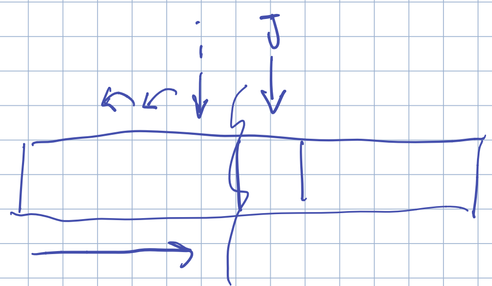
```
InsertionSort (A) {
    for (j ← 2 to A.length) {
        i ← j-1
        key ← A[j]
        while (i > 0 && A[i] > key) {
            A[i+1] ← A[i]
            i ← i-1
        }
        A[i+1] ← key
    }
}
```



Correttezza

InsertionSort (A) termina sempre con A ordinato

Invariante del for

All' inizio della j-esima iterazione del ciclo for $A[1..j-1]$ è ordinato

Dim

Per induzione su J

BASE

$J=2$

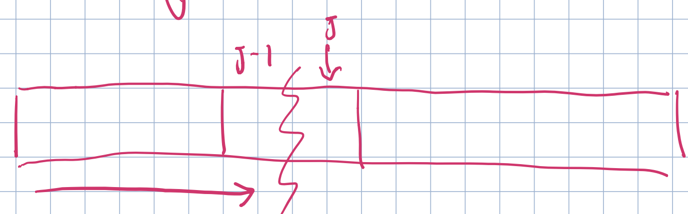
Ts) $A[1..1]$ è ordinato

Vero un vettore con un solo elemento è sempre ordinato ✓

PASSO induttivo

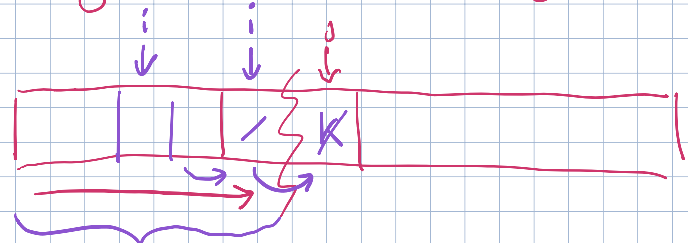
HpInd) All'inizio dell'iterazione J
$A[1..J-1]$ è ordinato

Ts) All'inizio dell'iterazione $J+1$
$A[1..J]$ è ordinato



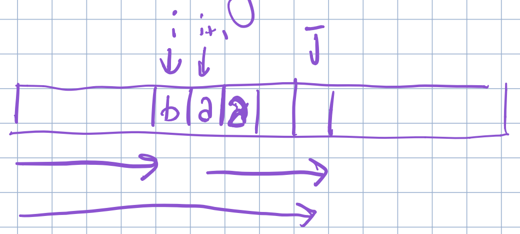
Hp Ind

Viene eseguita l'iterazione J-esima



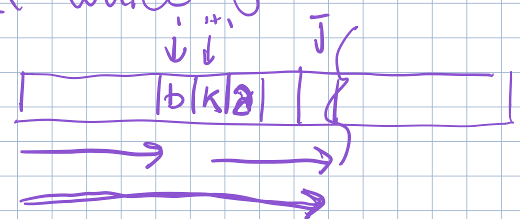
key = k

Al termine del ciclo while :

- i è fermo sul primo elemento minore o uguale a k
- 

$b \le k < a$

Fuori del while



A[1..j] ordinato

$\perp$

fine Invariante for

Correttezza

InsertionSort(A) termina sempre con A ordinato

Dim

Termina perché

- nell' while i viene decrementato e una delle guardie è "$i > 0$"
- nel for j non viene modificato

L' ultima "macro istruzione" eseguita è il for con $j = A.length + 1$

Dall' invariante

$A[1..A.length]$ è ordinato

Complessità

Risorse < Spazio / Tempo ⇐

Spazio
- In Place ⇝ Spazio costante oltre all'input
- Non In Place ⇝ Spazio allocato cresce al crescere dell'input

InsertionSort In-Place

Tempo

$$T : \mathbb{N} \rightarrow \mathbb{R}^+$$

$\mathbb{N}$: dimensione input (es. lunghezza vettore)

T_o caso ottimo

T_m caso medio

T_p caso pessimo

<u>Criterio di costo</u> Uniforme

Ogni operazione di base ha costo costante $\rightsquigarrow c$

assegnamenti, test

dipende dal PC

InsertionSort (A) {

- for ($j \leftarrow 2$ to A.length) { $2c \cdot n$
  - $i \leftarrow j-1$ $2c(n-1)$
  - key $\leftarrow A[j]$ $c(n-1)$
  - while ($i > 0$ && $A[i] >$ key) $2c\sum_{j=2}^{n} t_j$
    - $A[i+1] \leftarrow A[i]$ $2c\sum_{j=2}^{n}(t_j-1)$
    - $i \leftarrow i-1$ $2c\sum_{j=2}^{n}(t_j-1)$
  - }
  - $A[i+1] \leftarrow$ key $2c(n-1)$
- }

}

n = A.length

$j = 2 \ldots n+1$ $((n+1)-2)+1 = n$

$[a..b] \qquad b-a+1$

t_j = n.ro di volte che eseguo "while(..)" quando sono alla j-esima iterazione del for

$$T(n) = 2c\cdot n + 5c(n-1) + 2c\sum_{j=2}^{n} t_j +$$
$$4c\sum_{j=2}^{n}(t_j - 1)$$

$t_j \in [1..j]$

$i \in [0..j-1] \rightsquigarrow (j-1-0)+1 = j$

<u>Caso ottimo</u> $\quad t_j = 1 \quad$ vettore ordinato

$$T_o(n) = 2cn + 5c(n-1) + 2c\sum_{j=2}^{n} 1$$
$$= 2cn + 5c(n-1) + 2c(n-1)$$
$$= \Theta(n)$$

Caso Pessimo $t_j = j$ vettore ordinato al contrario

$$T_P(n) = 2cn + 5c(n-1) + 2c\sum_{j=2}^{n} j + 4c\sum_{\cancel{j=2}\ h=1}^{\cancel{n}\ n-1} (\cancel{j-1}\ h)$$

$\rightarrow$ Somma di Gauss

$$= 2cn + 5c(n-1) + 2c\left(\frac{n(n+1)}{2} - 1\right) + 4c\,\frac{n(n-1)}{2}$$

$$= \Theta(n^2)$$

Caso Medio $t_j = j/2 \rightsquigarrow T_M(n) = \Theta(n^2)$