Bubble Sort

(What are Algorithms and How to Analyze Algorithms)

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 16, 2016

Bubble Sort

Bubble Sort

Sorting

The sorting problem:

Given a sequence A of integers,
arrange them in ascending/descending order.

Sorting

The sorting problem:

Given a sequence A of integers,
arrange them in ascending/descending order.

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

Sorting

The sorting problem:

Given a sequence A of sortable elements,
arrange them in ascending/descending order.

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

A little more formalism: ordering relation "$<$" on A.

$\forall a, b, c \in A$,

Trichotomy: $a < b, a = b, a > b$

Transitivity: $a < b \wedge b < c \implies a < c$

Sorting Algorithms

Five important features of (sorting) algorithms:

Input: an integer array A

Ouput: A' sorted

Definiteness: precisely defined

Sorting Algorithms

Five important features of (sorting) algorithms:

Input: an integer array A

Ouput: A' sorted

Definiteness: precisely defined

Finiteness: termination

Sorting Algorithms

Five important features of (sorting) algorithms:

- **Input:** an integer array A
- **Ouput:** A' sorted
- **Definiteness:** precisely defined
- **Finiteness:** termination
- **Effectiveness:** operations are sufficiently basic
  CAS: compare and swap if out-of-order

Sorting Algorithms

Five important features of (sorting) algorithms:

- **Input:** an integer array A
- **Ouput:** A' sorted
- **Definiteness:** precisely defined
- **Finiteness:** termination
- **Effectiveness:** operations are sufficiently basic
  CAS: compare and swap if out-of-order

Roughly, what is computation?

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 is sorted $\iff$ A has *no* adjacent inversions.

Inversions

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

Proof.

$\Longrightarrow$: No inversions at all.

Inversions

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

Proof.

$\Longrightarrow$: No inversions at all.

$\Longleftarrow$: $\forall i \in [1, n-1] : a_i \le a_{i+1}$.

□

Bubble Sort

Bubble Sort

Basic idea: to eliminate all adjacent inversions

```
repeat
  compare adjacent elements and swap if out-of-order
until no adjacent inversion
```

Bubble Sort

Basic idea: to eliminate all adjacent inversions

```
  repeat
    compare adjacent elements and swap if out-of-order
  until no adjacent inversion
```

Definiteness!

```
repeat
  swapped = false
  for i = 1 to n-1
    if a_{i-1} > a_i
      swap(a_{i-1}, a_i)
      swapped = true
until not swapped
```

Bubble Sort

```
repeat
  swapped = false
  for i = 1 to n-1
    if $a_{i-1} > a_i$
      swap($a_{i-1}$, $a_i$)
      swapped = true
until not swapped
```

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

Bubble Sort

```
repeat
  swapped = false
  for i = 1 to n-1
    if a_{i-1} > a_i
      swap(a_{i-1}, a_i)
      swapped = true
until not swapped
```

Finiteness!

Proof.

The inner "for" loop:

I. not swapped $\implies$ terminates

II. counting #inversions

- swap$(a_{i-1}, a_i) \implies -1$ inversion; $+0$ inversion
- total #inversions is finite

Optimizing Bubble Sort

```
n ← len(A)
repeat
  swapped ← false
  for i = 1 to n-1
    if a_{i-1} > a_i
      swap(a_{i-1}, a_i)
      swapped ← true
  n ← n - 1
until not swapped
```

Optimizing Bubble Sort

```
n ← len(A)
repeat
  swapped ← false
  lsp ← 0
  for i = 1 to n-1
    if a_{i-1} > a_i
      swap(a_{i-1}, a_i)
      swapped ← true
      lsp ← i
  n ← lsp
until not swapped
```

Bubble Sort

Time Complexity of Bubble Sort

Finiteness

Time Complexity of Bubble Sort

Finiteness

$\lvert P \rvert$:	#Passes	(the "for" loops)
$\lvert C \rvert$:	#Comparisons	(`if` $a_{i-1} > a_i$)
$\lvert S \rvert$:	#Swaps	(`swap`(a_{i-1}, a_i))

Time Complexity of Bubble Sort

Finiteness

$\lvert P \rvert$:	#Passes	(the "for" loops)
$\lvert C \rvert$:	#Comparisons	($\texttt{if}\ a_{i-1} > a_i$)
$\lvert S \rvert$:	#Swaps	($\texttt{swap}(a_{i-1}, a_i)$)

Different inputs $\Longleftarrow$ different execution time:

Best-case, Worst-case, and Average-case Analysis

Best-case and Worst-case Analysis

Best-case: Worst-case:

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

Best-case and Worst-case Analysis

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

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

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

Best-case and Worst-case Analysis

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

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

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

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

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

$|S|$: #Swaps