Bubble Sort

(What are Algorithms and How to Analyze Algorithms)

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 18, 2016

Bubble Sort

1. Sorting
2. Bubble Sort
3. Analysis

Bubble Sort

1. Sorting
2. Bubble Sort
3. Analysis

Sorting

The sorting problem:

Input: A sequence A of n integers $a_1 \; a_2 \; \cdots \; a_n$.

Output: A permutation $a'_1 \; a'_2 \; \ldots \; a'_n$ of A *s.t.*
$a'_1 \leq a'_2 \leq \cdots \leq a'_n$ (non-decreasing order).

Sorting

The sorting problem:

Input: A sequence A of n integers $a_1\ a_2\ \cdots\ a_n$.

Output: A permutation $a'_1\ a'_2\ \ldots\ a'_n$ of A *s.t.*
$a'_1 \le a'_2 \le \cdots \le a'_n$ (non-decreasing order).

$$3\ 1\ 4\ 2 \Longrightarrow 1\ 2\ 3\ 4$$

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Algorithms

An algorithm is a sequence of operations that transform the input into the output.

Correctness!

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Correctness!

Definiteness: precisely defined steps

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Correctness!

Definiteness: precisely defined steps

Finiteness: termination

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Correctness!

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: RAM (Random-Access Machine) model

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Correctness!

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: RAM (Random-Access Machine) model

- unrealistic: `sort` instruction

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Correctness!

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: RAM (Random-Access Machine) model

- unrealistic: `sort` instruction
- realistic: arithmetic, data movement, and control

Algorithms

An algorithm is a sequence of operations
that transform the input into the output.

Correctness!

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: RAM (Random-Access Machine) model

- unrealistic: `sort` instruction
- realistic: arithmetic, data movement, and control
- CAS[1] for sort: compare and swap if out-of-order

[1] Forget about that CAS in computer architecture.

Inversions

$$A = a_1, a_2, \ldots, a_n.$$

If $i < j$ and $a_i > a_j$, then (a_i, a_j) is an **inversion**.

Inversions

$$A = a_1, a_2, \ldots, a_n.$$

If $i < j$ and $a_i > a_j$, then (a_i, a_j) is an **inversion**.

$$A = 3, 1, 4, 2.$$

Inversions: $(3, 1), (3, 2), (4, 2)$

Inversions

$$A = a_1, a_2, \ldots, a_n.$$

If $i < j$ and $a_i > a_j$, then (a_i, a_j) is an **inversion**.

$$A = 3, 1, 4, 2.$$

Inversions: $(3, 1), (3, 2), (4, 2)$

Adjacent inversion: $j = i + 1$

Inversions

$$A = a_1, a_2, \ldots, a_n.$$

If $i < j$ and $a_i > a_j$, then (a_i, a_j) is an **inversion**.

$$A = 3, 1, 4, 2.$$

Inversions: $(3,1), (3,2), (4,2)$

Adjacent inversion: $j = i + 1$

Inversions

$$A \text{ is sorted } \implies A \text{ has no inversions}$$

Inversions

$$\begin{aligned} A \text{ is sorted } &\implies A \text{ has no inversions} \\ &\implies A \text{ has no adjacent inversions.} \end{aligned}$$

Inversions

$$\begin{aligned} A \text{ is sorted } &\implies A \text{ has no inversions} \\ &\implies A \text{ has no adjacent inversions.} \end{aligned}$$

$$A \text{ has no adjacent inversions } \implies \forall i \in [1, n-1] : a_i \leq a_{i+1}$$

Inversions

$$\begin{aligned} A \text{ is sorted } &\implies A \text{ has no inversions} \\ &\implies A \text{ has no adjacent inversions.} \end{aligned}$$

$$\begin{aligned} A \text{ has no adjacent inversions } &\implies \forall i \in [1, n-1] : a_i \le a_{i+1} \\ &\implies A \text{ is sorted.} \end{aligned}$$

