

ALGORITMOS DvV

GENERAL: Disminuir al maximo la complejidad

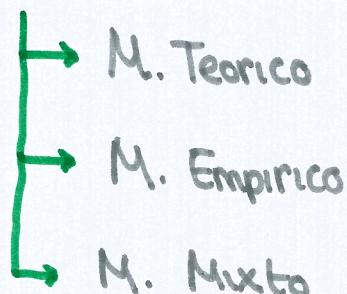
Complejidad:
$$\begin{cases} \Theta(n^k) & a < b^k \\ \Theta(n^k \log_b n) & a = b^k \\ n \log_b a & a > b^k \end{cases}$$

a: Subproblemas generados
b: Divisor del problema
k: Complejidad del algor.
de combinación

ESQUEMA:

Descomponer → Resolver → Combinar

Determinar umbrae de división de subproblemas \Rightarrow Optimiza



Pseudo código:

Función DvV (datos)

if (casoBase)

 Resuelve

else llamadas recursivas para llegar al caso base

Combinar (sol)

100% 100% 100%
100% 100% 100%

100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

100% 100% 100%

ALGORITMOS DvV

Busqueda Binaria: Encontrar un valor en un array ordenado

Complejidad: $\Theta(\log n)$

Pseudo Código:

Función BusBinaria (datos, objetivo)

Nos guardamos el primer y el último valor

$\text{if } (\text{objetivo} < \text{datos}[0]) \vee (\text{obj} > \text{datos}[\text{ult}])$

No está el valor $\Rightarrow -1$

else

while (prim <= ult)

Nos guardamos la posición del medio

$\text{if } (\text{obj} = \text{datos}[\text{med}])$

Devolvemos el medio

else if (datos[med] < obj)

Posicionamos el primero a la mitad + 1

else

Posicionamos el último a la mitad - 1

Devolvemos el último

MERGESORT: Método para ordenar array

Complejidad: $\Theta(n \log n)$

Mejor Pivote: Mediana

Pseudocódigo:

Función Ordenar (vector)

if (vector.length <= 1) Return

Nos guardamos la mitad, creamos 2 vectores izq/derecha, copiando con Copy of Range hasta la mitad

Ordenar (izq, der)

Combinar (vec, izq, der)

Función combinar (vec, izq, der)

Nos apoyamos en i, j = 0

for (vec.length)

if (i >= izq.length)

v[k] = der[j]

j++
continue

if (j >= der.length)

v[k] = izq[i]

i++
continue

if (izq[i] < der[j])

v[k] = izq[i]

i++

else

v[k] = der[j]

j++



ALGORITMOS DvV

QuickSelect: Búsqueda de K-ésimo menor número de un vector, así como mediana, mínimo, máximo

Complejidad: $\Theta(n) \xrightarrow{\text{mejor}} \Theta(n \log n)$

Pseudocódigo: Usamos `List<Int> = ArrayList(n-elem)`

Función QuickSelect (lista, valor)

Creamos dos variables, la del resultado a devolver y el pivote

Creamos 3 listas & Por Debajo, igualPivote, Por Encima , guardando en pivote el tamaño del arrayList de la mitad de la lista

for (Int num: lista)

if (num < pivot)

Añadimos num a porDebajo

else if (num > pivot)

Añadimos num a porEncima

else

Añadimos num a igualPivote

if (mitad < Debajo.size) resultado = QS(Debajo, mitad)

if (mitad < Debajo.size + igual.size) resultado =

resultado = QS(Encima, (mitad - Deb.size - Ig.size))

else

Devuelve el resultado

