# Algorithms and Data Structures

## 2nd Lecture: Growth of function / Sort

Yutaka Watanobe, Jie Huang, Yan Pei, Wenxi Chen,
S. Semba, Deepika Saxena, Yinghu Zhou, Akila Siriweera

University of Aizu

Last Updated: 2024/12/05

# Outline

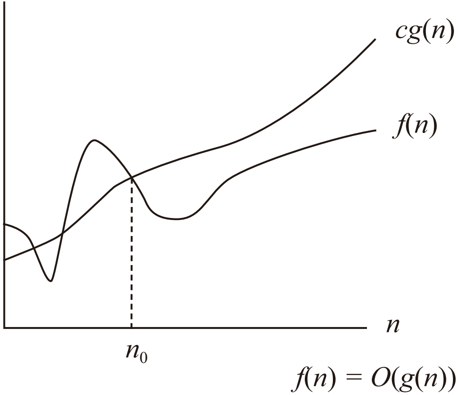
- Growth of Functions
- Bubble Sort
- Selection Sort
- Stability

# Asymptotic Notations

- When we look at input size large enough to make only the order of growth of the running time relevant, we are studying the asymptotic efficiency of algorithms.
- Usually, an algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.

# Big-Oh Notations

- O-notation (big-oh): asymptotic upper bound $O(g(n)) = \{f(n)$ : there exists positive constants $c$ and $n_0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.



# Notational Conventions

- Conventionally, we write $f(n) = O(g(n))$ to indicate that $f(n)$ is a member of the set $O(g(n))$, instead of writing $f(n) \in O(g(n))$.
- Moreover, we use asymptotic notations within mathematical formulas. For example, we write:

$$2n^2 + 3n + 1 = 2n^2 + O(n) = O(n^2)$$

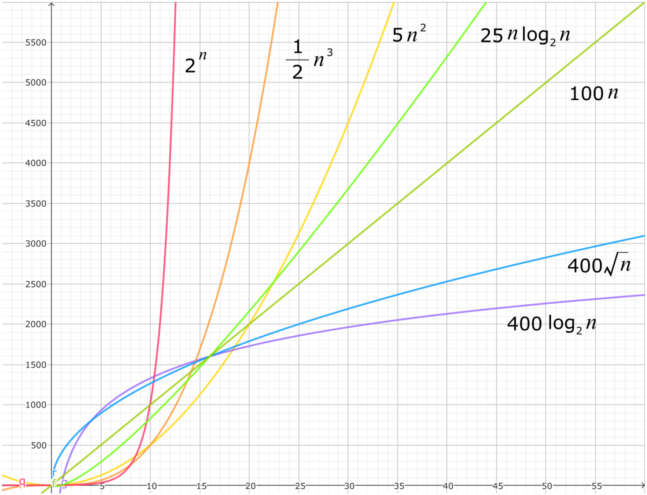
# Simple Examples

- $3n^2 + 2n + 5 = O(n^2)$
- $1000n + 5 = O(n)$
- $(3/2)^n = O(2^n)$
- $\log_2 n^2 = O(\log n)$
- $8n^5 + 2n^2 + 5 = O(n^5)$
- ..

# Comparison of Computational Complexity

| $n$ | $\log n$ | $\sqrt{n}$ | $n \log n$ | $n^2$ | $2^n$ | $n!$ |
|---|---|---|---|---|---|---|
| 5 | 2 | 2 | 10 | 25 | 32 | 120 |
| 10 | 3 | 3 | 30 | 100 | 1024 | 3628800 |
| 20 | 4 | 4 | 80 | 400 | 1048576 | $2.4 \times 10^{18}$ |
| 50 | 5 | 7 | 250 | 2500 | $10^{15}$ | $3.0 \times 10^{64}$ |
| 100 | 6 | 10 | 600 | 10000 | $10^{30}$ | $9.3 \times 10^{157}$ |
| 1000 | 9 | 31 | 9000 | 1000000 | $10^{300}$ | $4.0 \times 10^{2567}$ |
| 10000 | 13 | 100 | 130000 | 100000000 | $10^{3000}$ | $10^{35660}$ |
| 100000 | 16 | 316 | 1600000 | $10^{10}$ | $10^{30000}$ | $10^{456574}$ |
| 1000000 | 19 | 1000 | 19000000 | $10^{12}$ | $10^{300000}$ | $10^{5565709}$ |

