Combinatoria

Por Ariel Parra

[Γα=Ω5]

La regla de la suma (sum rule)

Si dos conjuntos A y B son disjuntos (es decir, $A \cap B = \emptyset$), entonces la cantidad total de elementos en la unión de A y B es igual a la suma de los elementos en A y los elementos en B, es decir:

$$A \cap B = \emptyset \Rightarrow |A| + |B| = |A \cup B|$$

En palabras más sencillas: si tenemos A formas de realizar la Tarea 1 y B formas de realizar la Tarea 2, entonces el número total de formas de elegir una de las dos tareas es igual a $A + B$.

En general, si hay N tareas y la i-ésima tarea se puede realizar de $a[i]$ maneras, entonces hay $a_1 + a_2 + a_3 + \cdots + a_n$ formas de hacer una de las tareas.

Ejemplo:

Imagina que tienes 3 sombreros diferentes, 2 camisas diferentes y 4 pantalones diferentes. Si quieres donar uno de los artículos, ¿cuántas formas diferentes hay de hacerlo?

Respuesta:

$$3 + 2 + 4 = 9$$

La regla del producto (product rule)

El número total de elementos en el producto cartesiano de dos conjuntos A y B es igual al producto del número de elementos en A y en B, es decir:

$$|A| \cdot |B| = |A \times B|$$

En palabras más sencillas: si tenemos A formas de realizar la Tarea 1 y B formas de realizar la Tarea 2, entonces el número total de formas de hacer ambas tareas es igual a $A \times B$.

En general, si hay N tareas y la i-ésima tarea se puede realizar de $a[i]$ maneras, entonces hay $a_1 \times a_2 \times a_3 \times \cdots \times a_n$ formas de hacer todas las tareas.

Ejemplo:

Imagina que tienes 3 sombreros diferentes, 2 camisas diferentes y 4 pantalones diferentes. Quieres vestirte y, para ello, debes usar 1 sombrero, 1 camisa y 1 pantalón. ¿De cuántas maneras puedes hacerlo?

Solución:

$$3 \times 2 \times 4 = 24$$

Los 4 Tipos de Permutaciones

1. **Permutación con repetición**

   Cuando se pueden repetir elementos en cada posición de la permutación.

   Ejemplo: ¿Cuántos números de 3 dígitos mayores que 500 se pueden formar con los dígitos 3, 4, 5 y 7?

   Solución: El primer dígito debe ser 5 o 7 (2 opciones), y los otros 2 pueden repetirse (4 opciones cada uno). Total de permutaciones: $2 \times 4 \times 4 = 32$.

2. **Permutación sin repetición**

   Cuando no se permiten repeticiones y cada elemento solo puede aparecer una vez en la permutación.

   Ejemplo: ¿Cuántos números de 3 dígitos divisibles por 3 se pueden formar con los dígitos 2, 4, 6 y 8 sin repetición?

   Solución: La suma de los dígitos debe ser divisible por 3. Existen dos combinaciones posibles (2, 4, 6 y 4, 6, 8), con $3! = 6$ permutaciones cada una. Total: $6 + 6 = 12$.

3. **r-permutación sin repetición**

   Disposición de *r* elementos de un total de *n*, sin repetir ningún elemento.

   Ejemplo: Una heladería tiene 10 sabores. ¿Cuántas combinaciones de 3 sabores diferentes se pueden hacer?

   Solución: Para el primer sabor hay 10 opciones, para el segundo 9, y para el tercero 8. Total: $10 \times 9 \times 8 = 720$. Fórmula general: $n \times (n-1) \times (n-2) = nPr$.

4. **r-permutación con repetición**

   Disposición de *n* elementos en *r* posiciones, permitiendo que los elementos se repitan.

   Ejemplo: Un policía visita una escena del crimen 3 veces a la semana. ¿Cuántas maneras hay de programar sus visitas sin restricción de días?

