Bubble Sort

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

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 18, 2016

Bubble Sort

1. Sorting
2. Bubble Sort
3. Analysis

Bubble Sort

Sorting

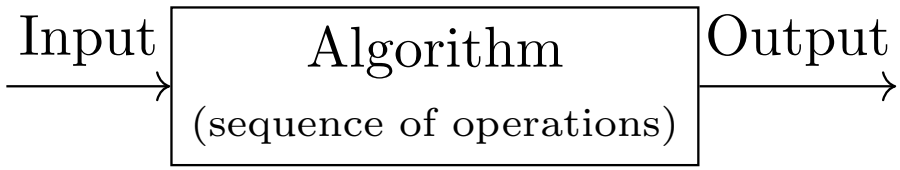
The sorting problem:

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

Output: A sorted (non-decreasing order).

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

Algorithms



Correctness!

Definiteness: precisely defined operations

Finiteness: termination

Effectiveness: a reasonable model; basic operations

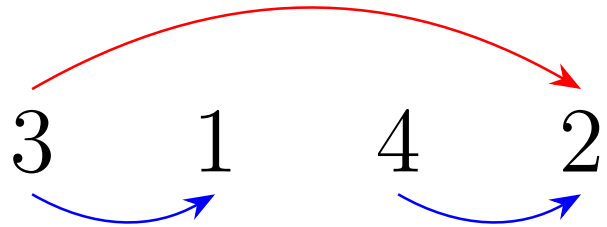
- unrealistic: `sort` operation
- realistic: arithmetic, data movement, and control
- CAS[1] for sort: compare and swap if out-of-order

[1] Forget about that CAS in computer architecture.

Inversions

$$A = a_1 \quad a_2 \quad \dots \quad a_n.$$

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



Adjacent inversion: $j = i + 1$

Inversions

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

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

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

Bubble Sort

Bubble Sort

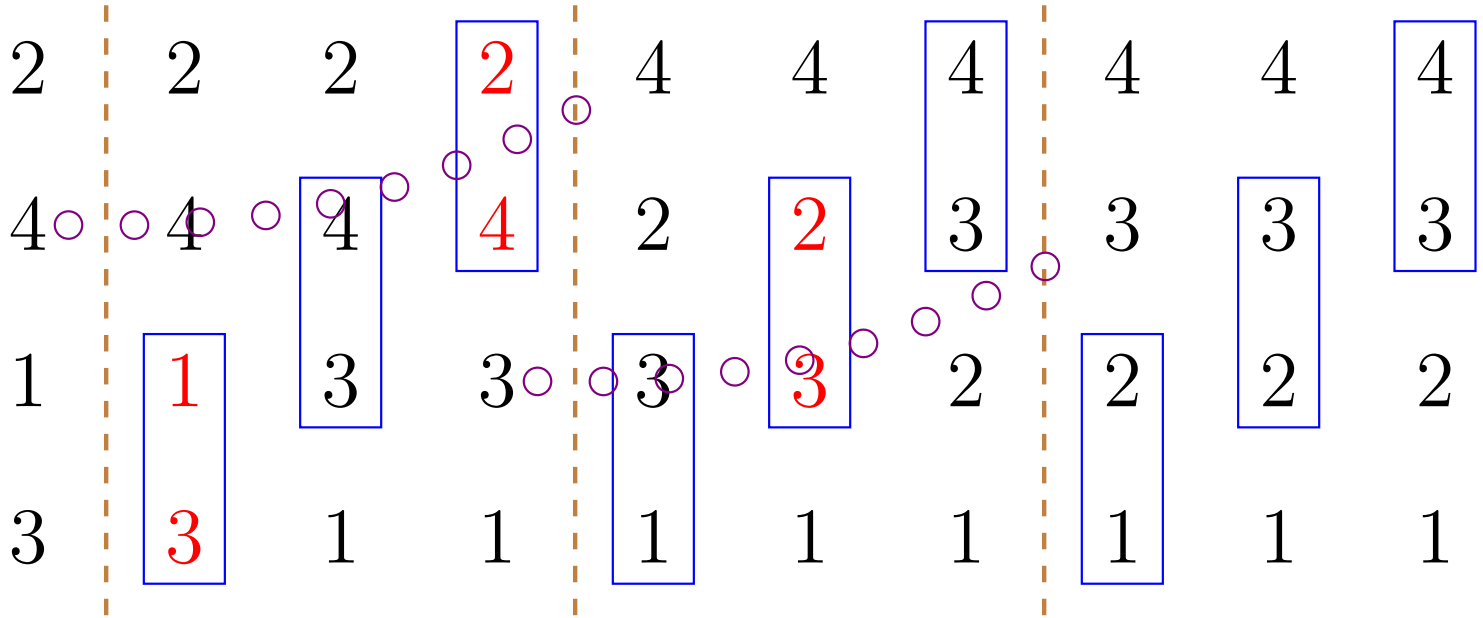
Basic idea: to eliminate all adjacent inversions

