# INSERTION SORT

**INPUT:** A sequence of 'n' numbers : $[a_1, a_2 \ldots, a_n]$

**OUTPUT:** A permutation (re-ordering) $[a_1', a_2' \ldots, a_n']$ of the input sequence such that $a_1' \leq a_2' \leq a_3' \leq \cdots \leq a_n'$

KEYS: Numbers that we wish to sort

→ Efficient algorithm for sorting a small number of elements
→ The algorithm sorts the input numbers IN PLACE - it rearranges the numbers within the array
→ STEPS: ① Divide Array into sorted & unsorted elements
② Select the first unsorted element
③ Swap other elements to the right to create the correct position & shift the unsorted element
④ Advance marker to the right by 1 element

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 10 | 9 | 20 | 3 | 6 |

## PSEUDOCODE

```
FOR J = 2 to A.length
    KEY = A[J]
    // To insert A[J] into sorted sequence A[1...j-1]
    I = J-1
    WHILE I>0 AND A[I] > KEY
        A[I+1] = A[I]
        I = I-1
    A[I+1] = KEY
```

## ANALYSIS

→ Time taken depends on input
→ Also depends on 'how sorted' the list already is

**BEST CASE**: $O(n)$ → already sorted array
→ Only outer loop is running n times

**AVERAGE CASE**: $O(n^2)$ → Jumbled order

**WORST CASE**: $O(n^2)$ → Ascending to descending.
→ Every individual element compared to rest of the elements: n-1 comparisons for every nth element

**SPACE**: $O(1)$

**USAGE** ① Only a few elements
② Only a few elements to sort

## ADVANTAGES

① Simple & efficient on smaller scale
② Stable & in-place