   Solución: Para cada visita hay 7 opciones (uno por día de la semana). Total de maneras: $7 \times 7 \times 7 = 343$.

Definición de Combinatoria

En combinatoria, una **combinación** de un conjunto de objetos distintos es simplemente el conteo de formas en las que se pueden seleccionar un número específico de elementos de un conjunto de cierto tamaño. En una combinación, el orden de los elementos no importa. Una selección no ordenada de *r* elementos de un conjunto se llama una **r-combinación** y se representa como $C(n, r)$.

Dado que una combinación es simplemente una permutación sin orden, el número de **r-combinaciones** se puede expresar en términos de **r-permutaciones**. La **r-permutación** se puede obtener al primero calcular la **r-combinación** y luego ordenar los elementos en cada r-combinación, lo cual se puede hacer de $P(r, r)$ maneras.

La relación es:

$$P(n, r) = C(n, r) \times P(r, r)$$

Usando esta relación, la fórmula para $ C(n, r) $ se obtiene como:

$$C(n, r) = \frac{P(n, r)}{P(r, r)}$$

$$= \frac{\frac{n!}{(n-r)!}}{\frac{r!}{r!}} = \frac{n!}{r!(n-r)!}$$

Así, el número de combinaciones de *r* elementos de un conjunto de *n* se calcula usando la fórmula:

$$C(n, r) = \frac{n!}{r!(n-r)!}$$

Ejemplo de combinatoria

Como ejemplo, consideremos el problema de contar el número de formas de representar un número entero n como una suma de enteros positivos. Por ejemplo, hay 8 representaciones para el número 4:

- $1 + 1 + 1 + 1$
- $1 + 1 + 2$
- $1 + 2 + 1$
- $2 + 1 + 1$
- $2 + 2$
- $3 + 1$
- $1 + 3$
- 4

Un problema combinatorio como este se puede resolver a menudo usando una función recursiva. En este caso, podemos definir una función $f(n)$ que da el número de representaciones de n. Según el ejemplo anterior, $f(4) = 8$. Los valores de la función pueden calcularse recursivamente de la siguiente manera:

$$f(n) = \begin{cases} 1 & \text{si } n = 0 \\ f(0) + f(1) + \cdots + f(n-1) & \text{si } n > 0 \end{cases}$$

El caso base es $f(0) = 1$, porque la suma vacía representa el número 0. Luego, si $n > 0$, consideramos todas las formas de elegir el primer número de la suma. Si el primer número es k, hay $f(n-k)$ representaciones para la parte restante de la suma. Así, calculamos la suma de todos los valores de la forma $f(n-k)$ donde $k < n$.

Los primeros valores de la función son:

$$f(0) = 1$$
$$f(1) = 1$$
$$f(2) = 2$$
$$f(3) = 4$$
$$f(4) = 8$$

A veces, una fórmula recursiva puede reemplazarse por una fórmula de forma cerrada. En este problema, la función se puede expresar como:

$$f(n) = 2^{n-1}$$

> Esta fórmula se basa en el hecho de que hay $n-1$ posiciones posibles para colocar los signos "+" en la suma, y podemos elegir cualquier subconjunto de esas posiciones.

Coeficientes binomiales

El coeficiente binomial $\binom{n}{k}$ (pronunciado como "n elige k" o, a veces, escrito como ${}_nC_k$) representa el número de maneras de elegir un subconjunto de k elementos de un conjunto de n elementos. Por ejemplo, $\binom{4}{2} = 6$, porque el conjunto (set) $\{1, 2, 3, 4\}$ tiene 6 subconjuntos (subsets) de 2 elementos:

$$\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}$$

Hay dos formas de calcular coeficientes binomiales:

- Método 1: Triángulo de Pascal (DP), $O(n^2)$
- Método 2: Definición Factorial (Inversos Modulares), $O(n + \log \mathrm{MOD})$

