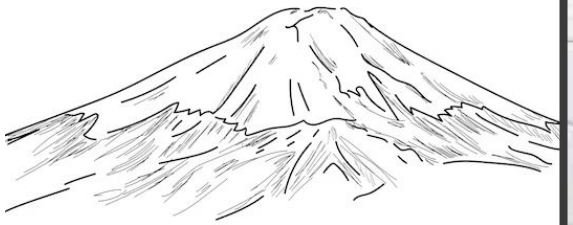
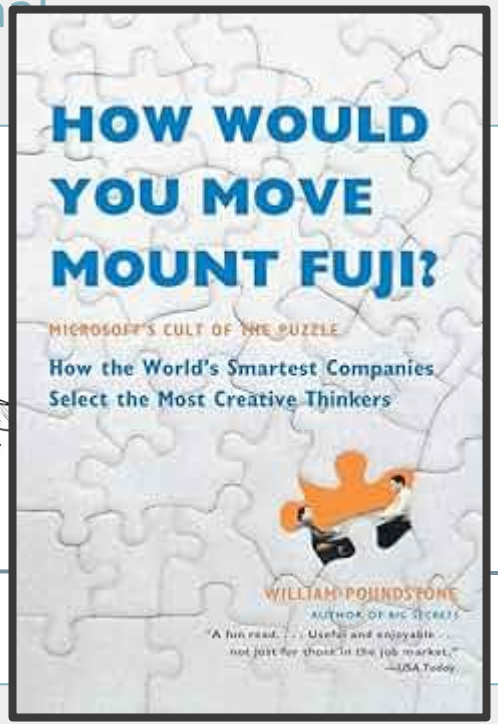
Divide y vencerás

Algoritmia y optimización
Grado en Ingeniería en Inteligencia Artificial

Resolución de problemas

Pensamiento computacional





Resolución de problemas

El reparto de bitcoins

- **Cinco** criptobros han descubierto una cartera digital con **100 bitcoins**...
  - Los bros están **jerarquizados**: 5 > 4 > 3 > 2 > 1.
  - Hacen **propuestas** en ese orden: se acepta la propuesta si hay **mayoría**; si no, **eliminan al que propone** y sigue el resto.
  - En caso de empate, el voto del proponente vale doble.
  - Todos son ***inteligentes y egoístas***: siempre van votar la mejor opción para ellos mismos.

¿Qué propuesta hará el criptobro 5?

Resolución de problemas

El reparto de bitcoins

- ¿Si fueran N criptobros y una cartera con K bitcoins?

Divide y vencerás

Divide y vencerás

Nociones

- **Estrategia** para resolver problemas complejos **dividiéndolos** en problemas más pequeños y manejables:
  - Se divide en partes más pequeñas, hasta que su resolución sea trivial.
  - Se combinan las soluciones para obtener la solución final.
- Gran cantidad de problemas, tanto computacionales como generales.

Divide y vencerás

Formalización

- Tres etapas:
  - **División**: Se divide el problema original en subproblemas más pequeños que son instancias del mismo tipo de problema.
  - **Resolución**: Cuando un subproblema es lo suficientemente simple (trivial), se resuelve de forma directa.
  - **Combinación**: Se combinan las soluciones de los subproblemas para obtener la solución del problema original.

Divide y vencerás

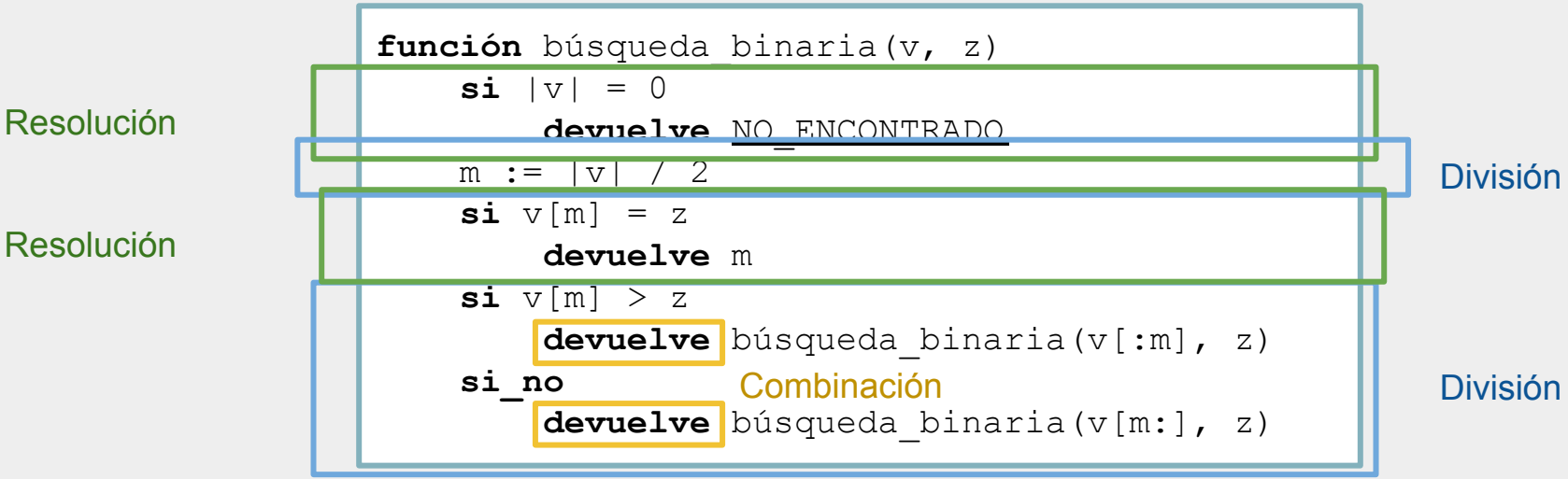
Esquema general

```
función dyv(n)
    si trivial(n)
        devuelve resolver(n)

    subproblemas = division(n)
    para i := 1 hasta |subproblemas|
        soluciones += dyv(subproblema)

    devuelve combinar(soluciones)
```

