

DIVIDE - ET - IMPERA

Problema Π su insieme S dei dati

$$\Pi(S) = \text{ordine } S$$

$$\Pi(S) = \max S \quad (\text{decomponibili})$$

• CASO BASE : $|S| \leq n_0 \Rightarrow T(S)$ esplicita

(es. $\int \frac{dx}{x^2} = -\frac{1}{x} + C$) $\int \equiv n_0 = 1$

• PASSO INDUTTIVO: $|S| > n_0$

$$S \rightarrow S_1 \cup S_2 \cup \dots \cup S_r$$

\cup = unione
disgiunta

DIVIDE

$\Pi(S_1), \Pi(S_2), \dots, \Pi(S_r)$ indipendentemente

IMPERA

$\Pi(S) \leftarrow \text{RICOMBINAZIONE}(\Pi(S_1), \Pi(S_2), \dots, \Pi(S_r))$ RICOMBINAZIONE

$$|S_i| < |S| \quad \forall i$$

Esempio MAX

MAXR(A, sx, dx):

$$n = dx - sx + 1$$

if $n \leq 0$:

return $-\infty$

if $n = 1$: // $sx = dx$

return $A[sx]$

// $n \geq 2$

$$cx = \frac{sx + dx}{2}$$

① $m_1 = \text{MAXR}(A, sx, cx)$

② $m_2 = \text{MAXR}(A, cx+1, dx)$

return $\max(m_1, m_2)$

chiamata iniziale $sx=0, dx=|A|-1$

$n_0 = 1$

Caso BASE

DIVIDE

PAIJO

INDUTTIVO

$r=2$

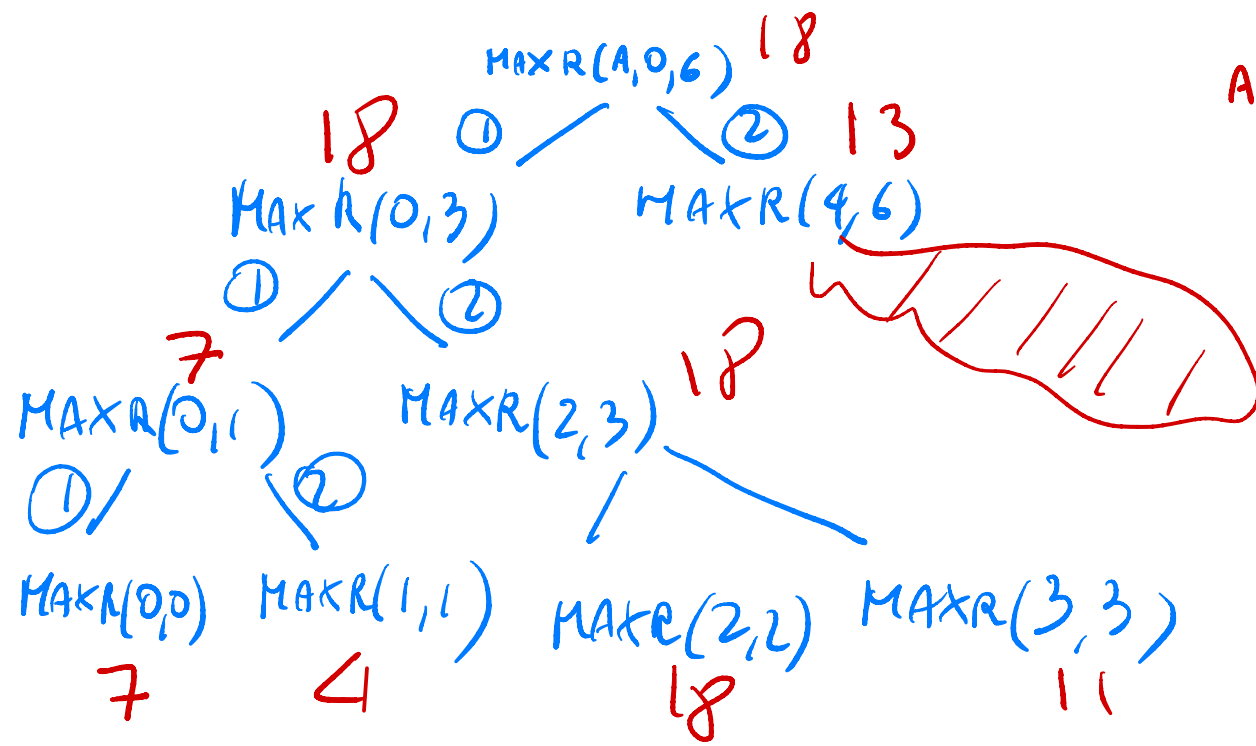
RICOMBINAZIONE

$S \rightarrow S_1 \cup S_2$

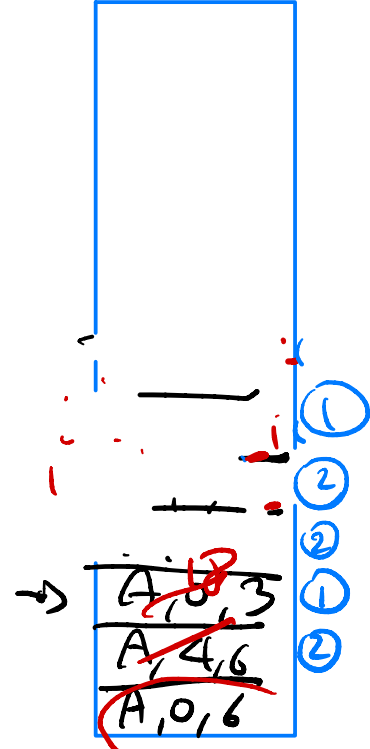
A



$A = [7, 4, 18, 11, 5, 2, 13]$
 $n = 7$



PILA



Abbiamo visto che il mergesort segue questo schema
dove le fusioni sono il passo di ricombinazione.

Equivaleentemente, la versione iterativa del mergesort è la seguente.