Ahora veremos la implementación del segundo metodo, ya que es más probable que un problema de coeficientes binomiales tenga numeros grandes y modulos a que sea uno sencillo capaz de correr en $O(n^2)$.

```cpp
const int MAXN = 1e6;                const int MOD = 1e9 + 7;
vector<ll> fac(MAXN + 1);            vector<ll> inv(MAXN + 1);
ll exp(ll x, ll n, ll m) {
        x %= m;
        ll res = 1;
        while (n > 0) {
                if (n % 2 == 1) { res = res * x % m; }
                x = x * x % m;
                n >>= 1;// n /=2
        }
        return res;
}
void factorial() {
        fac[0] = 1;
        for (int i = 1; i <= MAXN; i++) { fac[i] = fac[i - 1] * i % MOD; }
}
void inverses() {
        inv[MAXN] = exp(fac[MAXN], MOD - 2, MOD);
        for (int i = MAXN; i >= 1; i--) { inv[i - 1] = inv[i] * i % MOD; }
}
ll choose(int n, int r) { return fac[n] * inv[r] % MOD * inv[n - r] % MOD; }
```

Codigo principal

```
int main() {
        factorial();
        inverses();
        int n;
        cin >> n;
        for (int i = 0; i < n; ++i) {
                int a, b;
                cin >> a >> b;
                cout << choose(a, b) << '\n';
        }
}
```

Derangements: Inclusion-exclusion principle

Supongamos que tenemos eventos $E_1, E_2, \ldots, E_n$, donde el evento E_i corresponde a que la persona i recibe su propio sombrero. Queremos calcular $n! - |E_1 \cup E_2 \cup \cdots \cup E_n|$.

Restamos de $n!$ el número de formas en las que puede ocurrir cada evento; es decir, consideremos la cantidad $n! - |E_1| - |E_2| - \cdots - |E_n|$. Esto subcuenta, ya que estamos restando los casos donde ocurren más de un evento demasiadas veces. Específicamente, para una permutación donde ocurren al menos dos eventos, subcontamos una vez. Entonces, añadimos de nuevo el número de formas en las que ocurren dos eventos. Podemos continuar este proceso para cada tamaño de subconjunto de índices. La expresión queda como:

$$n! - |E_1 \cup E_2 \cup \cdots \cup E_n| = \sum_{k=1}^{n} (-1)^k \cdot (\text{´mero de permutaciones con } k \text{ puntos fijos})$$

Para un conjunto de tamaño k, el número de permutaciones con al menos k índices se puede calcular eligiendo un conjunto de tamaño k que sea fijo y permutando los otros índices. En términos matemáticos

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

Entonces, el problema se reduce a calcular:

$$n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$$

En código se vería así:

```cpp
int main() {
    int n, m;
    cin >> n >> m;

    int c = 1;  // Start with c = 1
    for (int i = 1; i <= n; i++) {
        c = (1LL * c * i) % m;  // Multiply by i and take mod m
        c = (c + (i % 2 == 1 ? -1 : 1) + m) % m;  // Add -1 or +1, and take mod m to handle negatives
        cout << c << ' ';
    }
    cout << endl;
}
```

Principio del Palomar (Pigeonhole Principle)

El **principio del palomar** establece que si distribuimos n palomas en m nidos y $n > m$, al menos un nido contendrá más de una paloma. En otras palabras, si tienes más objetos que contenedores en los cuales colocarlos, al menos un contenedor debe tener más de un objeto. Este principio se basa en la analogía de palomas (objetos) colocadas en agujeros de palomar (contenedores).

Versión Generalizada del Principio del Palomar

La versión generalizada del principio dice que si distribuimos $km + n$ palomas en m nidos, donde $n \geq 1$, entonces al menos un nido contendrá al menos $k + 1$ palomas. Este principio simple tiene numerosas aplicaciones en problemas de combinatoria y teoría de números, aunque a veces no es evidente identificar cómo aplicarlo correctamente, ya que puede ser difícil reconocer qué representan los "nidos" y las "palomas" en cada problema.