Inversions

$$\begin{aligned} A \text{ is sorted } &\implies A \text{ has no inversions} \\ &\implies A \text{ has no adjacent inversions.} \end{aligned}$$

$$\begin{aligned} A \text{ has no adjacent inversions } &\implies \forall i \in [1, n-1] : a_i \leq a_{i+1} \\ &\implies A \text{ is sorted.} \end{aligned}$$

$$\boxed{A \text{ is sorted } \iff A \text{ has no adjacent inversions.}}$$

Bubble Sort

1. Sorting
2. Bubble Sort
3. Analysis

Bubble Sort

Basic idea: to eliminate all adjacent inversions

---

procedure BUBBLESORTOVERVIEW($A : a_1\ a_2\ \cdots\ a_n$)

  repeat

    Pick any i

    if $a_i > a_{i+1}$ **then** ▷ CAS

      SWAP(a_i, a_{i+1})

  until no adjacent inversions

Bubble Sort

Basic idea: to eliminate all adjacent inversions

procedure BubbleSortOverview($A : a_1\ a_2\ \cdots\ a_n$)
    repeat
        Pick any i ▷ **Definiteness!**
        if $a_i > a_{i+1}$ **then** ▷ CAS
            Swap(a_i, a_{i+1})
    until no adjacent inversions ▷ **Definiteness!**

Bubble Sort

Basic idea: to eliminate all adjacent inversions

```
procedure BubbleSortOverview(A : a_1 a_2 ··· a_n)
    repeat
        Pick any i                                  ▷ Definiteness!
        if a_i > a_{i+1} then                       ▷ CAS
            Swap(a_i, a_{i+1})
    until no adjacent inversions                    ▷ Finiteness! Definiteness!
```

Bubble Sort: Definiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        [          ]
        for i ← 1 : n − 1 do                    ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                [          ]
    until [          ]
```

Bubble Sort: Definiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        [swapped ← false]
        for i ← 1 : n − 1 do                          ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                [          ]
    until [          ]
```

Bubble Sort: Definiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        [swapped ← false]
        for i ← 1 : n − 1 do                    ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                [swapped ← true]
    until [            ]
```

Bubble Sort: Definiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        [swapped ← false]
        for i ← 1 : n − 1 do                          ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                [swapped ← true]
    until [swapped = false]
```

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2

4

1

3

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2
4	4
1	$\boxed{1}$
3	$\boxed{3}$

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2	2
4	4	4
1	1	3
3	3	1

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2	2	2
4	4	4	4
1	1	3	3
3	3	1	1

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2	2	2	4
4	4	4	4	2
1	1	3	3	3
3	3	1	1	1

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2	2	2	4	4
4	4	4	4	2	2
1	1	3	3	3	3
3	3	1	1	1	1

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2	2	2	4	4	4
4	4	4	4	2	2	3
1	1	3	3	3	3	2
3	3	1	1	1	1	1

Bubble Sort: Example

procedure BUBBLESORT($A : a_1\ a_2\ \cdots\ a_n$)
  repeat
    swapped $\leftarrow$ false
    for $i \leftarrow 1 : n-1$ **do** ▷ Pick i
      if $a_i > a_{i+1}$ **then**
        SWAP(a_i, a_{i+1})
        swapped $\leftarrow$ true
  until swapped $=$ false ▷ No swaps

2	2	2	2	4	4	4	4
4	4	4	4	2	2	3	3
1	1	3	3	3	3	2	2
3	3	1	1	1	1	1	1

Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

2	2	2	2	4	4	4	4	4
4	4	4	4	2	2	3	3	3
1	1	3	3	3	3	2	2	2
3	3	1	1	1	1	1	1	1

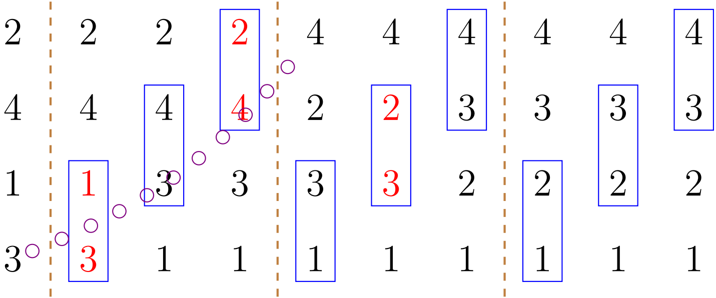
Bubble Sort: Example

procedure BUBBLESORT($A : a_1\ a_2\ \cdots\ a_n$)
    repeat
        swapped $\leftarrow$ false
        for $i \leftarrow 1 : n-1$ **do** ▷ Pick i
            if $a_i > a_{i+1}$ **then**
                SWAP(a_i, a_{i+1})
                swapped $\leftarrow$ true
    until swapped $=$ false ▷ No swaps

