | 2 | **4** |

Ejemplo: `findDivisors(60)`

$$60 = 2^2 \cdot 3 \cdot 5 \qquad \texttt{fact} = [(2,2),(3,1),(5,1)]$$

Azul: valores recién agregados

4. Procesamos $p = 3$

Fijamos $\texttt{sz} = 3$ y agregamos $1 \cdot 3$, $2 \cdot 3$ y $4 \cdot 3$.

`divisors =` | 1 | 2 | 4 | **3** | **6** | **12** |

Ejemplo: `findDivisors(60)`

$$60 = 2^2 \cdot 3 \cdot 5 \qquad \texttt{fact} = [(2,2),(3,1),(5,1)]$$

Azul: valores recién agregados

5. Procesamos $p = 5$ y terminamos

Con $\texttt{sz} = 6$, multiplicamos por 5 los seis valores existentes.

`divisors =`	1	2	4	3	6	12
	5	**10**	**20**	**15**	**30**	**60**

El vector contiene todos los divisores, aunque no está ordenado.

Outline

Aritmética Modular

Trabajar módulo m significa conservar solamente el resto. Dos enteros son **congruentes** cuando tienen el mismo resto al dividir por m.

Congruencia

$$a \equiv b \pmod{m} \iff m \mid (a-b).$$

Operaciones compatibles

Podemos reducir los operandos antes de sumar, restar o multiplicar:

$$(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m,$$
$$(a-b) \bmod m = ((a \bmod m) - (b \bmod m)) \bmod m,$$
$$(a \cdot b) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m.$$

Detalles importantes

En C++, el resto puede ser negativo: `-8 % 5 = -3`. Para obtener un valor entre 0 y $m-1$, usamos `((x % m) + m) % m`. La división requiere un inverso modular.

División modular e inversos

En aritmética modular no podemos dividir de manera directa. Para dividir por b, multiplicamos por su **inverso modular**:

$$\frac{a}{b} \equiv a \cdot b^{-1} \pmod{m}, \qquad b \cdot b^{-1} \equiv 1 \pmod{m}.$$

El inverso de b módulo m existe si y solo si $\gcd(b, m) = 1$.

Teorema de Euler–Fermat

Si $\gcd(b, m) = 1$, entonces $b^{\varphi(m)} \equiv 1 \pmod{m}$. Por lo tanto, $b^{-1} \equiv b^{\varphi(m)-1} \pmod{m}$.

Caso primo

Si $m = p$ es primo, entonces $\varphi(p) = p - 1$. Para $b \not\equiv 0 \pmod{p}$:

$$b^{-1} \equiv b^{p-2} \pmod{p}.$$

Esta potencia se calcula eficientemente con exponenciación binaria.

Exponenciación binaria

Calcular a^b multiplicando a un total de b veces cuesta $\mathcal{O}(b)$ operaciones.

La exponenciación binaria aprovecha que podemos dividir el exponente por 2 en cada paso:

$$a^b = \begin{cases} \left(a^{b/2}\right)^2, & \text{si } b \text{ es par,} \\ a\left(a^{(b-1)/2}\right)^2, & \text{si } b \text{ es impar.} \end{cases}$$

- Si b es impar, multiplicamos la respuesta por la base actual.
- Elevamos la base al cuadrado y dividimos b por 2.
- Repetimos hasta que $b = 0$.

Complejidad

Como el exponente se reduce a la mitad, el algoritmo cuesta $\mathcal{O}(\log b)$ tiempo y $\mathcal{O}(1)$ memoria.

Ejemplo: calcular 3^{13}

Primero observamos que

$$13 = (1101)_2 = 8 + 4 + 1, \qquad 3^{13} = 3^8 \cdot 3^4 \cdot 3.$$

b	base	resultado	acción
13	3	3	impar: multiplicar por 3
6	9	3	par: no multiplicar
3	81	243	impar: multiplicar por 81
1	6561	1594323	impar: multiplicar por 6561

En cada fila elevamos la base al cuadrado y reemplazamos b por $\lfloor b/2 \rfloor$.

Example

El resultado final es $3^{13} = 1594323$, usando solamente cuatro iteraciones.

Exponenciación binaria: código C++

Potencia modular

```
long long binpow(long long a, long long b) {
    long long result = 1;

    while (b > 0) {                 // O(log b)
        if (b % 2 == 1) {
            result = result * a % MOD;
        }
        a = a * a % MOD;
        b /= 2;
    }
    return result;
}
```

La versión modular evita que el resultado crezca innecesariamente. Se asume que cada producto entra en un `long long`.

Funciones modulares

Implementación general

```
int add(int a, int b) return (a + b) % MOD;
int sub(int a, int b) return ((a - b) % MOD + MOD) % MOD;
int mul(long long a, long long b) return a * b % MOD;

int fpow(int a, int b) {
    long long r = 1;
    a %= MOD;
    while (b > 0) {
        if (b & 1) r = mul(r, a);
        a = mul(a, a); b /= 2;
    }
    return r;
}
int div(int a, int b) return mul(a, fpow(b, MOD - 2));
```

Funciones modulares: suma y resta rápidas

Precondición

Estas versiones se pueden usar cuando $0 \leq a, b < \texttt{MOD}$.

Implementación con un `if`

```
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;
}
```

Evitamos usar %, que suele ser más costoso que una comparación y una suma o resta.