1: **procedure** BUBBLESORT($A : a_1\ a_2\ \cdots\ a_n$)
2: **repeat**
3: pick any i ▷ **Definiteness!**
4: **if** $a_i > a_{i+1}$ **then** ▷ CAS
5: SWAP(a_i, a_{i+1})
6: **until** no adjacent inversions ▷ **Finiteness! Definiteness!**

Bubble Sort: Definiteness

1: **procedure** BUBBLESORT($A : a_1\ a_2\ \cdots\ a_n$)
2: **repeat**
3: swapped $\leftarrow$ false
4: **for** $i \leftarrow 1 : n-1$ **do** ▷ Pick i
5: **if** $a_i > a_{i+1}$ **then**
6: SWAP(a_i, a_{i+1})
7: swapped $\leftarrow$ true
8: **until** no adjacent inversionsswapped = false

Bubble Sort: Example



Bubble Sort: Finiteness

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

The inner "**for**" loops:

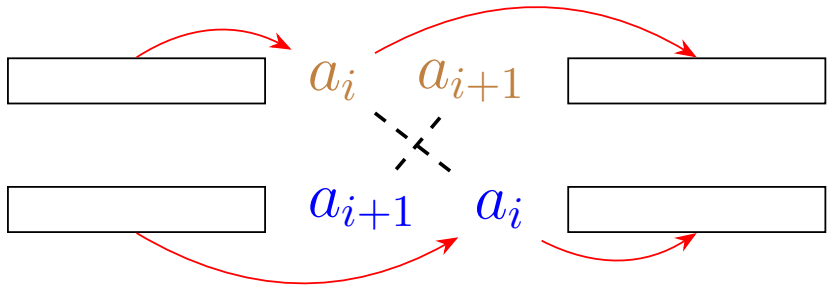
1) $\exists$ loop : no swaps $\Longrightarrow$ swapped = false $\Longrightarrow$ terminates

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

Bubble Sort: Finiteness

Total #inversions is finite.

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



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

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

Bubble Sort: Correctness

Finiteness $\Longrightarrow$ $\exists$ loop : no swaps

$\Longrightarrow$ A has no adjacent inversions any more

$\Longrightarrow$ A is already sorted.

Optimizing Bubble Sort (I)[2]

Idea: After each "**for**" loop, one more element is settled.

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

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

Bubble Sort

Time Complexity of Bubble Sort

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

  $|C|$: #Comparisons (**if** $a_i > a_{i+1}$)

  $|S|$: #Swaps ($\textsc{Swap}(a_i, a_{i+1})$)
- Different inputs $\Longrightarrow$ $|C|$ and $|S|$ vary:
  - Best-case, worst-case, and average-case analysis

Best-case and Worst-case Analysis

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

	Best-case: ascendingly sorted	Worst-case: descendingly sorted
$\lvert C\rvert$	$= (\min: n-1,$	$\max: \frac{n^2-n}{2});$
$\lvert S\rvert$	$= (\min: 0,$	$\max: \frac{n^2-n}{2}).$

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

$$\#\text{inversions} = (n-1) + (n-2) + \cdots + 1 = \frac{n^2-n}{2}.$$

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

Assumptions on inputs:

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

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

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

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

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

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

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

$$\mathbb{E}(I_{ij}) = \mathbb{P}\{I_{ij} = 1\} = \frac{1}{2} \qquad (a_i > a_j \vee a_i < a_j)$$

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

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

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

hengxin0912@gmail.com

Bubble Sort

4. Appendix

Bubble Sort: Correctness

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

Loop invariant:

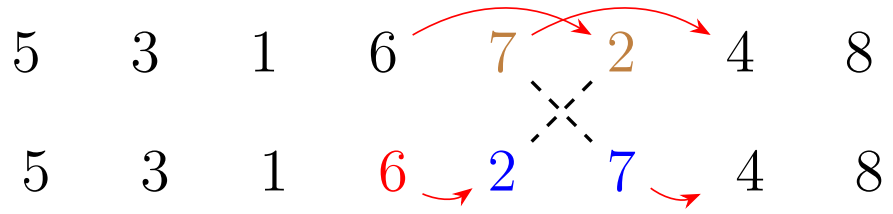
Before the k-th ($k \geq 1$) "**for**" loop, $a_{n-(k-1)} \cdots a_n$
(1) consists of the largest $(k-1)$ elements
(2) in sorted order.

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:



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

Optimizing Bubble Sort (II)

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

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