# Comparison of Computational Complexity



# Sorting Algorithms

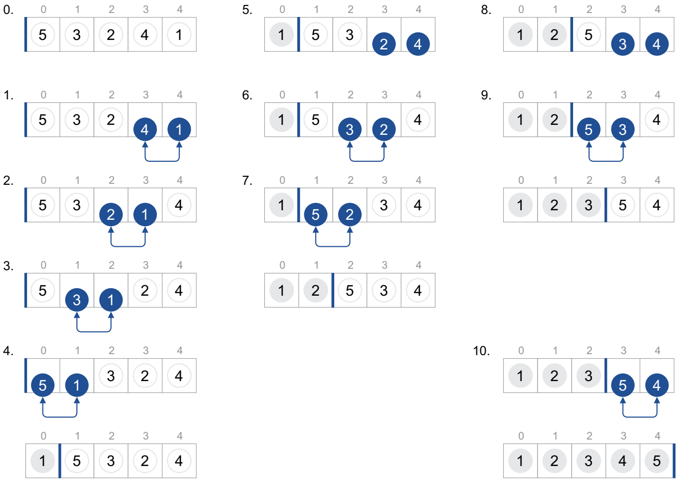
- Insertion Sort
- Bubble Sort
- Selection Sort
- Shell Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Counting Sort
- Bucket Sort
- etc.

# Bubble Sort

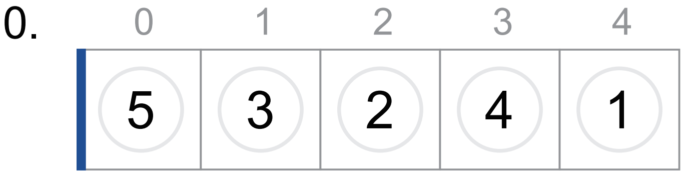
Bubble Sort is a popular sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order.

```
bubbleSort() // 0-origin
    for i = 0 to N-1
        for j = N-1 downto i + 1
            if A[j] < A[j - 1]
                swap A[j] and A[j - 1]
```

# Bubble Sort

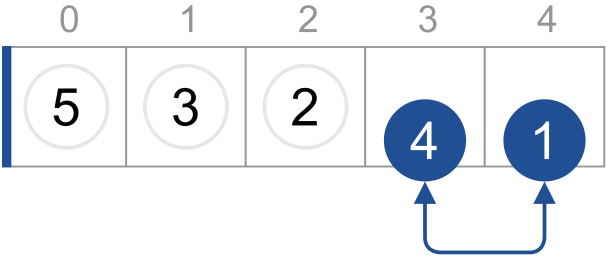


# Bubble Sort

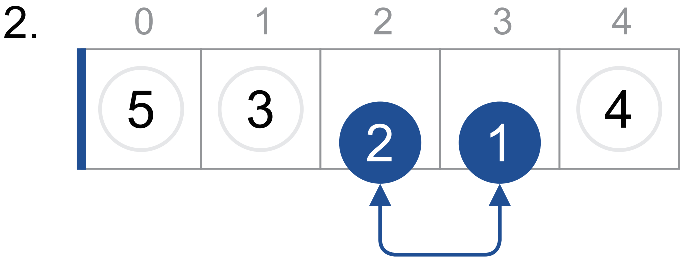


# Bubble Sort

1.

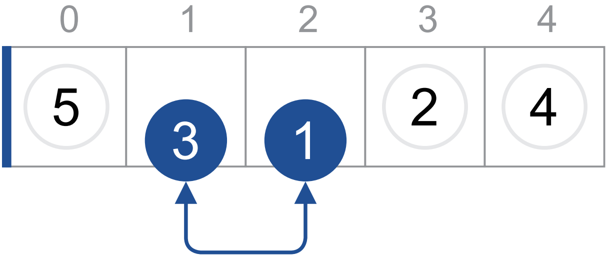


# Bubble Sort

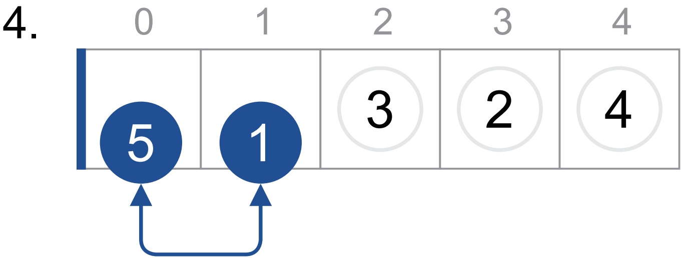


# Bubble Sort

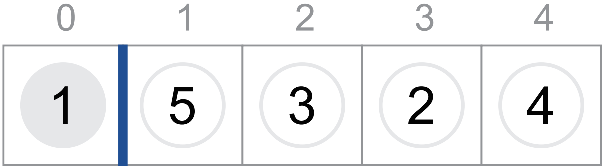
3.



