Problem: Input ⟶ Output

Algorithm: → (I) correctness
→ (II) efficient → time efficient
→ space efficient

Sorting Problem: Input: $a_1, a_2, \dots, a_n$ (array of size n)

Output: reordering (permutation)

$a_{i_1}, a_{i_2}, \dots, a_{i_n}$

$a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_n}$

Example Input: 7, 10, 1, -2, 4

Output: -2, 1, 4, 7, 10

Insertion sort

Merge sort

Quick sort

linear sorting (radix sort, bucket sort, counting sort)

Insertion sort

Pseudo-code

```
Insertion_Sort ( A[1 ... n] )

    for   i=1   to   n

(1)         aux = A[i]
(2)         j = i-1
(3)         while ( j>0 && A[j] > aux )

                A[j+1] = A[j]
                j = j-1

(4)         A[j+1] = aux
```

constant time

time to run: t_i

Example

Input: 7 9 6 1 8

i=1 : [7] 9 6 1 8 → A[1] sorted

i=2 : [7 9] 6 1 8 → A[1 2] sorted

i=3 : 7 6 9 1 8

6 7 9 1 8 → A[1 2 3] sorted

i=4 : 1 6 7 9 8 → A[1 ... 4] sorted

i=5 | 6 7 8 9 → A[1 ··· 5) sorted

	(1)	(2)	(3)	(4)
SC : constant	constant	constant	X	X
in-place				
TC	constant c	constant c	$t_1+t_2+\cdots+t_n$	constant c

t_i := time needed to run line (3)

for-loop running time:

i=1: $\underbrace{(1)+(2)+(4)}_{3c} + \underbrace{(3)}_{t_1} = 3c+t_1$

i=2: $3c+t_2$

⋮ ⋮

i=n: $3c+t_n$

total TC = $\underbrace{3c+\cdots+3c}_{n \text{ times}} + (t_1+\cdots+t_n) = (3c)n + \sum_{i=1}^{n} t_i$