Bubble Sort

(What are Algorithms and How to Analyze Algorithms)

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 17, 2016

Bubble Sort

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.* $a'_1 \leq a'_2 \leq \cdots \leq a'_n$ (non-decreasing order).

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

Sorting Algorithms

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

Sorting Algorithms

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

Definiteness: precisely defined steps

Sorting Algorithms

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

Definiteness: precisely defined steps

Finiteness: termination

Sorting Algorithms

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

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: operations are sufficiently basic

Sorting Algorithms

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

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: operations are sufficiently basic

- RAM (Random-Access Machine) model

Sorting Algorithms

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

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: operations are sufficiently basic

- RAM (Random-Access Machine) model
- unrealistic: `sort` instruction

Sorting Algorithms

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

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: operations are sufficiently basic

- RAM (Random-Access Machine) model
- unrealistic: `sort` instruction
- realistic: arithmetic, data movement, and control

Sorting Algorithms

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

Definiteness: precisely defined steps

Finiteness: termination

Effectiveness: operations are sufficiently basic

- RAM (Random-Access Machine) model
- unrealistic: `sort` instruction
- realistic: arithmetic, data movement, and control
- CAS for sort: compare and swap if out-of-order

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

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

Inversions

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

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

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 ··· 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 ··· a_n)
    repeat
        Pick any i
        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
        if a_i > a_{i+1} then                ▷ CAS
            Swap(a_i, a_{i+1})
    until no adjacent inversions
```

Definiteness!

Finiteness!

Bubble Sort

```
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

```
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

---

procedure $\textsc{BubbleSort}(A : a_1\ a_2\ \cdots\ a_n)$
    repeat
        $\boxed{\text{swapped} \leftarrow \text{false}}$
        for $i \leftarrow 1 : n-1$ **do** ▷ Pick i
            if $a_i > a_{i+1}$ **then**
                $\textsc{Swap}(a_i, a_{i+1})$
                $\boxed{\text{swapped} \leftarrow \text{true}}$
    until $\boxed{\phantom{\text{swapped}}}$

Bubble Sort

procedure $\textsc{BubbleSort}(A : a_1\ a_2\ \cdots\ a_n)$

  repeat

    $\boxed{\text{swapped} \gets \text{false}}$

    for $i \gets 1 : n-1$ **do** $\triangleright$ Pick i

      if $a_i > a_{i+1}$ **then**

        $\textsc{Swap}(a_i, a_{i+1})$

        $\boxed{\text{swapped} \gets \text{true}}$

  until $\boxed{\text{swapped} = \text{false}}$

Bubble Sort

```
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]
```

Definiteness!

Bubble Sort

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

Bubble Sort

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

Finiteness!

Bubble Sort

```
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
```

Finiteness!

The inner "**for**" loops:

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

Bubble Sort

```
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
```

Finiteness!

The inner "**for**" loops:

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

2) $\forall$ loop : has swaps

Bubble Sort

```
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
```

Finiteness!

The inner "**for**" loops:

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

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

Bubble Sort

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

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

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

$-1 : (7, 2)$

Bubble Sort

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

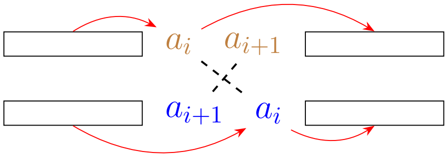
$$5 \quad 3 \quad 1 \quad 6 \quad 7 \quad 2 \quad 4 \quad 8$$

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

$-1 : (7, 2)$ $\qquad$ $+2 : (6, 2), (7, 4)$

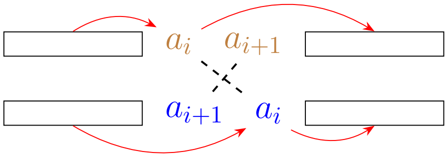
Bubble Sort

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



Bubble Sort

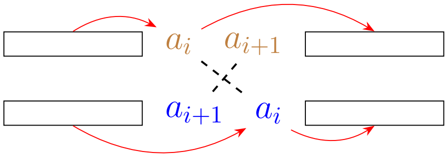
Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions:



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

Bubble Sort

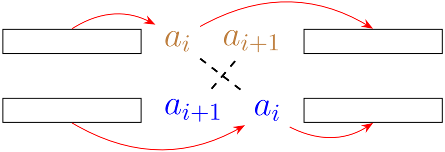
Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions:



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

Bubble Sort

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



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

Total #inversions is finite.

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

```
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

```
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

```
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

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\Longrightarrow$ Quantitative finiteness

Time Complexity of Bubble Sort

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

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

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

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

Time Complexity of Bubble Sort

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

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

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

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

- Different inputs $\implies$ 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:

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

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

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

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

Best-case and Worst-case Analysis

Best-case: Worst-case:

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

Best-case and Worst-case Analysis

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

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

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

Best-case and Worst-case Analysis

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

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

$$\begin{aligned}
|P| &= (\min : 1, \qquad\qquad\qquad); \\
|C| &= (\min : n - 1, \qquad\qquad\qquad); \\
|S| &= (\min : 0, \qquad\qquad\qquad).
\end{aligned}$$

Best-case and Worst-case Analysis

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

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

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)

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

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

I_{ij} : 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)

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

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

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

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

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