2	2	2	2	4	4	4	4	4	4
4	4	4	4	2	2	3	3	3	3
1	1	3	3	3	3	2	2	2	2
3	3	1	1	1	1	1	1	1	1

Bubble Sort: Example

procedure BubbleSort($A : a_1\ a_2\ \cdots\ a_n$)
    repeat
        swapped $\leftarrow$ false
        for $i \leftarrow 1 : n-1$ **do** ▷ Pick i
            if $a_i > a_{i+1}$ **then**
                Swap(a_i, a_{i+1})
                swapped $\leftarrow$ true
    until swapped = false ▷ No swaps



Bubble Sort: Example

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                    ▷ No swaps
```

2	2	2	2	4	4	4	4	4	4
4	4	4	4	2	2	3	3	3	3
1	1	3	3	3	3	2	2	2	2
3	3	1	1	1	1	1	1	1	1

Bubble Sort: Finiteness

procedure BUBBLESORT($A : a_1 \; a_2 \; \cdots \; a_n$)
  repeat
    swapped $\leftarrow$ false
    for $i \leftarrow 1 : n-1$ **do** ▷ Pick i
      if $a_i > a_{i+1}$ **then**
        SWAP(a_i, a_{i+1})
        swapped $\leftarrow$ true
  until swapped $=$ false ▷ No swaps

Bubble Sort: Finiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

The inner "**for**" loops:

1) $\exists$ loop : no swaps $\implies$ swapped = false $\implies$ terminates

Bubble Sort: Finiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                   ▷ No swaps
```

The inner "**for**" loops:

