Bubble Sort

(A Taste of Algorithms: Definition, Design, and Analysis)

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 19, 2016

Bubble Sort

1. The Sorting Problem
2. Bubble Sort
3. Analysis of Bubble Sort

1. The Sorting Problem
2. Bubble Sort
3. Analysis of Bubble Sort

Algorithms

What is an algorithm?

Algorithms

What is an algorithm? What is computation?

Algorithms

What is an algorithm? What is computation?

Algorithms

What is an algorithm? What is computation?

Correctness!

Algorithms

What is an algorithm? What is computation?

Correctness!

Definiteness: precisely defined operations

Algorithms

What is an algorithm? What is computation?

Correctness!

Definiteness: precisely defined operations

Finiteness: termination

Algorithms

What is an algorithm? What is computation?

$$\xrightarrow{\text{Input}} \boxed{\begin{array}{c}\text{Algorithm}\\ \text{(sequence of operations)}\end{array}} \xrightarrow{\text{Output}}$$

Correctness!

Definiteness:	precisely defined operations
Finiteness:	termination
Effectiveness:	a reasonable model; basic operations

Algorithms

What is an algorithm? What is computation?

Input → Algorithm (sequence of operations) → Output

Correctness!

Definiteness: precisely defined operations

Finiteness: termination

Effectiveness: a reasonable model; basic operations

- for sorting: compare, swap

Sorting

The sorting problem:

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

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

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

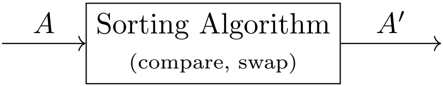
Sorting

The sorting problem:

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

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

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



Inversions

$$A = a_1 \quad a_2 \quad \cdots \quad a_i \quad \cdots \quad a_j \quad \cdots \quad a_n.$$

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

Inversions

$$A = a_1 \quad a_2 \quad \cdots \quad a_i \quad \cdots \quad a_j \quad \cdots \quad a_n.$$

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

Adjacent inversion: (a_i, a_{i+1})

Inversions

$$A = a_1 \quad a_2 \quad \cdots \quad a_i \quad \cdots \quad a_j \quad \cdots \quad a_n.$$

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

Adjacent inversion: (a_i, a_{i+1})

$$3 \quad 1 \quad 4 \quad 2$$

#inversions = 3
#adjacent inversions = 2

Inversions

$$A = a_1 \quad a_2 \quad \cdots \quad a_i \quad \cdots \quad a_j \quad \cdots \quad a_n.$$

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

Adjacent inversion: (a_i, a_{i+1})

3 1 4 2

$\#\text{inversions} = 3$
$\#\text{adjacent inversions} = 2$

4 3 2 1

$\#\text{inversions} = 3 + 2 + 1 = 6$
$\#\text{adjacent inversions} = 3$

Inversions

$$A = a_1 \quad a_2 \quad \cdots \quad a_i \quad \cdots \quad a_j \quad \cdots \quad a_n.$$

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

Adjacent inversion: (a_i, a_{i+1})

3 1 4 2

#inversions = 3
#adjacent inversions = 2

4 3 2 1

#inversions = 3 + 2 + 1 = 6
#adjacent inversions = 3

1 2 3 4

#inversions = 0
#adjacent inversions = 0

Inversions

Theorem: A is sorted $\iff$ A has no adjacent inversions.

Inversions

Theorem: A is sorted $\iff$ A has no adjacent inversions.

A is sorted $\implies$ A has no adjacent inversions.

Inversions

Theorem: A is sorted $\iff$ A has no adjacent inversions.

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

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

Inversions

Theorem: A is sorted $\iff$ A has no adjacent inversions.