Ejemplos del Principio del Palomar

1. **Ejemplo de bolas de colores**: Supongamos que tenemos una bolsa con bolas de cinco colores diferentes. ¿Cuál es el número mínimo de bolas que debemos extraer, sin mirarlas, para asegurar que obtenemos dos bolas del mismo color?
    - Aplicación: Aquí, los "nidos" representan los colores de las bolas. Dado que hay cinco colores (5 nidos), si extraemos 6 bolas, al menos dos de ellas serán del mismo color.
2. **Ejemplo de equipos de fútbol**: Imagina que n equipos de fútbol juegan partidos cada fin de semana, enfrentándose todos contra todos. Demuestra que, al final de cada fin de semana, siempre hay dos equipos que han jugado el mismo número de partidos.
    - Aplicación: Aquí, los "nidos" son las posibles cantidades de partidos que un equipo ha jugado hasta el momento, desde 0 (si no ha jugado) hasta $n-1$ (si ha jugado contra todos los demás). Sin embargo, los valores 0 y $n-1$ son incompatibles porque no puede haber un equipo que haya jugado contra todos mientras otro no ha jugado ningún partido. Por lo tanto, el número de nidos efectivos es $n-1$. Con n equipos y solo $n-1$ nidos posibles, el principio del palomar asegura que al menos dos equipos han jugado el mismo número de partidos.

Numeros de Catalan

Los números de Catalán son una secuencia de números enteros positivos que se utilizan para contar diversas combinaciones posibles en problemas combinatorios. La fórmula general para el término C_n es:

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

Alternativamente, cada número de Catalán también puede calcularse a partir del término anterior usando:

$$C_n = C_{n-1} \times \frac{4n-2}{n+1}$$

Los primeros números de Catalán para $n = 0, 1, 2, 3, \ldots$ son: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ...

Los números de Catalán aparecen en muchos problemas de conteo interesantes, tales como:

1. Contar el número de expresiones con n pares de paréntesis correctamente emparejados. Por ejemplo, para $n = 3$, las expresiones posibles son $()(()), ()(()), ()()(), (())(), (()())$.
2. Contar el número de árboles de búsqueda binarios posibles con n claves.
3. Contar el número de árboles binarios completos con $n + 1$ hojas.
4. Dado un número n, contar el número de formas de dibujar n cuerdas en un círculo con $2n$ puntos de manera que no se crucen dos cuerdas.

Algoritmo para imprimir los primeros n numeros de Catalan

```
void catalan(int n) {
    int res = 1;
    cout << res << " ";
    // Iterate till n
    for (int i = 1; i < n; i++) {
        // Calculate the ith Catalan number
        res = (res * (4 * i - 2)) / (i + 1);
           cout << res << " ";
    }
}// O(n)
int main() {
    int n = 10;
    catalan(n);
    return 0;
}
```

The Most Important Sequence: The Catalan Numbers

Referencias

- Black_Fate. (2022). *[Educational] Combinatorics Study Notes (1)*. Recuperado de https://codeforces.com/blog/entry/110376
- Choe, J. et al. (s.f.). *Combinatorics*. Recuperado de https://usaco.guide/gold/combo?lang=cpp
- Davis, T. (2016). *Catalan Numbers*. Recuperado de http://www.geometer.org/mathcircles/catalan.pdf
- GeeksforGeeks. (2024). *Program for nth Catalan Number*. Recuperado de https://www.geeksforgeeks.org/program-nth-catalan-number/?ref=shm
- Laaksonen, A. (2018). *Competitive Programmer's Handbook*. Recuperado de https://cses.fi/book/book.pdf
- topcoder. (2018). *Basics of Combinatorics*. Recuperado de https://www.topcoder.com/thrive/articles/Basics of Combinatorics
- Vaibhav . (2024). *Basics of Combinatorics for Competitive Programming*. Recuperado de https://www.geeksforgeeks.org/basics-of-combinatorics-for-competitive-programming/