# Bubble Sort

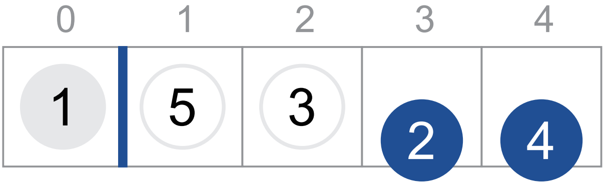


# Bubble Sort



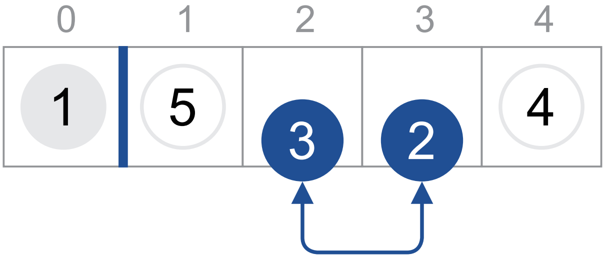
# Bubble Sort

5.



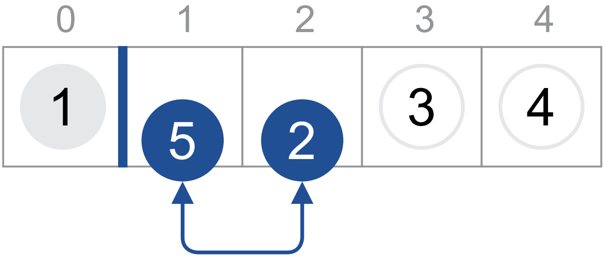
# Bubble Sort

6.

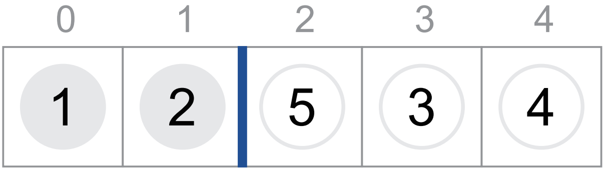


# Bubble Sort

7.

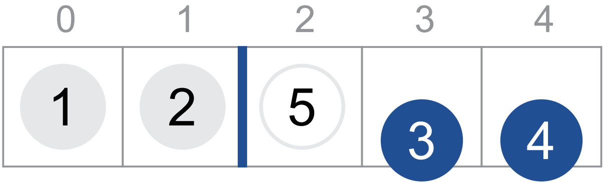


# Bubble Sort



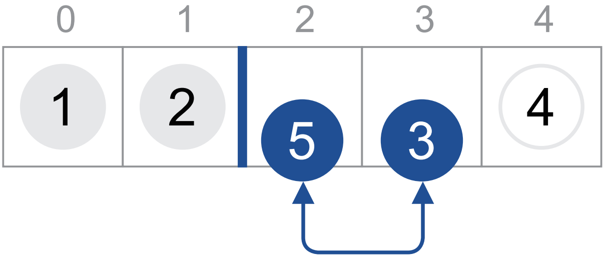
# Bubble Sort

8.

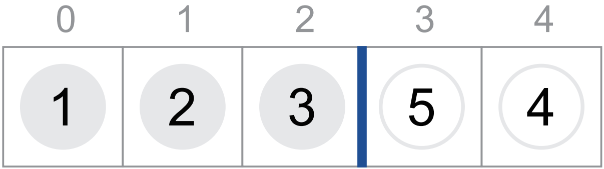


# Bubble Sort

9.