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

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

Bubble Sort

1. The Sorting Problem
2. Bubble Sort
3. Analysis of Bubble Sort

Bubble Sort: Basic Idea

Basic idea: to eliminate all adjacent inversions

Bubble Sort: Basic Idea

Basic idea: to eliminate all adjacent inversions

---

repeat
  pick any i
  if $a_i > a_{i+1}$ **then**
    SWAP(a_i, a_{i+1})
until no adjacent inversions

Bubble Sort: Basic Idea

Basic idea: to eliminate all adjacent inversions

```
repeat
    pick any i                                ▷ Definiteness!
    if a_i > a_{i+1} then
        SWAP(a_i, a_{i+1})
until no adjacent inversions                  ▷ Definiteness!
```

Bubble Sort: Basic Idea

Basic idea: to eliminate all adjacent inversions

repeat
    pick any i ▷ **Definiteness!**
    if $a_i > a_{i+1}$ **then**
        SWAP(a_i, a_{i+1})
until no adjacent inversions ▷ **Finiteness! Definiteness!**

Bubble Sort: Basic Idea

Basic idea: to eliminate all adjacent inversions

---

repeat
  pick any i ▷ **Definiteness!**
  if $a_i > a_{i+1}$ **then**
    $\textsc{Swap}(a_i, a_{i+1})$
until no adjacent inversions ▷ **Finiteness! Definiteness!**

$$\text{Finiteness} \implies \text{Correctness}.$$

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

Bubble Sort: Definiteness

```
procedure BubbleSort(A : a_1 a_2 ··· a_n)
    repeat
        
        for            do                         ▷ Pick i
            if a_i > a_{i+1} then
                Swap(a_i, a_{i+1})
                
    until no adjacent inversions
```

Bubble Sort: Definiteness

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

  repeat

    for $\boxed{i \leftarrow 1 : n-1}$ **do** ▷ Pick i

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

        Swap(a_i, a_{i+1})

  until $\boxed{\text{no adjacent inversions}}$

Bubble Sort: Definiteness

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

  until no adjacent inversions

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 [no adjacent inversions]
```

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 [no adjacent inversions]
```

Bubble Sort: Definiteness

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

Bubble Sort: Example

2

4

1

3

Bubble Sort: Example

2	2
4	4
1	1
3	3

Bubble Sort: Example

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

Bubble Sort: Example

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

Bubble Sort: Example

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

Bubble Sort: Example

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

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

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

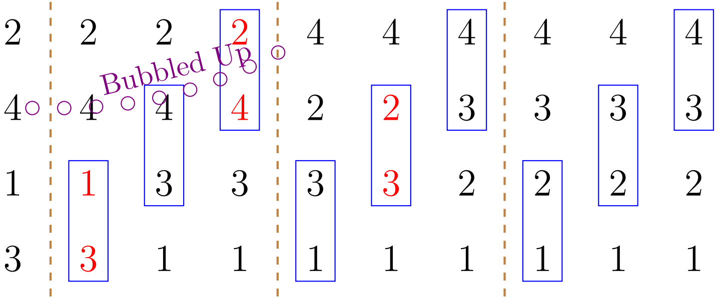
Bubble Sort: Example

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

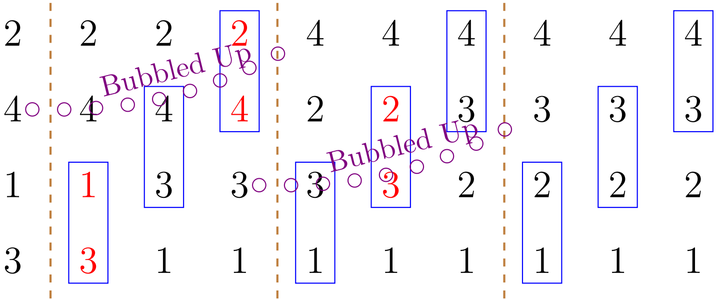
Bubble Sort: Example

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

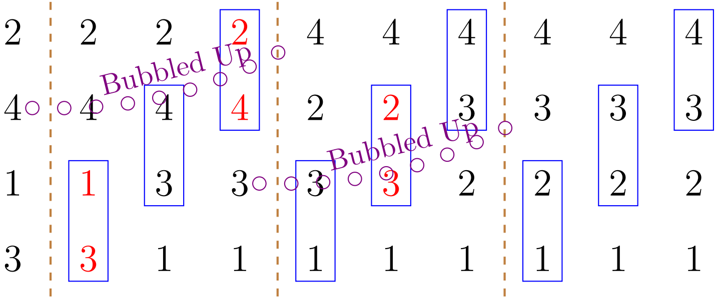
Bubble Sort: Example



Bubble Sort: Example



Bubble Sort: Example



After each "**for**" loop, one more element is bubbled up to its final position.

