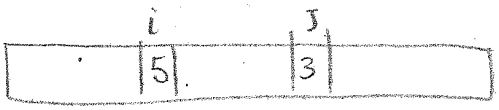
(II) Dato un array $A[1,n]$ di interi una inversione è una coppia di indici i, j $\quad i<j \quad A[i] > A[j]$



Dare un algoritmo divide et impera

$Inv(A)$

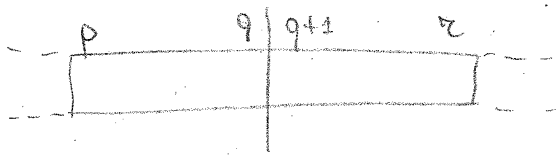
che ritorna il numero di inversioni in A

OSSERVAZIONE

una inversione in $A[p,r]$ è

$$p \leq i < j \leq r \qquad A[i] > A[j]$$

se dividiamo $A[p,r]$



possiamo avere

- $p \leq i < j \leq q \quad \rightsquigarrow$ inversione in $A[p,q]$
- $q+1 \leq i < j \leq r \quad \rightsquigarrow$ " " $A[q+1,r]$
- $p \leq i \leq q \;,\; q+1 \leq j \leq r \quad \rightarrow$ inversione "a cavallo"

quindi

$$\# Inv(A[p,r]) = \# Inv(A[p,q]) + \# Inv\, A[q+1,r] +$$

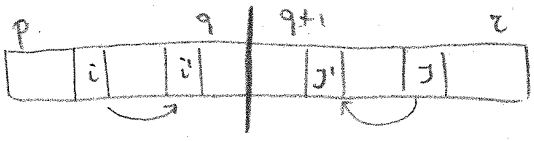
$$+ \# \left\{ (i,j) \mid \begin{array}{l} i \in [p,q] \\ j \in [q+1,r] \end{array} \quad A[i] > A[j] \right\}$$

Questo porta all'algoritmo

```
Inv (A, p, r)                    ~ conta le inversioni e ordina!
  if p < r
    q = ⌊(p+r)/2⌋

    return Inv (A, p, q) +
           Inv (A, q+1, r) +
           InvMerge (A, p, q, r)   ~ calcola (*)
```

Come realizzare InvMerge? 3 osservazioni



a) dopo essere stati ordinati $A[p,q]$ e $A[q+1,r]$ contengono gli stessi elementi di prima (ordinamento in loco)

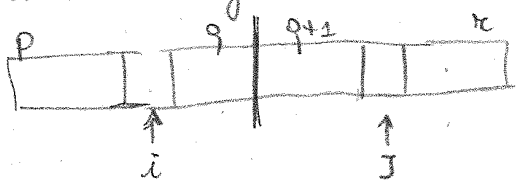
$\Downarrow$

(b) se (i,j) con $i \in [p,q]$, $j \in [q+1,r]$ era inversione prima dell'ordin.

detti i', j' le nuove posizioni a cui sono stati spostati gli elementi (i',j') è ancora inversione (e viceversa!)

$\rightsquigarrow$ la quantità $(*)$ è invariata!

(c) nei sottoarray ordinati



se (i,j) è inversione (ovvero $A[i] > A[j]$)

allora (i',j) è inversione per ogni $i \le i' \le q$

$\rightsquigarrow$ $q-i+1$ inversioni

```
InvMerge (A, p, q, r)
    n1 = q - p + 1
    n2 = r - q
    L[1, n1] ← A[p, q]
    R[1, n2] ← A[q+1, r]
    L[n1+1] = R[n2+1] = +∞

    i = j = 1
    inv = 0
    for k = p to r
        if L[i] ≤ R[j]
            A[k] = L[i]
            i++
        else
            A[k] = R[j]
            j++
            inv = inv + n1 - i + 1
    return inv
```