1) $\exists$ loop : no swaps $\implies$ swapped = false $\implies$ terminates

2) $\forall$ loop : has swaps

Bubble Sort: Finiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false                    ▷ No swaps
```

The inner "**for**" loops:

1) $\exists$ loop : no swaps $\implies$ swapped = false $\implies$ terminates

2) $\forall$ loop : has swaps **Impossible!**

Bubble Sort: Finiteness

Effects of $\textsc{Swap}(a_i, a_{i+1})$ on adjacent inversions:

$$
\begin{array}{cccccccc}
5 & 3 & 1 & 6 & 7 & 2 & 4 & 8 \\
5 & 3 & 1 & 6 & 2 & 7 & 4 & 8
\end{array}
$$

Bubble Sort: Finiteness

Effects of $\textsc{Swap}(a_i, a_{i+1})$ on adjacent inversions:

$$5 \quad 3 \quad 1 \quad 6 \quad 7 \quad 2 \quad 4 \quad 8$$

$$5 \quad 3 \quad 1 \quad 6 \quad 2 \quad 7 \quad 4 \quad 8$$

$-1 : (7, 2)$

Bubble Sort: Finiteness

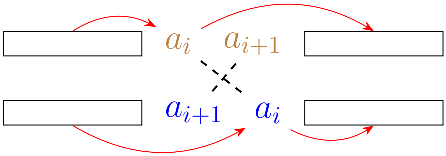
Effects of $\textsc{Swap}(a_i, a_{i+1})$ on adjacent inversions:

$$\begin{array}{cccccccc} 5 & 3 & 1 & 6 & 7 & 2 & 4 & 8 \\ 5 & 3 & 1 & \color{red}{6} & \color{blue}{2} & \color{blue}{7} & 4 & 8 \end{array}$$

$\color{blue}{-1 : (7, 2)}$ $\quad\quad$ $\color{red}{+2 : (6, 2), (7, 4)}$

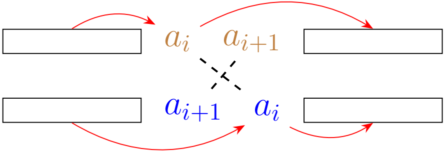
Bubble Sort: Finiteness

Effects of $\text{SWAP}(a_i, a_{i+1})$ on #inversions:



Bubble Sort: Finiteness

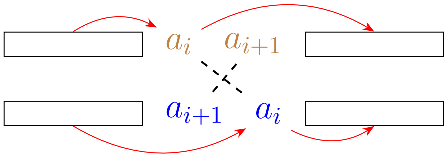
Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions:



$-1 : (a_i, a_{i+1})$

Bubble Sort: Finiteness

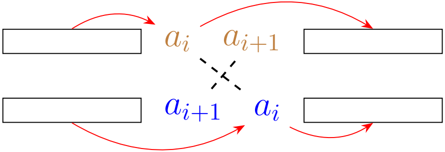
Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions:



$-1 : (a_i, a_{i+1})$ $+0$

Bubble Sort: Finiteness

Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions:



$-1 : (a_i, a_{i+1})$ $+0$

Total #inversions is finite.

Bubble Sort: Correctness

$$\boxed{A \text{ is sorted} \iff A \text{ has no adjacent inversions.}}$$

Bubble Sort: Correctness

$$\boxed{A \text{ is sorted} \iff A \text{ has no adjacent inversions.}}$$

Finiteness

Bubble Sort: Correctness

$$\boxed{A \text{ is sorted} \iff A \text{ has no adjacent inversions.}}$$

$$\textbf{Finiteness} \implies \exists \text{ loop} : \text{no swaps}$$

Bubble Sort: Correctness

$$\boxed{A \text{ is sorted} \iff A \text{ has no adjacent inversions.}}$$

Finiteness $\implies \exists$ loop : no swaps

$\implies A$ has no adjacent inversions any more

Bubble Sort: Correctness

$$\boxed{A \text{ is sorted} \iff A \text{ has no adjacent inversions.}}$$

$$
\begin{aligned}
\textbf{Finiteness} \quad &\Longrightarrow \exists \text{ loop} : \text{ no swaps} \\
&\Longrightarrow A \text{ has no adjacent inversions any more} \\
&\Longrightarrow A \text{ is already sorted.}
\end{aligned}
$$

Bubble Sort: Correctness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Loop invariant?
            if a_i > a_{i+1} then                          ▷ CAS
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false
```

Bubble Sort: Correctness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Loop invariant?
            if a_i > a_{i+1} then                        ▷ CAS
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false
```

Loop invariant:

Before the k-th $(k \geq 1)$ "**for**" loop, $a_{n-(k-1)} \cdots a_n$
(1) consists of the largest $(k-1)$ elements
(2) in sorted order.

Bubble Sort: Correctness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do                ▷ Loop invariant?
            if a_i > a_{i+1} then            ▷ CAS
                Swap(a_i, a_{i+1})
                swapped ← true
    until swapped = false
```

Loop invariant:

Before the k-th ($k \geq 1$) "**for**" loop, $a_{n-(k-1)} \cdots a_n$
(1) consists of the largest $(k-1)$ elements
(2) in sorted order.

Correctness: Initialization + Maintenance + Termination

Optimizing Bubble Sort

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    n ← len(A)
    repeat
        swapped ← false
        for i ← 1 : n − 1 do
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
        n ← n − 1                          ▷ One maximal bubbles up
    until swapped = false
```

Optimizing Bubble Sort

$$3 \quad 1 \quad 4 \quad 2 \quad 5 \quad 6 \quad 7$$

$$1 \quad 3 \quad 2 \quad \color{red}{4} \quad \color{red}{5} \quad 6 \quad 7$$

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        swapped ← false
        lsp ← 0                      ▷ lsp: the last swap position
        for i ← 1 : n − 1 do
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                swapped ← true
                lsp ← i              ▷ Update lsp
        n ← lsp                      ▷ Elements after lsp are sorted
    until swapped = false
```

Optimizing Bubble Sort

3	1	4	2	5	6	7
1	3	2	4	5	6	7

procedure BubbleSort($A : a_1\ a_2\ \cdots\ a_n$)
    repeat
        swapped $\leftarrow$ false
        lsp $\leftarrow 0$ ▷ lsp: the last swap position
        for $i \leftarrow 1 : n-1$ **do**
            if $a_i > a_{i+1}$ **then**
                Swap(a_i, a_{i+1})
                swapped $\leftarrow$ true
                lsp $\leftarrow$ i ▷ Update lsp
        $n \leftarrow$ lsp ▷ Elements after lsp are sorted
    until swapped $=$ false

