Input: A[i..j] con $i \le j$

PMAE (A, i, j)

1. if $i = j$ return A[i] else
2. trova PMAE (A, i, ⌊(i+j)/2⌋)
3. trova PMAE (A, ⌊(i+j)/2⌋ + 1, j)
4. trova PMAE che contiene sia A[⌊(i+j)/2⌋] che A[⌊(i+j)/2⌋+1]
5. return il MASSIMO delle tre seq trovate

Detto T(N) il numero di operazioni di PMAE (A, 1, N), abbiamo:

1. richiede tempo $O(1)$, 2 e 3 richiedono tempo $T(N/2)$,
4. richiede temp $O(N)$, 5. richiede $O(1)$ $\qquad O(1)$ = costante

$$T(N) = 2T(N/2) + O(N) \Rightarrow T(N) = O(N \log N)$$

$$T(N) \le 2T(N/2) + cN \le 2^h T(N/2^h) + hcN$$

Ponendo $h = \log N$, si ha $2^h = 2^{\log N} = N$

$$T(N) \le NT(1) + (\log N)cN = O(N \log N)$$

DIVID - et - IMPERA
NUMERO OPERAZIONI

Ne è valsa la pena?

Confrontiamo gli algoritmi su di un pc che esegue un milione di operazioni al secondo

Alg	1	2	3
tempo per risolvere con dim 10^2	1 s	1/100 s	0.000 6 084 s
10^3	16,67 m	1 s	0 . 0 e . . . s
10^4	.		
10^5	.		
10^6	31688,09 y	277,78 h	19 . . . .

La differenza su moltissimi numeri è enorme, i problemi vanno