9. Divide y Vencerás (Divide and Conquer)

La técnica **Divide y Vencerás (Divide and Conquer)** es un paradigma algorítmico potente para el diseño de algoritmos eficientes. Consiste en descomponer recursivamente un problema grande en dos o más subproblemas más pequeños y del mismo tipo, hasta que estos subproblemas son lo suficientemente simples como para ser resueltos directamente. Las soluciones de estos subproblemas se combinan luego para formar la solución al problema original.

—

0.1. ¿Qué es la Técnica de Divide y Vencerás?

El paradigma Divide y Vencerás sigue un proceso de tres pasos:

1. **Dividir (Divide):** El problema original se divide en varios subproblemas más pequeños del mismo tipo que el problema original. Estos subproblemas son partes independientes y no superpuestas del problema inicial. Si el problema es lo suficientemente pequeño como para ser resuelto directamente, se considera el caso base.
2. **Vencer (Conquer):** Se resuelven los subproblemas recursivamente. Si un subproblema es lo suficientemente pequeño, se resuelve directamente (este es el caso base de la recursión).
3. **Combinar (Combine):** Las soluciones de los subproblemas se combinan para construir la solución del problema original.

Este patrón es inherentemente recursivo, ya que los pasos "Dividirτ "Vencer.ª menudo implican llamadas recursivas a la misma función con entradas más pequeñas.

—

0.2. Ejemplos Clásicos de Divide y Vencerás en C++

Muchos algoritmos eficientes se basan en este paradigma. Hemos visto **Merge Sort** y **Quick Sort** en la sección de algoritmos de ordenación, que son ejemplos excelentes de Divide y Vencerás. Aquí, exploraremos otro ejemplo clásico que ilustra bien los tres pasos.

Cálculo de Potencia (x^n)

Calcular x^n (x elevado a la potencia n) de manera eficiente.

Listing 1: Cálculo de Potencia usando Divide y Vencerás

```
1 #include <iostream>
2
3 using namespace std; // Incluyendo namespace std como se
      solicit
```

```cpp

// Funci n recursiva para calcular x^n usando Divide y Vencer s
double power(double x, int n) {
    // Caso Base: x^0 = 1
    if (n == 0) {
        return 1.0;
    }
    // Caso Base: x^1 = x
    if (n == 1) {
        return x;
    }

    // Manejar exponente negativo
    if (n < 0) {
        return 1.0 / power(x, -n);
    }

    // Dividir: Calcular la potencia para la mitad del exponente
    double half_power = power(x, n / 2);

    // Vencer y Combinar:
    // Si n es par, x^n = (x^(n/2)) * (x^(n/2))
    if (n % 2 == 0) {
        return half_power * half_power;
    }
    // Si n es impar, x^n = x * (x^(n/2)) * (x^(n/2))
    else {
        return x * half_power * half_power;
    }
}

int main() {
    double base = 2.0;
    int exponente = 10; // 2^10 = 1024
    cout << base << " elevado a la " << exponente << " es: " << power(base, exponente) << endl;

    base = 3.0;
    exponente = 5; // 3^5 = 243
    cout << base << " elevado a la " << exponente << " es: " << power(base, exponente) << endl;

    base = 2.0;
    exponente = -2; // 2^-2 = 0.25
    cout << base << " elevado a la " << exponente << " es: " << power(base, exponente) << endl;

    return 0;
```

```
49 }
```

—

0.3. Análisis de Eficiencia de Divide y Vencerás

La complejidad temporal de los algoritmos de Divide y Vencerás se analiza típicamente usando **ecuaciones de recurrencia**. El **Teorema Maestro (Master Theorem)** es una herramienta poderosa para resolver muchas de estas recurrencias y determinar la complejidad Big O.

Una ecuación de recurrencia común para Divide y Vencerás es de la forma:

$$T(N) = aT(N/b) + f(N)$$

Donde:

- N: Tamaño del problema.
- a: Número de subproblemas en los que se divide el problema.
- N/b: Tamaño de cada subproblema (se asume que todos tienen el mismo tamaño).
- $f(N)$: Costo de dividir el problema y combinar las soluciones de los subproblemas.
- **Merge Sort:**
  - Recurrencia: $T(N) = 2T(N/2) + \mathcal{O}(N)$.
  - Complejidad Temporal: $\mathcal{O}(N \log N)$.
  - Complejidad Espacial: $\mathcal{O}(N)$ (por el array temporal en la fusión).
- **Quick Sort:**
  - Recurrencia (caso promedio): $T(N) = 2T(N/2) + \mathcal{O}(N)$.
  - Complejidad Temporal: $\mathcal{O}(N \log N)$ en promedio.
  - Complejidad Espacial: $\mathcal{O}(\log N)$ en promedio (debido a la profundidad de la pila de recursión).
- **Cálculo de Potencia (x^n):**
  - Recurrencia: $T(N) = T(N/2) + \mathcal{O}(1)$.
  - Complejidad Temporal: $\mathcal{O}(\log N)$. El número de llamadas recursivas es logarítmico respecto a n.
  - Complejidad Espacial: $\mathcal{O}(\log N)$ (debido a la pila de llamadas recursivas).

El paradigma Divide y Vencerás es la base de algoritmos con complejidades muy eficientes, especialmente $\mathcal{O}(N \log N)$ y $\mathcal{O}(\log N)$.

—

0.4. Aplicaciones Comunes de Divide y Vencerás

Este paradigma es la base de muchos algoritmos fundamentales y eficientes en informática:

- **Algoritmos de Ordenación:** Merge Sort, Quick Sort.
- **Búsqueda Binaria:** Aunque a menudo se ve como una técnica de búsqueda, su naturaleza es dividir el problema a la mitad.
- **Transformada Rápida de Fourier (FFT):** Un algoritmo crucial para el procesamiento de señales.
- **Multiplicación de Matrices (Strassen's Algorithm):** Un algoritmo más rápido que el método estándar para multiplicar matrices grandes.
- **Encontrar el elemento máximo/mínimo en un array.**
- **Problemas de Geometría Computacional:** Por ejemplo, encontrar el par de puntos más cercanos.

La técnica de Divide y Vencerás es una herramienta conceptual poderosa para transformar problemas complejos en subproblemas manejables, llevando a soluciones eficientes y elegantes.