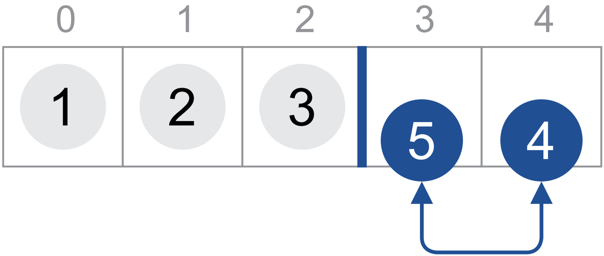


# Bubble Sort

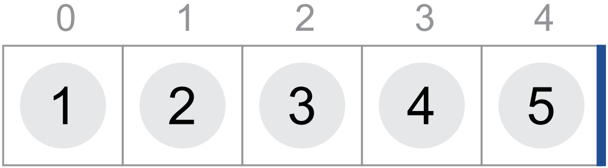


# Bubble Sort

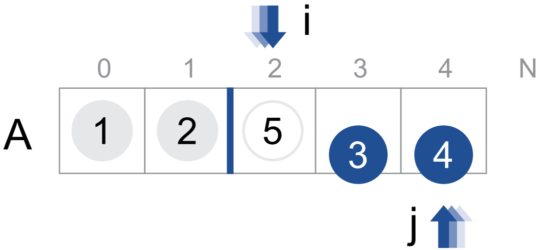
10.



# Bubble Sort



# Variables for Bubble Sort




| | |
|---|---|
| *A* | The input array with *N* integers |
| *i* | The loop variable which indicates the first element of the unsorted sub-array |
| *j* | The loop variable which indicates the two adjacency elements in the unsorted sub-array |

# Analysis of Bubble Sort

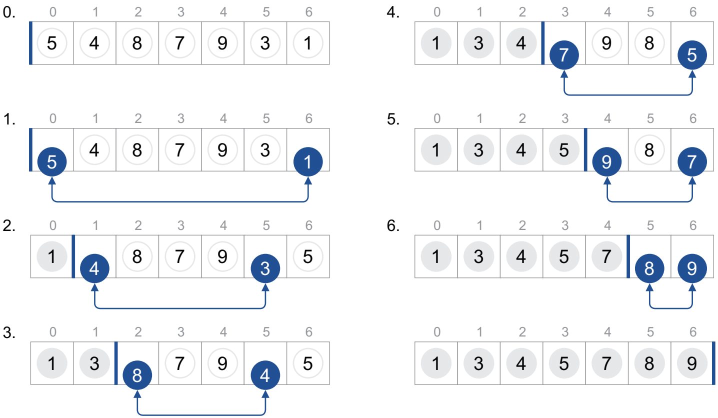
- $T(N) = (N-1) + (N-2) + (N-3) + ... + 1 = \frac{N(N-1)}{2}$
- Then, complexity of Bubble Sort is $O(N^2)$

# Selection Sort

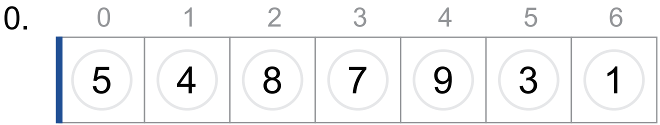
Consider sorting $N$ numbers stored in an array $A$ by first finding the smallest element of $A$ and exchanging it with the element in $A[0]$. Then find the second smallest element of $A$, and exchange it with $A[1]$. Continue in this manner for the first $n - 1$ elements of $A$.

```
selectionSort() // 0-origin
    for i = 0 to N - 2
        minj = i
        for j = i to N - 1
            if A[j] < A[minj]
                minj = j
        swap A[i] and A[minj]
```

# Selection Sort

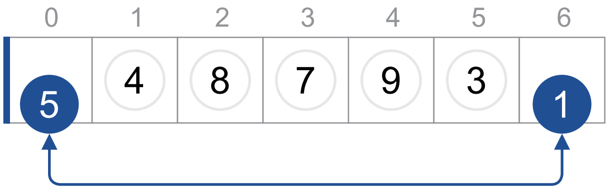


# Selection Sort



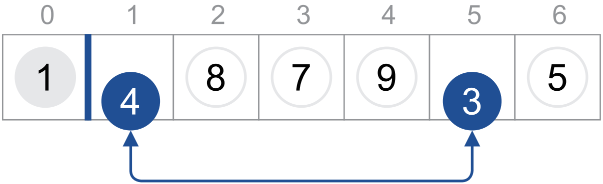
# Selection Sort

1.