for ($i=1$; $i \leq \lg n$; $i++$) passo i = 2 segmenti consecutivi di
dimensione 2^{i-1} e fondili in uno di 2^i

A C E I , I N O O S T T T U Z 4

A C I N O S T T E I I N O T U Z 3

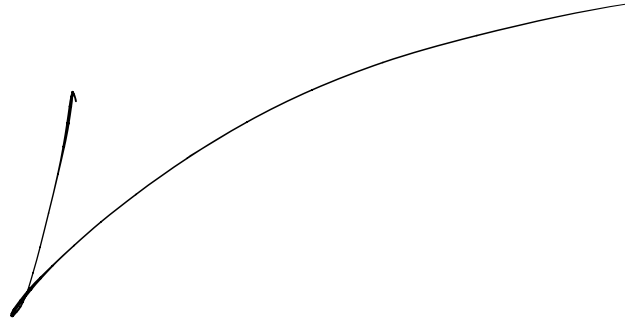
A I N T C O S T I T U Z E I N O 2

A N I T C O S T I T U Z I O E N 1

A A N T I C O S T I T U Z I O N E 0

costo totale:

$\lg_2 n$ livelli, ciascuno lineare $\Rightarrow O(n \lg n)$ tempo



PROBLEMA: Ricerca in un array ordinato

• array non ordinato \Rightarrow fooling argument $\Omega(n)$ tempo

for($i=0$; $i < n$; $i++$)

if ($A[i] == key$) return i $O(n)$ tempo

return -1

• array ordinato: SFRUTTARE TRANSITIVITA'

• LOWER BOUND - LIMITE INFERIORE : seguendo lo schema utilizzato per l'ordinamento mediante confronti

• dopo t confronti, l'algoritmo può discriminare al più 3^t situazioni

• per la ricerca l'algoritmo deve discriminare tre

$n+1$ situazioni $\left\{ \begin{array}{l} \text{la chiave non appare in } A \\ \text{la chiave appare in posizione } i=0, \dots, n-1 \end{array} \right.$

$\Rightarrow 3^t \geq n+1$ condizione necessaria alla correttezza dell'algoritmo

$\Rightarrow t = \Omega(\log n)$

Nel caso di array ordinato non è necessario leggere tutto l'input
(per esempio, il massimo/minimo richiede $O(1)$ tempo
in questo scenario)