Optimizing Bubble Sort (I)[1]

After each "**for**" loop, one more element is bubbled up to its final position.

---

procedure $\textsc{BubbleSort}(A : a_1\ a_2\ \cdots\ a_n)$
    $n \leftarrow \text{len}(A)$
    repeat
        swapped $\leftarrow$ false
        for $i \leftarrow 1 : n-1$ **do**
            if $a_i > a_{i+1}$ **then**
                $\textsc{Swap}(a_i, a_{i+1})$
                swapped $\leftarrow$ true
        $n \leftarrow n - 1$ $\triangleright$ One maximal bubbles up
    until swapped $=$ false

---

[1] See Appendix for "Optimizing Bubble Sort (II)".

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

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

The inner "**for**" loops:

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

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

Bubble Sort: Finiteness

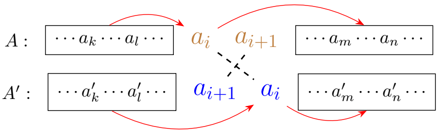
$\boxed{\text{Fact: total \#inversions is finite.}}$

[2] Not on #adjacent inversions! Think about it.

Bubble Sort: Finiteness

$\boxed{\text{Fact: total \#inversions is finite.}}$

Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions[2]:

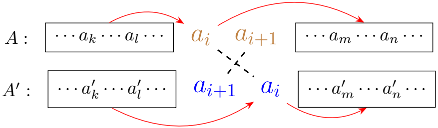


[2]Not on #adjacent inversions! Think about it.

Bubble Sort: Finiteness

$\boxed{\text{Fact: total \#inversions is finite.}}$

Effects of $\text{SWAP}(a_i, a_{i+1})$ on #inversions[2]:



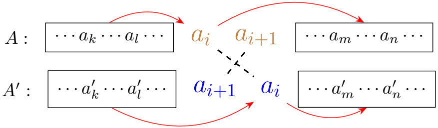
$$-1 : (a_i, a_{i+1})$$

[2] Not on #adjacent inversions! Think about it.

Bubble Sort: Finiteness

$\boxed{\text{Fact: total \#inversions is finite.}}$

Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions[2]:



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

$+0$: relative order between any other two elements does not change!

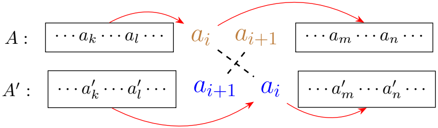
$$\langle a_k, a_l \rangle, \langle a_m, a_n \rangle, \langle a_k, a_i \rangle, \langle a_i, a_m \rangle$$

[2] Not on #adjacent inversions! Think about it.

Bubble Sort: Finiteness

$\boxed{\text{Fact: total \#inversions is finite.}}$

Effects of $\textsc{Swap}(a_i, a_{i+1})$ on #inversions[2]:



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

$+0$: relative order between any other two elements does not change!

$$\langle a_k, a_l \rangle, \langle a_m, a_n \rangle, \langle a_k, a_i \rangle, \langle a_i, a_m \rangle$$

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

[2] Not on #adjacent inversions! Think about it.

Bubble Sort

1. The Sorting Problem
2. Bubble Sort
3. Analysis of Bubble Sort

Time Complexity of Bubble Sort

- Finiteness is NOT enough $\Longrightarrow$ Quantifying finiteness

Time Complexity of Bubble Sort

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

Time Complexity of Bubble Sort

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

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

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

Time Complexity of Bubble Sort

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

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

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

- Different inputs $\implies$ $|C|$ and $|S|$ vary:

Time Complexity of Bubble Sort

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

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

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

- Different inputs $\implies$ $|C|$ and $|S|$ vary:
  - Best-case, worst-case, and average-case analysis