Optimizing Bubble Sort

3	1	4	2	5	6	7
1	3	2	4	5	6	7

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        lsp ← 0                          ▷ lsp: the last swap position
        for i ← 1 : n − 1 do
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                lsp ← i                  ▷ Update lsp
        n ← lsp                          ▷ Elements after lsp are sorted
    until n = 0
```

Bubble Sort

1. Sorting
2. Bubble Sort
3. Analysis

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\Longrightarrow$ Quantitative finiteness

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\implies$ Quantitative finiteness
- Time on real computers varies $\implies$ #Ops on RAM model:

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\implies$ Quantitative finiteness
- Time on real computers varies $\implies$ #Ops on RAM model:

$$|P|: \ \#\text{Passes} \qquad\qquad (\text{the "}\textbf{for}\text{" loops})$$

$$|C|: \ \#\text{Comparisons} \qquad\qquad (\textbf{if } a_{i-1} > a_i)$$

$$|S|: \ \#\text{Swaps} \qquad\qquad (\textsc{Swap}(a_{i-1}, a_i))$$

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\implies$ Quantitative finiteness
- Time on real computers varies $\implies$ #Ops on RAM model:

$$
\begin{aligned}
|P| &: \text{\#Passes} && \text{(the "\textbf{for}" loops)} \\
|C| &: \text{\#Comparisons} && (\textbf{if } a_{i-1} > a_i) \\
|S| &: \text{\#Swaps} && (\textsc{Swap}(a_{i-1}, a_i))
\end{aligned}
$$

$$|C| \geq |S|$$

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\Longrightarrow$ Quantitative finiteness
- Time on real computers varies $\Longrightarrow$ #Ops on RAM model:

$|P|$: #Passes (the "**for**" loops)

$|C|$: #Comparisons ($\textbf{if } a_{i-1} > a_i$)

$|S|$: #Swaps ($\textsc{Swap}(a_{i-1}, a_i)$)

$$|C| \geq |S|$$

- Different inputs $\Longrightarrow$ different execution time:

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\Longrightarrow$ Quantitative finiteness
- Time on real computers varies $\Longrightarrow$ #Ops on RAM model:

$|P|$: #Passes (the "**for**" loops)

$|C|$: #Comparisons ($\textbf{if } a_{i-1} > a_i$)

$|S|$: #Swaps ($\textsc{Swap}(a_{i-1}, a_i)$)

$$|C| \geq |S|$$

- Different inputs $\Longrightarrow$ different execution time:
  - Best-case, worst-case, and average-case analysis

Best-case and Worst-case Analysis

Best-case: Worst-case:

$$
\begin{aligned}
|P| &= (\qquad);\\
|C| &= (\qquad);\\
|S| &= (\qquad).
\end{aligned}
$$

Best-case and Worst-case Analysis

Best-case: 1 2 3 4 5 6 7 8

	Best-case: ascendingly sorted	Worst-case:

$$|P| = (\qquad);$$

$$|C| = (\qquad);$$

$$|S| = (\qquad).$$

Best-case and Worst-case Analysis

Best-case: 1 2 3 4 5 6 7 8

$$\underbrace{\text{Best-case:}}_{\text{ascendingly sorted}} \qquad \underline{\text{Worst-case:}}$$

$$|P| = (\min : 1, \qquad);$$

$$|C| = (\min : n-1, \qquad);$$

$$|S| = (\min : 0, \qquad).$$

Best-case and Worst-case Analysis

Best-case: 1 2 3 4 5 6 7 8

$$\begin{array}{lll} & \underline{\text{Best-case:}} & \underline{\text{Worst-case:}} \\ & \text{ascendingly sorted} & \text{descendingly sorted} \\ |P| = (& \min : 1, &); \\ |C| = (& \min : n-1, &); \\ |S| = (& \min : 0, &). \end{array}$$

