Algoritmo *InsertionSort* ricorsivo

Per ordinare un vettore di n elementi

- ordiniamo il vettore dei primi $n-1$ elementi
- inseriamo l'ultimo elemento nella posizione giusta

Base della ricorsione: ordinare un vettore di $n = 1$ elemento

Passo ricorsivo:

- isolare l'ultimo elemento a_n del vettore
- ordinare un sottovettore $A[1,n-1]$ di $n-1$ elementi
- inserire l'elemento a_n nel sottovettore ordinato $A[1,n-1]$

Algoritmo *InsertionSort* ricorsivo

```
InsertionSort (A,s,d)
    If  (s < d) then
        {esegui InsertionSort sui primi n-1 elementi}
        InsertionSort (A,s,d-1);
        {aggiungi l'ultimo elemento nella posizione corretta}
        AddSorted (A[d],A,s,d-1);
```

Conversione iterativa automatica

La chiamata ricorsiva non è terminale!

Per eliminare la ricorsione, occorre una pila esplicita

```
InsertionSort (n,A[1,...,n])
    stack = s_create()
    For j = d downto s+1 do                  {poni via via i dati sulla pila}
        stack = s_push(stack,A[j])
    For j = s+1 to d do
        x = s_top(stack)                     {estrai via via i dati dalla pila}
        AddSorted(x,A,s,j-1)                 {operazione terminale}
        stack = s_pop(stack)
    s_destroy(stack)
```

Conversione iterativa automatica

Ma si può usare la parte finale del vettore come una pila

```
InsertionSort (n,A[1,...,n])
    stack = s_create()
    For j = d downto s+1 do                          {poni via via i dati sulla pila}
            stack = s_push(stack,A[j])
    For j = s+1 to d do
            x = s_top(stack) A[j];        {estrai via via i dati dalla pila};
            AddSorted(x,A,s,j-1)                     {operazione terminale}
            stack = s_pop(stack)
    s_destroy(stack)
```

Algoritmo *SelectionSort* ricorsivo

Per ordinare un vettore di *n* elementi

- estraiamo fisicamente l'elemento massimo
- ordiniamo il vettore residuo
- accodiamo l'elemento estratto al vettore ordinato

Base della ricorsione: ordinare un vettore di *n = 1* elemento

Passo ricorsivo:

- estrarre fisicamente l'elemento massimo da un vettore non ordinato
- scorrere tutti gli elementi, determinando il massimo
- estrarre l'elemento massimo e ricompattare il vettore

Algoritmo *SelectionSort* ricorsivo

```
SelectionSort (A,s,d)
    If  (s < d) then
        x = ExtractMax(A,s,d)
        SelectionSort (A,s,d-1)
        Append(x,A,d)
```

Conversione iterativa automatica

La chiamata ricorsiva è terminale

Ma applichiamo comunque la soluzione con pila ausiliaria

```
SelectSort (A,s,d)
    stack = s_create()
    For j = d downto s+1 do                    {poni via via i dati sulla pila}
            x = ExtractMax(A,s,j)
            stack = s_push(stack,x)
    For j = s+1 to d do
            x = s_top(stack)                   {estrai via via i dati dalla pila}
            Append(x,A,s,j-1)                  {operazione terminale}
            stack = s_pop(stack)
    s_destroy(stack)
```

Implementazione efficiente

Anche qui si può usare la parte finale del vettore come una pila

```
SelectionSort (A,s,d)
    stack = s_create()
    For j = d downto s+1 do                    {poni via via i dati sulla pila}
          x = ExtractMax(A,s,j)
          stack = s_push(stack,x)
    For j = s+1 to d do
          x = s_top(stack)                     {estrai via via i dati dalla pila}
          Append(x,A,s,j-1)                    {operazione terminale}
          stack = s_pop(stack)
    s_destroy(stack)
```

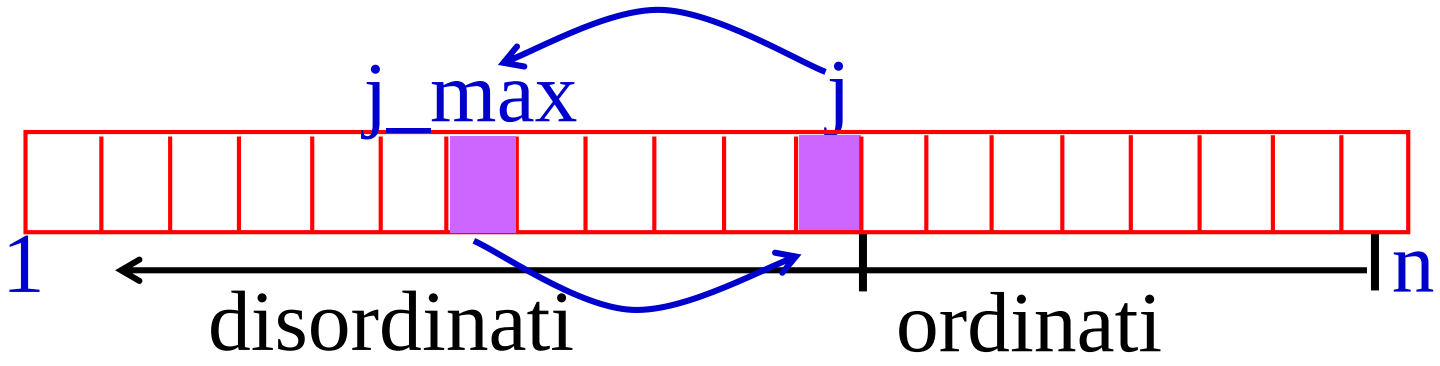
```
SelectionSort(n,A)
For j = d downto s+1 do                        {estrai il massimo e mettilo in coda}
      j_ max = FindMax(A,s,j)
      Scambia(A[j_max],A[j]);
```

SelectionSort

Scandisce la sequenza dall'ultimo elemento al primo

Ad ogni iterazione (*ExtractMax*)

- cerca l'elemento massimo A[j_max] nella sottosequenza corrente A[1,…,j]
- scambia l'ultimo elemento con quello massimo (facendo uscire l'elemento massimo dalla sequenza e ricompattandola)



Complessità $O(n^2)$ in ogni caso (anche in quello migliore)