Máximo común divisor (MCD) y mínimo común múltiplo (MCM)

📜 Historia y curiosidad sobre el Máximo Común Divisor (MCD)

El concepto de **máximo común divisor (MCD)** aparece ya en la Antigüedad.
El primero en estudiarlo de manera sistemática fue **Euclides de Alejandría** (~300 a.C.), en su obra *Los Elementos*.

Allí presentó el **algoritmo de Euclides**, considerado uno de los algoritmos más antiguos que aún se usan.
Este algoritmo lo estudiaremos hoy en clases.

✨ ¿Por qué es interesante el MCD?

- Sirve para **simplificar fracciones** (por ejemplo, convertir 150/210 en 5/7).
- Se utiliza en **problemas de sincronización**: si dos eventos ocurren cada `a` y `b` segundos, el MCD ayuda a entender su ciclo común.
- Es esencial en **criptografía** y algoritmos modernos como RSA.
- ¡Y lo mejor! Un algoritmo de hace más de **2000 años** sigue resolviendo problemas actuales de informática. 🚀

MCD: máximo común divisor

🔹 Máximo Común Divisor (MCD)

El **Máximo Común Divisor (MCD)** de dos números enteros positivos `a` y `b` es el **mayor número entero que divide a ambos sin dejar residuo**.

Ejemplo:

`MCD(18, 24) = 6`

Porque los divisores de `18` son `{1, 2, 3, 6, 9, 18}` y los de `24` son `{1, 2, 3, 4, 6, 8, 12, 24}`. El mayor divisor común es `6`.

🔎 Algoritmo por fuerza bruta para el MCD

Idea del algoritmo por fuerza bruta

1. Tomamos dos números `a` y `b`.
2. El MCD nunca puede ser mayor que el menor de los dos.
3. Recorremos desde ese número hacia abajo buscando el **mayor divisor que divida a ambos**.
4. El primero que encontremos será el **MCD**.

💻 Código Python fuerza bruta

```python
def mcd_fuerza_bruta(a, b):
    menor = min(a, b)
    for d in range(menor, 0, -1):
        if a % d == 0 and b % d == 0:
            return d
```

⏱️ Complejidad del algoritmo por fuerza bruta para el MCD

- El algoritmo recorre todos los números desde `min(a, b)` hasta 1, buscando el mayor divisor común.
- Si definimos `n = min(a, b)`, entonces la complejidad del algoritmo es: `O(min(a, b)) ≡ O(n)`

> 📝 **Nota importante:**
>
> Imaginemos que queremos calcular el MCD de dos números muy grandes:
>
> ```
> a = 10^10 + 7
> b = 10^10 + 9
> ```
>
> - Con **fuerza bruta**, tendríamos que revisar todos los números desde `min(a, b) ≈ 10^10` hasta 1, ¡lo que tomaría muchísimo tiempo!

🔎 Algoritmo de Euclides para el MCD

Idea del algoritmo de Euclides

1. Tomamos dos números `a` y `b`.
2. Reemplazamos el número mayor por el **resto** de dividirlo entre el menor:
   `MCD(a, b) = MCD(b, a % b)`
3. Repetimos el proceso hasta que el resto sea 0.
4. El último número distinto de cero es el **MCD**.

💻 Código Python algoritmo de Euclides

```python
def mcd_euclides(a, b):
 while b != 0:
     a, b = b, a % b
 return a
```

⏱️ Complejidad del algoritmo de Euclides

- Cada paso reduce significativamente el tamaño de los números.
- Complejidad en el **peor caso**: O(log n), donde `n = min(a, b)`.
- Mucho más eficiente que la fuerza bruta, especialmente para números grandes.

> 📝 **Nota comparativa: Fuerza bruta vs Euclides**
>
> Supongamos que queremos calcular el MCD de dos números grandes, por ejemplo:
>
> ```
> a = 10^10 + 7
> b = 10^10 + 9
> ```
>
> - **Fuerza bruta:** hemos analizado que tendría que revisar **aproximadamente 10^10 números**, una cantidad enorme de operaciones que tardaría muchísimo tiempo.
> - **Algoritmo de Euclides:** solo necesita ≈ **log2(10^10)** ≈ **34 iteraciones**, ¡muchísimo más rápido! 🚀
>
> **Conclusión:** aunque ambos algoritmos resuelven el mismo problema, la **eficiencia** cambia drásticamente según el tamaño de los números.

MCM: mínimo común múltiplo

Definición MCM

El **mínimo común múltiplo (MCM)** de dos números enteros positivos `a` y `b` es el múltiplo más pequeño que es divisible por ambos.

`MCM(12, 18) = 36`

Ejemplo: Porque los múltiplos de `12` son `{12, 24, 36, 48, ...}` y los de `18` son `{18, 36, 54, ...}`. El primero que coincide es `36`.

Relación con el MCD

Existe una relación muy útil entre MCD y MCM:

`MCM(a, b) * MCD(a, b) = a * b`

Por lo tanto, podemos calcular el MCM fácilmente si conocemos el MCD:

`MCM(a, b) = (a * b) / MCD(a, b)`

3. Algoritmo usando Euclides

💻 Código Python para calcular el MCD

```python
def mcd_euclides(a, b):
 while b != 0:
     a, b = b, a % b
 return a

def mcm(a, b):
    return a * b // mcd_euclides(a, b)
```

📝 Problemas: Clase 3 – Máximo común divisor (MCD) y Mínimo común múltiplo (MCM)