Big O → analisi caso peggiore (questo alg non avrà complessità peggiore di ...)

Θ → PRECISO

Ω → analisi caso migliore (= migliore ...)

Analisi di algoritmi di ordinamento con complessità quadratica
(nel caso peggiore)

★SELECTION SORT

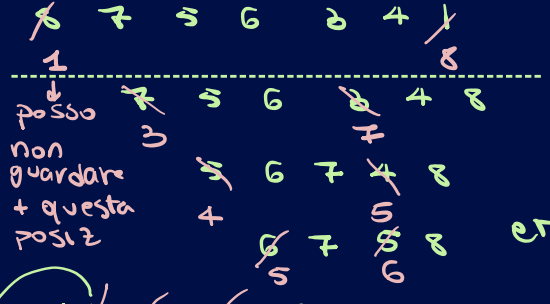
```
                                SELECTION SORT
*****************************************************
void selectionSort(vector<int>& v)
{
    int current_min_index;
    unsigned int size = v.size();
    for (unsigned int i=0; i<size; ++i)
    {
      current_min_index = i;
       for (unsigned int j=i+1;j<size; ++j)
          if (v[current_min_index] > v[j])
              current_min_index = j;
       scambia(v, i, current_min_index);
    }
}
```

min nell'esempio → indice della cella in cui voglio mettere a posto min della seq

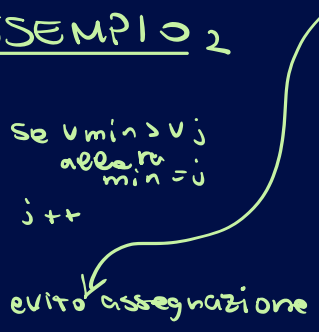
per scandire sequenza a destra della cella i

Caso migliore = Caso peggiore = Θn^2

ESEMPIO



ESEMPIO 2



quanto mi beneficio dell fatto che sia già ordinato?

ho confrontato 1 con 2,3,4 ($n-1$) operazioni

ho confrontato 2 con 3,4 ($n-2$) operazioni

etc...

(n = n° celle)

QUINDI $(n-1)+(n-2)\dots$ non è altro che la sommatoria

$$\sum_{k=1}^{n} k \in \Theta n^2$$

★ INSERTION SORT (algoritmo adattivo)

```
                                    INSERTION SORT
*****************************************************
void insertionSort(vector<int>& v)
{
   int current, prev;
   unsigned int size = v.size();
   for (unsigned int i=1; i<size; ++i)
   { current=i;  → curr nell'esempio
     prev=i-1;
     while(prev>=0 && v[current]<v[prev])
        {
         scambia(v, current, prev);
         --current;
         --prev;
        }
   }
}
```

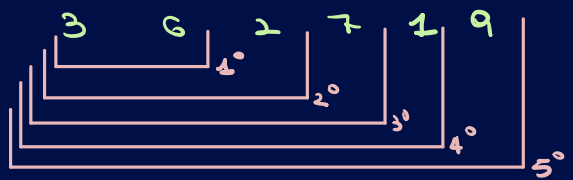
Voglio crescente ↑ ho sequenza: 9,8,7,6

Caso peggiore → ordine opposto a quello che sto cercando (tanti scambi) $= \sum_{k=1}^{n} i$ scambi cioè $\frac{n(n+1)}{2}$ scambi $\rightarrow \Theta n^2$

CASO Migliore: n confronti ma 0 swap xk già ordinato $= \Theta n$

→ fa sì che nel momento in cui due num sono già ae loro posto non avviene lo swap e neppure analisi di quelli che precedono

ESEMPIO



Considera prima 2 elem; poi 3, poi 4, etc

1°	3 < 6 ?	Sì → non fa nulla	2 elem, 0 scambi	3 6 2 7 1 9
2°	6 < 2 ?	NO → swap	3 elem, 2 scambi	2 3 6 7 1 9
	3 < 2 ?	NO → swap		
3°	6 < 7 ?	Sì	4 elem, 0 scambi	2 3 6 7 1 9
4°	7 < 1 ?	NO → swap	5 elem, 4 scambi	1 2 3 6 7 9
	6 < 1 ?	NO → swap		
	3 < 1 ?	NO → swap		
	2 < 1 ?	NO → swap		
5°	7 < 9 ?	Sì	6 elem, 0 scambi	1 2 3 6 7 9
				OK

PSEUDO-COD

prev = i-1 (i=1) curr = i

3 6 2 7 1 9

prev i curr

while prev ≥ 0 ? sì 6 < 3 ? no

ESCO DAL WHILE i++

3 6 2 7 1 9

prev i curr

while prev ≥ 0 ? sì 2 < 6 ? SÌ → entro nel while e swap

etc ...

*1 Così come in tutti i casi di sommatoria, cioè casi in cui faccio azioni ripetute per una costante aumentata o diminuita di un tot, esempio: $n+(n+1)+(n+2)+\dots\}=\sum$ $n+(n-1)+(n-2)+\dots$

abbiamo (a lezione) dimostrato che $\sum = \frac{n(n+1)}{2}$ quindi xk arriva a Θ^2

$\frac{n(n+1)}{2} = n(n+1) \cdot \frac{1}{2}$ → è una costante moltiplicativa e non la considero

$= n(n+1) = n^2+n$ non considero n xk è lo stesso termine di n^2 ma di ordine inferiore → elevato ad un numero + piccolo

REGOLA

REGOLA

★BUBBLESORT

```
                                BUBBLESORT
*************************************************************
void bubbleSort(vector<int>& v)
{
   unsigned int size = v.size();
   bool scambiati;
   for (unsigned int i=1; i<size; ++i)
      {
      scambiati = false;
      for (unsigned int j=0; j<size-i; ++j)
          if(v[j]>v[j+1])
            {
              scambia(v, j, j+1);
              scambiati = true;
            }
      if (!scambiati) return;
      }
}
```

Caso peggiore → $\Theta(n^2)$

Caso migliore → $\Theta(n)$ poiché se "scambiati" rimane falso dopo il primo giro (esempio in cui è già ordinato) non si rientra nel ciclo

(fatto da me NO PROOF)

size 4

2 3 5 8 (caso migliore) tutto ordinato

i = 1, j = 0 — i, i+1

i = 2, j = 1 — 2 > 3? NO

3 > 5? NO

ETC

no scambi, n confronti? $\Theta(n)$

BOH