Bubble Sort

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

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 19, 2016

Bubble Sort

Algorithms

What is an algorithm? What is computation?

Input $\longrightarrow$ Algorithm (sequence of operations) $\longrightarrow$ 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.* $a'_1 \leq a'_2 \leq \cdots \leq a'_n$ (non-decreasing order).

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

$$\xrightarrow{A} \boxed{\begin{array}{c}\text{Sorting Algorithm} \\ \text{(compare, swap)}\end{array}} \xrightarrow{A'}$$

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

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

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

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

Bubble Sort: 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!
```

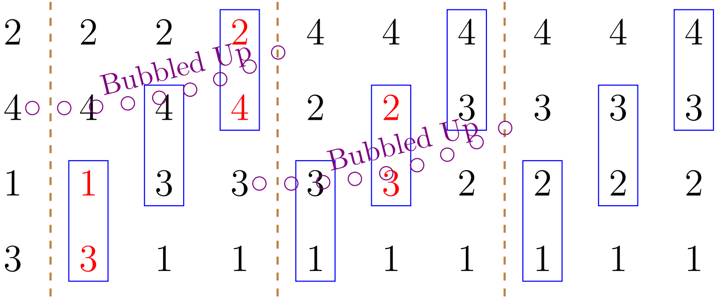
Finiteness $\implies$ Correctness.

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

Bubble Sort: Definiteness

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

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
        $n \leftarrow n - 1$ ▷ 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 ··· 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:

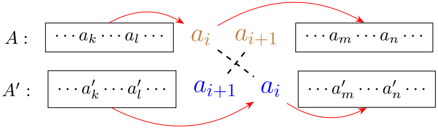
1) $\exists$ loop : no swaps $\implies$ swapped = false $\implies$ terminates

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

Bubble Sort: Finiteness

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!

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

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

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

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: $1 \quad 2 \cdots n$

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

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")
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)

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

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

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

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

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

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*

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

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

hengxin0912@gmail.com

4. Appendix

Bubble Sort: Correctness

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

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
```

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:

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

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

Optimizing Bubble Sort (II)

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

```
procedure BubbleSort(A : a_1 a_2 ··· 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 = falselsp = 0
```