Divide y vencerás

Ejemplo: búsqueda binaria

La búsqueda binaria es un ejemplo de divide y vencerás:



Algoritmos de ordenación

Divide y vencerás

Algoritmos de ordenación

- ¿Cómo ordenar un vector siguiendo el esquema divide y vencerás?

4	3	7	8	9	2	7	9	0

Ordenación por mezcla: MergeSort

Idea principal

- **Idea principal**: es más simple obtener una lista ordenada a partir de dos sublistas ya ordenadas.
- En las etapas del esquema:
  - **División**: se divide el vector en dos partes iguales.
  - **Resolución**: cuando la lista es de tamaño 1, ya está ordenada.
  - **Combinación**: dadas dos listas ordenadas, mezclar sus elementos manteniendo el orden.

Ordenación por mezcla: MergeSort

Traza

4	3	7	8	9	2	7	9	0

Ordenación por mezcla: MergeSort

Algoritmo

```
función ordenación_por_mezcla(v):
    si |v| <= 1:
        devuelve v

    mitad := |v| / 2
    v_i := ordenación_por_mezcla(v[:mitad])
    v_d := ordenación_por_mezcla(v[mitad:])

    v := mezclar(v_i,v_d)
    devuelve v
```

Ordenación por mezcla: MergeSort

Función *mezclar*

```
función mezclar(v_i, v_d):
    i,d := 0
    resultado := []

    mientras i < |v_i| y d < |v_d|:
        si v_i[i] <= v_d[d]:
            resultado += v_i[i]
            i := i + 1
        si no:
            resultado += v_d[d]
            d := d + 1

     resultado += v_i[i:]
     resultado += v_d[d:]

    devuelve resultado
```

Ordenación por mezcla: MergeSort

Complejidad

```
función ordenación_por_mezcla(v):
    si len(v) <= 1:
        devuelve v

    mitad := |v| / 2
    v_i := ordenación_por_mezcla(v[:mitad])
    v_d := ordenación_por_mezcla(v[mitad:])

    v := mezclar(v_i,v_d)
    devuelve v
```

$$T(n) \in \mathcal{O}(n \log n)$$

Ordenación rápida: QuickSort

Idea principal

- Escoger un elemento (pivote) y colocarlo en su sitio **correcto**.
- Dividir el resto de elementos en dos conjuntos:
  - Los elementos **menores** que el pivote (izquierda)
  - Los elementos **mayores** que el pivote (derecha)
- Ordenar **recursivamente** estos dos conjuntos

Ordenación rápida: QuickSort

Algoritmo

```
función qsrt(v):
    si |v| <= 1
        devuelve v
    si no
        pivote := elegir-pivote(v)
        izquierda, derecha := partición(v,pivote)
        devuelve [qsrt(izquierda),pivote,qsrt(derecha)]
```

Ordenación rápida: QuickSort

Algoritmo: funciones auxiliares

- **elegir-pivote(v)**: ¿cómo se puede elegir un pivote? ¿cuál sería el *mejor* pivote? ¿qué coste tendrían estas elecciones?
- **partición(v, pivote):** ¿cómo se puede dividir el vector en dos partes a partir del pivote? ¿con qué coste?