Outline

Factorial

Definición

Para un entero $n \geq 1$ definimos

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1, \qquad 0! = 1.$$

Ejemplo

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

$n!$ cuenta cuántas formas hay de ordenar n objetos distintos: hay n opciones para el primer lugar, $n-1$ para el segundo, y así sucesivamente.

Permutaciones de una palabra

Si una palabra tiene n letras todas distintas, puede ordenarse de

$$n!$$

maneras diferentes.

Ejemplo: SOL

Hay $3! = 6$ permutaciones:

$$\texttt{SOL},\ \texttt{SLO},\ \texttt{OSL},\ \texttt{OLS},\ \texttt{LSO},\ \texttt{LOS}.$$

¿Qué cambia cuando algunas letras son iguales? Intercambiarlas no produce una palabra nueva.

Permutaciones con letras repetidas

Primero imaginemos que las dos letras A de CASA tienen etiquetas:

$$\mathtt{C}\, A_1\, \mathtt{S}\, A_2.$$

Estas dos permutaciones etiquetadas son diferentes:

$$\mathtt{C}\, A_1\, \mathtt{S}\, A_2 \qquad \text{y} \qquad \mathtt{C}\, A_2\, \mathtt{S}\, A_1,$$

pero al borrar colores y etiquetas ambas son CASA. Por eso cada palabra fue contada 2! veces y debemos dividir:

$$\frac{4!}{2!} = 12.$$

En general, si las repeticiones son $c_1, c_2, \ldots, c_k$:

$$\boxed{\frac{n!}{c_1!c_2!\cdots c_k!}}.$$

Ejemplo: permutaciones de BANANA

Para distinguir temporalmente las letras iguales, las coloreamos:

$$\mathrm{B}\, A_1\, N_1\, A_2\, N_2\, A_3.$$

- Hay 6! órdenes si todas las letras tienen etiqueta.
- Las tres A se pueden intercambiar de 3! formas sin cambiar la palabra.
- Las dos N se pueden intercambiar de 2! formas sin cambiarla.

Por lo tanto, la cantidad de palabras distintas es

$$\frac{6!}{3!\,2!} = \frac{720}{12} = \boxed{60}.$$

Número combinatorio

El número combinatorio $\binom{n}{k}$ cuenta cuántas formas hay de elegir k elementos de un conjunto de n, sin importar el orden:

$$\boxed{\binom{n}{k} = \frac{n!}{k!(n-k)!}}.$$

Ejemplo

Para elegir 3 representantes entre 5 estudiantes:

$$\binom{5}{3} = \frac{5!}{3!\,2!} = 10.$$

Elegir a Ana, Beto y Carla es lo mismo que elegir a Carla, Ana y Beto; por eso dividimos por 3!.

Dos identidades útiles:

$$\binom{n}{k} = \binom{n}{n-k}, \qquad \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

Número combinatorio: código C++

Precomputación y consulta

```
vector<int> fact, invFact;
void init_fact(int n) {
    fact.assign(n + 1, 1);
    invFact.assign(n + 1, 1);
    for (int i = 1; i <= n; i++)
        fact[i] = mul(fact[i - 1], i);
    invFact[n] = div(1, fact[n]);
    for (int i = n; i > 0; i--)
        invFact[i - 1] = mul(invFact[i], i);
}

int comb(int n, int k) {
    if (k < 0 || k > n) return 0;
    return mul(fact[n], mul(invFact[k], invFact[n - k]));
}
```

Cajitas y bolitas: se permiten cajas vacías

Queremos repartir n objetos iguales entre k cajas distintas. Esto equivale a contar las soluciones de

$$x_1 + x_2 + \cdots + x_k = n, \qquad x_i \geq 0.$$

Usamos n estrellas y $k-1$ barras. Por ejemplo,

$$\underbrace{\star\star}_{x_1=2} \mid \underbrace{\star\star\star}_{x_2=3} \mid \underbrace{\star\star}_{x_3=2}$$

representa $(x_1, x_2, x_3) = (2, 3, 2)$.

Ejemplo: 7 caramelos entre 3 personas

Hay 7 estrellas y 2 barras. Elegimos las posiciones de las barras:

$$\boxed{\binom{7+3-1}{3-1} = \binom{9}{2} = 36}.$$

Cajitas y bolitas: al menos una por caja

Ahora contamos las soluciones de

$$x_1 + x_2 + \cdots + x_k = n, \qquad x_i \geq 1.$$

Colocamos primero un objeto en cada caja. Quedan $n - k$ objetos para repartir sin restricciones, así que

$$\boxed{\binom{n-1}{k-1}}.$$

Ejemplo: 10 caramelos entre 4 personas

Si cada persona debe recibir al menos uno, entregamos primero 4 caramelos y distribuimos libremente los 6 restantes:

$$\binom{6+4-1}{4-1} = \binom{9}{3} = \boxed{84}.$$

Importante: estas fórmulas requieren objetos indistinguibles y cajas distinguibles.

Outline

Problemas para entrenar

CSES Problem Set

1. Exponentiation — exponenciación binaria
2. Counting Divisors — divisores
3. Binomial Coefficients — combinatorios
4. Creating Strings II — permutaciones
5. Distributing Apples — cajitas y bolitas
6. Common Divisors — MCD y múltiplos

Codeforces

- 26A – Almost Prime — factores primos
- 630C – Lucky Numbers — conteo y potencias

¡Gracias!

¿Preguntas?