Worst-case: 8 7 6 5 4 3 2 1

Best-case and Worst-case Analysis

Best-case: 1 2 3 4 5 6 7 8

$$
\begin{aligned}
& & & \underline{\text{Best-case:}} & & \underline{\text{Worst-case:}} \\
& & & \text{ascendingly sorted} & & \text{descendingly sorted} \\
|P| & = (& & \min : 1, & & \max : n); \\
|C| & = (& & \min : n-1, & & \max : \tfrac{n^2-n}{2}); \\
|S| & = (& & \min : 0, & & \max : \tfrac{n^2-n}{2}).
\end{aligned}
$$

Worst-case: 8 7 6 5 4 3 2 1

$|S|$: #Swaps (Average Analysis)

Assumptions on inputs:

1. The input is a random permutation
2. All numbers are distinct

$|S|$: #Swaps (Average Analysis)

Assumptions on inputs:

1. The input is a random permutation
2. All numbers are distinct

$$\boxed{\textsc{Swap}(a_i, a_{i+1}) \implies -1 \text{ inversion}}$$

$|S|$: #Swaps (Average Analysis)

Assumptions on inputs:

1. The input is a random permutation
2. All numbers are distinct

$$\boxed{\textsc{Swap}(a_i, a_{i+1}) \implies -1 \text{ inversion}}$$

Question: What is the expected #inversions?

$|S|$: #Swaps (Average Analysis)

I_{ij} : indicator of inversion (a_i, a_j)

$|S|$: #Swaps (Average Analysis)

$$I_{ij} : \text{indicator of inversion } (a_i, a_j)$$

$$X = \sum_j \sum_{i<j} I_{ij}$$

$|S|$: #Swaps (Average Analysis)

$$I_{ij} : \text{indicator of inversion } (a_i, a_j)$$

$$X = \sum_j \sum_{i<j} I_{ij}$$

$$E(X) = E(\sum_j \sum_{i<j} I_{ij}) = \sum_j \sum_{i<j} E(I_{ij})$$

$|S|$: #Swaps (Average Analysis)

$$I_{ij} : \text{indicator of inversion } (a_i, a_j)$$

$$X = \sum_j \sum_{i<j} I_{ij}$$

$$E(X) = E(\sum_j \sum_{i<j} I_{ij}) = \sum_j \sum_{i<j} E(I_{ij})$$

$$E(I_{ij}) = P\{I_{ij}\} = \frac{1}{2}$$

$|S|$: #Swaps (Average Analysis)

$$I_{ij} : \text{indicator of inversion } (a_i, a_j)$$

$$X = \sum_j \sum_{i<j} I_{ij}$$

$$E(X) = E(\sum_j \sum_{i<j} I_{ij}) = \sum_j \sum_{i<j} E(I_{ij})$$

$$E(I_{ij}) = P\{I_{ij}\} = \frac{1}{2}$$

$$E(X) = \sum_j \sum_{i<j} \frac{1}{2} = \binom{n}{2} \cdot \frac{1}{2} = \frac{n(n-1)}{4}$$

More Sorting Algorithms

It took a good deal of work to analyze the bubble sort; and although [. . .], the results are disappointing since they tell us that the bubble sort isn't really very good at all.

— *Donald E. Knuth*

More Sorting Algorithms

It took a good deal of work to analyze the bubble sort; and although [. . .], the results are disappointing since they tell us that the bubble sort isn't really very good at all.

— *Donald E. Knuth*

faster: $O(n^2) \to O(n \lg n)$?

More Sorting Algorithms

> *It took a good deal of work to analyze the bubble sort; and although [...], the results are disappointing since they tell us that the bubble sort isn't really very good at all.*
>
> — *Donald E. Knuth*

$$\text{faster: } O(n^2) \to O(n \lg n)?$$

$$\ldots \text{and faster: } O(n \lg n) \to O(n)?$$

More Sorting Algorithms

It took a good deal of work to analyze the bubble sort; and although [...], the results are disappointing since they tell us that the bubble sort isn't really very good at all.

— *Donald E. Knuth*

faster: $O(n^2) \to O(n \lg n)$?

... and faster: $O(n \lg n) \to O(n)$?

hengxin0912@gmail.com