Regole pseudocodice

- Java / C / pascal
- for / while / do While
  - for: sappiamo quante iterazioni dobbiamo fare
  - while / do While: se può essere interrotto
- if then else
- indentazione
- Begin / end { }
- //, Δ oppure /* ... */
- Assegnamenti : ←, =, :=
- condizione ==
- Valore delle variabili è locale
- gli arrays sono globali, contengono valori da un solo tipo vanno da 1 a n
- Dati : oggetti con con degli attributi
- scriveremo funzioni / procedure
  - ricevono parametri e forniscono 1 output
  - non restituisce nulla
- passaggio parametri per valore
- Assumiamo di avere memoria infinita ad accesso diretto
- 1 sola CPU

Algoritmi di ordinamento

Array → Sort

- BubbleSort *

ci mettono entrambi circa n^2 ma il secondo usa un'altro array, quindi più spazio

X Array appoggio → più piccolo in I pos. II in II *

selectionSort

- I → I , II → II no appoggio

quasi uguali

X Max in fondo, secondo max in penultimo

lentissimo

X quando trovo un numero "piccolo", shift per fare posto n^3

- InsertionSort

se è ordinato al contrario è lentissimo

al peggio n^2, ma se va bene è velocissimo

```
void SelectionSort ( V[] ){
    n = lenght(V)   solo per comodità
C(n-1)→ for( j = 1 to n-1 ){
    C(n-1)→ Pmin = j
n(n+1)/2 → for (i = j+1 to n ){        n + (n-1) + (n-2) + ··· + 3 + 2 + 1
                                       Σ_{j=1}^{n} j = (n+1)·n / 2
        C·n(n+1)/2 → if( V[i] < V[Pmin] ){
                Pmin = i ← C·Tif
            }
        }
        App = V[j]  ← C·(n-1)
        V[j] = V[Pmin] ← C(n-1)
        V[Pmin] = APP   C(n-1)
    }
}
```

$$T(n) = 5C(n-1) + \cancel{2}\,c\,\frac{n(n+1)}{\cancel{2}} + cT_{if}$$

il tempo di exec. dei for non varia in base all'ordine degli elementi.
L'unica variabile è l'if

Migliore $t_{if} = 0 \rightarrow$ V è ordinato

$$t_{migliore}(n) = 5C(n-1) + cn(n+1) \approx n^2$$

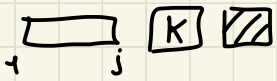
Peggiore $t_{if} = \left(\sum_{j=1}^{n} j\right) \cdot \frac{\cancel{2}}{\cancel{2}} = \frac{n(n+1)}{2}$

$$T_{pegg}(n) \approx n^2$$

Medio $\approx n^2$

```
void InsertionSort(V[])
C·(n-1)  for(i=2 to length(V)){
  C(n-1)   K = V[i]
  C(n-1)   j = i-1
3C Σ_{i=2}^{n} tw_i   while((K < V[j] AND (j>0))
                        V[j+1] = V[j]
                        j--
                      }
  C(n-1)   V[j+1] = K
}
```

Nella prima porzione gli elementi sono già ordinati



$$T_{insSort}(n) = 4C(n-1) + 3C\sum_{i=2}^{n} T_{wi}$$

caso migliore $tw_i = 0 \quad \forall i = 2 \dots n$

↳ V è ordinato

$t_m\ insSort = 4C(n-1) \approx n$

caso peggiore $tw_i = i-1$ controllo tutti i numeri prima

↳ V è ordinato al contrario (e non ci sono elementi uguali)

$$T_P\ insSort = 4C(n-1) + 3C\sum_{i=2}^{n} i-1$$

↳ $1+2+3+\dots n-1$

$$= 4C(n-1) + 3C\sum_{i=1}^{n-1} i$$

$$\sum_{i=1}^{n-1} i = \left(\sum_{i=1}^{n} i\right) - n = \frac{n(n+1)}{2} - n = \frac{(n-1)n}{2}$$

$$4C(n-1) + 3C\frac{n(n-1)}{2} \approx \frac{3n^2}{2}$$

Medio $tw_i = \frac{i}{2}$ $T_{medio} \approx \frac{3}{4}n^2$