Best-case and Worst-case Analysis

Best-case: Worst-case:

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

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

Best-case and Worst-case Analysis

Best-case: $1 \quad 2 \cdots n$

$$
\begin{aligned}
|C| &= (\underbrace{\qquad\qquad\qquad}_{\substack{\text{Best-case:}\\ \text{non-decreasingly sorted}}} \quad \underbrace{\qquad\qquad\qquad}_{\text{Worst-case:}});\\
|S| &= (\qquad\qquad\qquad\qquad\qquad\qquad).
\end{aligned}
$$

Best-case and Worst-case Analysis

Best-case: $1 \quad 2 \cdots n$

$$
\begin{aligned}
|C| &= \Big(\underbrace{\min : n-1,}_{\text{Best-case: non-decreasingly sorted}} \qquad \underbrace{\qquad\qquad}_{\text{Worst-case:}} \Big); \\
|S| &= \Big(\min : 0, \qquad\qquad \Big).
\end{aligned}
$$

Best-case and Worst-case Analysis

Best-case: $1 \quad 2 \cdots n$

$$
\begin{aligned}
|C| &= \big(\, \overbrace{\min : n-1,}^{\substack{\text{Best-case:} \\ \text{non-decreasingly sorted}}} \qquad \overbrace{\qquad\qquad\qquad}^{\substack{\text{Worst-case:} \\ \text{non-increasingly sorted}}} \big); \\
|S| &= \big(\, \min : 0, \qquad\qquad\qquad\qquad \big).
\end{aligned}
$$

Worst-case: $n \quad n-1 \cdots 1$

Best-case and Worst-case Analysis

Best-case: $1 \quad 2 \cdots n$

$$
\begin{aligned}
|C| &= (\underbrace{\min : n-1,}_{\substack{\text{Best-case:} \\ \text{non-decreasingly sorted}}} \quad \underbrace{\max : \tfrac{n^2-n}{2}}_{\substack{\text{Worst-case:} \\ \text{non-increasingly sorted}}});\\
|S| &= (\min : 0, \quad \max : \tfrac{n^2-n}{2}).
\end{aligned}
$$

Worst-case: $n \quad n-1 \cdots 1$

$|S|$: #Swaps (Average-case Analysis)[3]

Assumptions on inputs:

1. The input is a random permutation ("average input")

[3] An exercise: what is $|C|$ (#Comparisions) in average?

$|S|$: #Swaps (Average-case Analysis)[3]

Assumptions on inputs:

1. The input is a random permutation ("average input")
2. All numbers are different (for simplicity)

[3] An exercise: what is $|C|$ (#Comparisions) in average?

$|S|$: #Swaps (Average-case Analysis)[3]

Assumptions on inputs:

1. The input is a random permutation ("average input")
2. All numbers are different (for simplicity)

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

$$|S| = \mathbb{E}(\#\text{inversions})$$

[3] An exercise: what is $|C|$ (#Comparisions) in average?

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

$$I_{ij} = \begin{cases} 1 & (a_i, a_j) \text{ is an inversion} \\ 0 & \text{o.w.} \end{cases}$$

$$X = \sum_{1 \le i < n} \sum_{i < j \le n} I_{ij} \qquad (\#\text{inversions})$$

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

$$I_{ij} = \begin{cases} 1 & (a_i, a_j) \text{ is an inversion} \\ 0 & \text{o.w.} \end{cases}$$

$$X = \sum_{1 \le i < n} \sum_{i < j \le n} I_{ij} \qquad (\#\text{inversions})$$

$$\mathbb{E}(X) = \mathbb{E}(\sum_{i} \sum_{j > i} I_{ij})$$

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

$$I_{ij} = \begin{cases} 1 & (a_i, a_j) \text{ is an inversion} \\ 0 & \text{o.w.} \end{cases}$$

$$X = \sum_{1 \le i < n} \sum_{i < j \le n} I_{ij} \qquad (\#\text{inversions})$$

$$\mathbb{E}(X) = \mathbb{E}(\sum_{i} \sum_{j>i} I_{ij}) = \sum_{i} \sum_{j>i} \mathbb{E}(I_{ij}) \quad (\text{linearity of expectation})$$