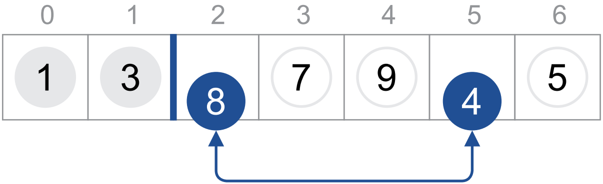
# Selection Sort

2.

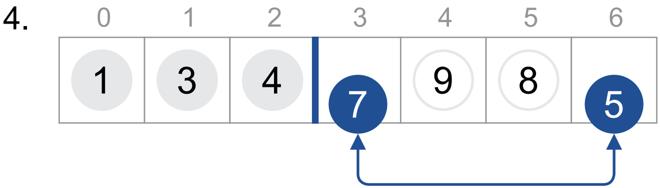


# Selection Sort

3.

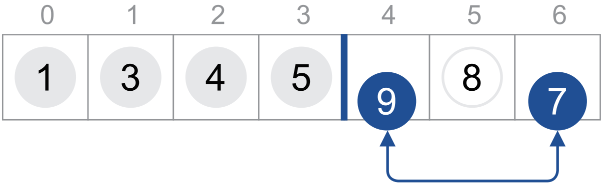


# Selection Sort



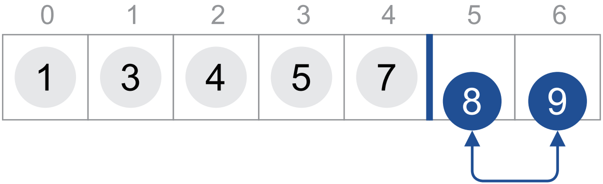
# Selection Sort

5.

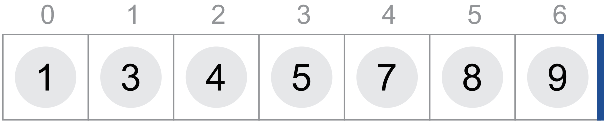


# Selection Sort

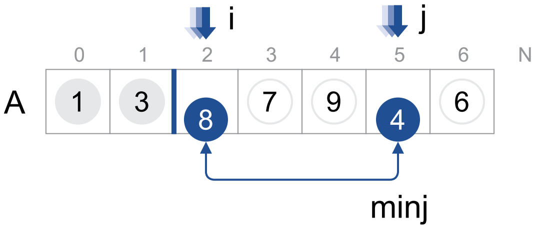
6.



# Selection Sort



# Variables for Selection Sort




| | |
|---|---|
| $A$ | The input array with $N$ integers |
| $i$ | The loop variable which indicates the first element of the unsorted sub-array |
| $j$ | The loop variable which traverses the unsorted sub-array |
| $minj$ | The pointer which indicates the minimum element in the unsorted sub-array |

# Analysis of Selection Sort

- $T(N) = (N-1) + (N-2) + (N-3) + ... + 1 = \frac{N(N-1)}{2}$
- Then, complexity of Selection Sort is $O(N^2)$

# Stability

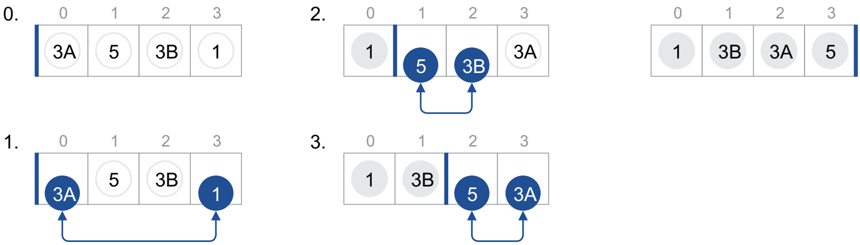
- Stability is an important property of sorting algorithms.
- In the stable sort, numbers with the same value appear in the output array in the same order as they do in the input array.
- That is, ties between two numbers are broken by the rule that whichever number appears first in the input array appears first in the output array.
- The property of stability is important only when satellite data are carried around with the element being sorted.

# Stability

- Bubble Sort is a stable sorting algorithm because it swaps adjacent elements, and only if the first one is strictly greater than the second one.
- Selection Sort is not a stable sorting algorithm because the order of elements with the same key can be changed after swap. It swaps elements which are not adjacent.
- What about other sorting algorithms?

# Stability

An example of unstable sort by Selection Sort.



# Reference

1. Introduction to Algorithms (third edition), Thomas H.Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. The MIT Press, 2012.