Ordenación rápida: QuickSort

Traza

4	3	7	8	9	2	7	9	0

Ordenación rápida: QuickSort

Complejidad

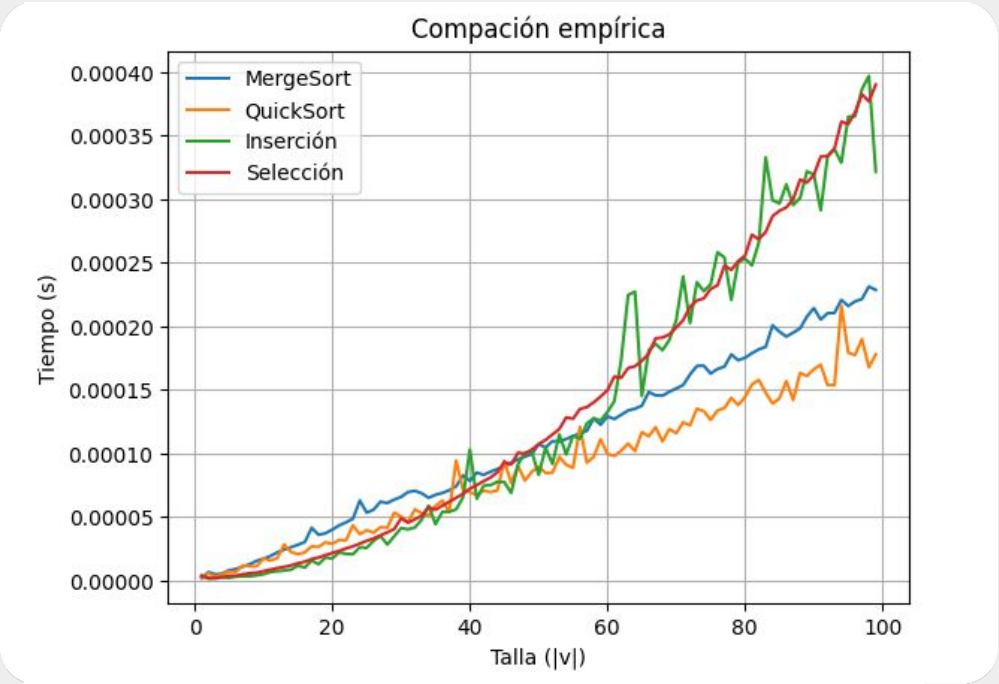
```
función qsrt(v):
    si |v| <= 1
        devuelve v
    si no
        pivote := elegir-pivote(v)
        izquierda, derecha := partición(v,pivote)
        devuelve [qsrt(izquierda),pivote,qsrt(derecha)]
```

Mejor caso: $T(n) \in \mathcal{O}(n \log n)$

Peor caso: $T(n) \in \mathcal{O}(n^2)$

Algoritmos de ordenación

Comparativa de complejidad



Consideraciones finales

Divide y vencerás

Consideraciones

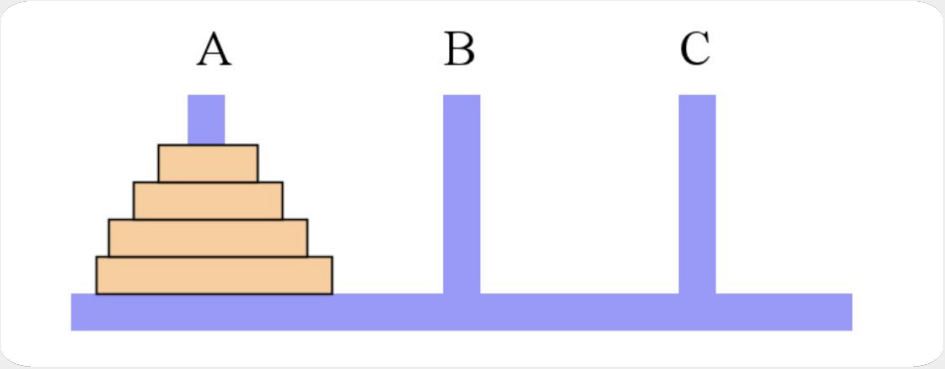
- No siempre un problema de talla menor es más fácil de resolver.
- La solución de los subproblemas no implica necesariamente que la solución del problema original se pueda obtener fácilmente
- Aplicable si encontramos:
  - Forma de descomponer un problema en subproblemas de talla menor
  - Forma directa de resolver problemas menores a un tamaño determinado
  - Forma de combinar las soluciones de los subproblemas que permita obtener la solución del problema original

Problemas

Divide y vencerás

Problema: Torres de Hanoi

- Llevar los discos de A a C, pudiendo usar B (auxiliar).



- **Reglas**: los discos solo se pueden mover uno a uno y nunca se puede poner un disco sobre uno más pequeño.

Divide y vencerás

Problema: el juego del Nim

- Hay N fichas en un tablero y cada jugador retira, alternativamente, 1 o más fichas de la mesa hasta un máximo de M.
- El juego termina cuando no quedan fichas sobre la mesa, y pierde el jugador que tiene el turno y no puede retirar ninguna ficha.
- ¿Hay una jugada ganadora para un N y M dados?

Divide y vencerás

Problema: Contar número de inversiones

- Dado un vector v, ¿cuántas inversiones contiene?
- El par (i,j) forma una inversión si v[i] > v[j] y i < j.

4	3	7	0

4 inversiones

- Búsqueda exhaustiva, ¿complejidad?
- ¿Podemos hacerlo mejor?