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

$$I_{ij} = \begin{cases} 1 & (a_i, a_j) \text{ is an inversion} \\ 0 & \text{o.w.} \end{cases}$$

$$X = \sum_{1 \le i < n} \sum_{i < j \le n} I_{ij} \qquad (\#\text{inversions})$$

$$\mathbb{E}(X) = \mathbb{E}(\sum_i \sum_{j>i} I_{ij}) = \sum_i \sum_{j>i} \mathbb{E}(I_{ij}) \quad (\text{linearity of expectation})$$

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

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

$$I_{ij} = \begin{cases} 1 & (a_i, a_j) \text{ is an inversion} \\ 0 & \text{o.w.} \end{cases}$$

$$X = \sum_{1 \le i < n} \sum_{i < j \le n} I_{ij} \qquad (\#\text{inversions})$$

$$\mathbb{E}(X) = \mathbb{E}(\sum_i \sum_{j>i} I_{ij}) = \sum_i \sum_{j>i} \mathbb{E}(I_{ij}) \quad (\text{linearity of expectation})$$

$$\mathbb{E}(I_{ij}) = \mathbb{P}\{I_{ij} = 1\} = \frac{1}{2} \qquad (a_i \neq a_j; \text{half: } a_i < a_j, \text{half: } a_i > a_j)$$

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

$$I_{ij} = \begin{cases} 1 & (a_i, a_j) \text{ is an inversion} \\ 0 & \text{o.w.} \end{cases}$$

$$X = \sum_{1 \le i < n} \sum_{i < j \le n} I_{ij} \qquad (\#\text{inversions})$$

$$\mathbb{E}(X) = \mathbb{E}(\sum_i \sum_{j>i} I_{ij}) = \sum_i \sum_{j>i} \mathbb{E}(I_{ij}) \quad (\text{linearity of expectation})$$

$$\mathbb{E}(I_{ij}) = \mathbb{P}\{I_{ij} = 1\} = \frac{1}{2} \qquad (a_i \neq a_j; \text{half: } a_i < a_j, \text{half: } a_i > a_j)$$

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

Faster 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 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*

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

Faster 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*

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

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

Faster 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*

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

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

hengxin0912@gmail.com

4 Appendix

Bubble Sort: Correctness

Finiteness

Bubble Sort: Correctness

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

Bubble Sort: Correctness

$$\textbf{Finiteness} \implies \exists \text{ loop} : \text{ no swaps}$$
$$\implies A \text{ has no adjacent inversions any more}$$

Bubble Sort: Correctness

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

Bubble Sort: Correctness

procedure BUBBLESORT($A : a_1\ a_2\ \cdots\ a_n$)
  repeat
    swapped $\leftarrow$ false
    for $i \leftarrow 1 : n-1$ **do** ▷ **Loop invariant?**
      if $a_i > a_{i+1}$ **then** ▷ CAS
        SWAP(a_i, a_{i+1})
        swapped $\leftarrow$ 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

Bubble Sort: Finiteness

Idea: well-founded relation over $\mathbf{N}$

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

5 3 1 6 7 2 4 8

5 3 1 6 2 7 4 8

Bubble Sort: Finiteness

Idea: well-founded relation over $\mathbf{N}$

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

Idea: well-founded relation over $\mathbf{N}$

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

5 3 1 6 7 2 4 8

5 3 1 6 2 7 4 8

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

Optimizing Bubble Sort (II)

Idea: After each "**for**" loop, all elements after "lsp" are settled.

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 (II)

Idea: After each "**for**" loop, all elements after "lsp" are settled.

```
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 (II)

Idea: After each "**for**" loop, all elements after "lsp" are settled.

procedure BubbleSort($A : a_1\ a_2\ \cdots\ a_n$)
    repeat
        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})
                lsp $\leftarrow$ i ▷ Update lsp
        $n \leftarrow$ lsp ▷ Elements after lsp are sorted
    until lsp $= 0$