Algoritmo per trovare il minimo fra *n* numeri

- leggere i numeri $a_1, a_2, a_3, \ldots, a_n$;
- sia *min* una variabile;
- porre $min = a_1$;
- for $i = 2$ to n step 1
    - if $a_i < min$ then $min = a_i$;
- la variabile *min* contiene il minimo.

Questo semplice algoritmo compie $n - 1$ confronti. Pertanto ha una complessità lineare.

L'idea è quella di memorizzare in *min* il primo numero e poi confrontare *min* con tutti i numeri seguenti. Se si trova un elemento minore allora esso sarà il nuovo valore da memorizzare in *min.*

Algoritmo di ordinamento *Selection Sort*

Questo metodo di ordinamento agisce su un vettore di dati e lavora sul posto (ossia non ha bisogno di memoria ausiliaria o altre strutture per memorizzare i dati). Si cerca il minimo fra gli elementi e lo si scambia con il primo elemento del vettore. Dopodiché si cerca il minimo fra gli elementi dal secondo in poi e questo secondo minimo si scambia col secondo elemento. Dopodiché si cerca il minimo fra gli elementi dal terzo in poi e questo terzo minimo lo si scambia con il terzo elemento del vettore. Si continua così fino a che il vettore non risulta ordinato.
es:
vettore dato: 5 3 6 1 2 8

Si cerca il minimo, che è 1, e si scambia con il primo elemento. Si ottiene:
1 3 6 5 2 8.

Si cerca il minimo a partire dal secondo elemento. Esso è 2 e si scambia con il secondo elemento. Si ottiene:
1 2 6 5 3 8.

Si cerca il minimo a partire dal terzo elemento. Esso è 3 e si scambia con il terzo elemento. Si ottiene:
1 2 3 5 6 8.

Si cerca il minimo a partire dal quarto elemento. Esso è 5 e si scambia con il quarto elemento. Notare che praticamente non si effettua nessuno scambio poiché il minimo coincide proprio con il quarto elemento. Si ottiene:
1 2 3 5 6 8.

Da questo punto in poi nell'esempio considerato non si effettuano più scambi anche se la procedura deve essere continuata fino alla fine.

Al primo passaggio si deve cercare il minimo fra n numeri, pertanto si compiono $n - 1$ confronti (vedi algoritmo precedente). Al secondo passaggio deve essere cercato il minimo fra $n - 1$ elementi, pertanto si eseguono $n - 2$ confronti. Continuando a contare i confronti necessari ad ogni passaggio per trovare il minimo e sommandoli tutti insieme si ottiene:

$1 + 2 + 3 + \ldots + (n-2) + (n-1) = n(n-1)/2$
che costituisce la complessità del Selection Sort nel caso peggiore.

Algoritmo di ordinamento *Insertion Sort*

Anche tale metodo di ordinamento agisce su un vettore e non ha bisogno di memoria aggiuntiva. L'idea è la seguente: supponiamo di avere un certo numero di elementi già ordinati in senso crescente. Quando dobbiamo inserire un nuovo elemento *x*, si procede confrontandolo con quello più a destra di quelli già ordinati. Se *x* è più grande allora *x* rimane dov'è, altrimenti si scambia. Poi *x* si confronta con l'elemento ancora precedente e si scambia con esso se *x* è minore. Si continua così finché *x* non si confronta con un elemento più piccolo. A questo punto *x* è arrivato nella giusta posizione, poiché sarà più grande di tutti gli elementi alla sua sinistra.

es:

vettore dato: 20 35 18 8 14 41 3 39

20	35	18	8	14	41	3	39
20	35	18	8	14	41	3	39
18	20	35	8	14	41	3	39
8	18	20	35	14	41	3	39
8	14	18	20	35	41	3	39
8	14	18	20	35	41	3	39
3	8	14	18	20	35	41	39
3	8	14	18	20	35	39	41

Al primo passo il 20 viene lasciato lì dov'è.
Al secondo passo il 35 si confronta con il 20 ed essendo più grande rimane dov'è.
Al terzo passo il 18 si confronta col 35. Essendo più piccolo avanza di un posto verso sinistra scambiandosi col 35. A questo punto si confronta con il 20 ed essendo più piccolo anche del 20 avanza di un'altra posizione verso sinistra scambiandosi col 20.

In grigio sono evidenziati i numeri che è necessario spostare verso destra per far risalire verso sinistra gli elementi che sono più piccoli di loro.

Per la complessità si osserva che al primo passo non si devono fare confronti né spostamenti (il 20 non va comparato con nessuno). Al secondo passo si fa un confronto (si compara il 35 col 20) e un scambio. Al terzo passo faremo al più due confronti (il 18 si compara col 35 e poi col 20) e due scambi. In generale all'*i*-esimo passo faremo al massimo $i - 1$ confronti e scambi per fare risalire verso sinistra l'elemento considerato. Sommando tutti i confronti che nel caso peggiore dovremmo fare si ottiene:

$$1 + 2 + 3 + \ldots + (n-2) + (n-1) = n(n-1)/2$$

che costituisce la complessità dell'Insertion Sort nel caso peggiore.