Sorting Algorithms & Run-Time Complexity

L. Hinrichs

Nebraska Wesleyan Univ.

April 9, 2015

Introduction

Algorithms

- Combinatorics
- Sorting Algorithms

Bubble sort algorithm

- Process
- Run-Time Complexity

Merge sort algorithm

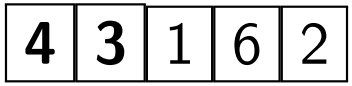
- Process
- Run-Time Complexity

Algorithms

- **Algorithm** - a finite sequence of unambiguously defined steps that carries out a task
- **Sorting Algorithm** - arrange certain objects from an array [by size, for our purposes]
- Assume numbers are unique positive integers

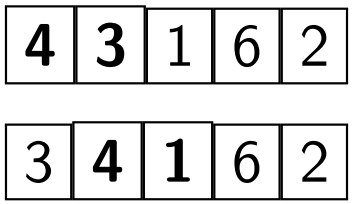
Bubble sort algorithm

$n = 5$



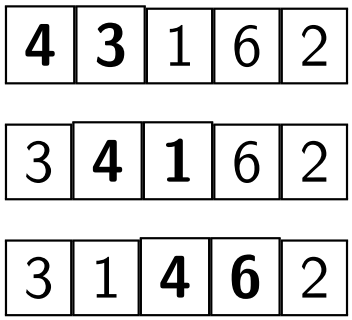
Bubble sort algorithm

$n = 5$



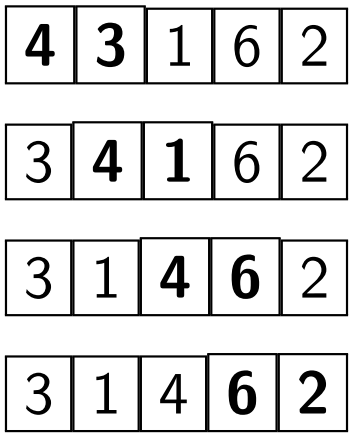
Bubble sort algorithm

$n = 5$



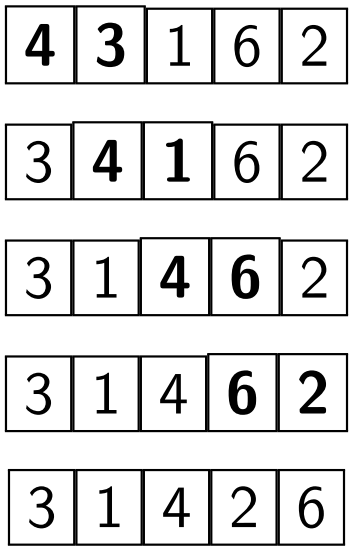
Bubble sort algorithm

$n = 5$



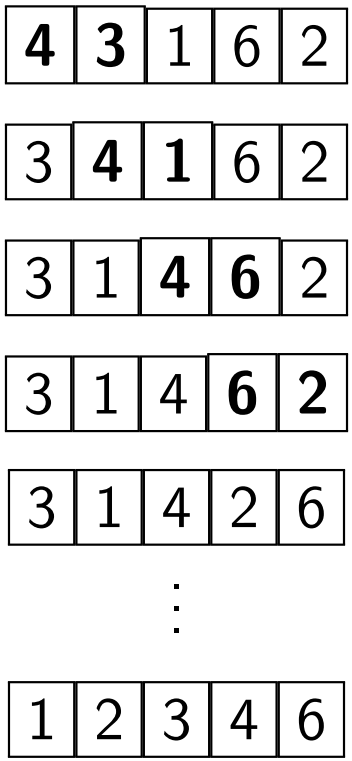
Bubble sort algorithm

$n = 5$



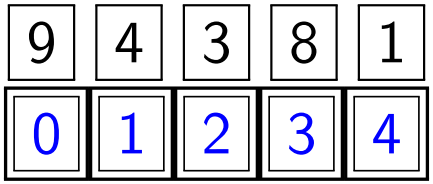
Bubble sort algorithm

$n = 5$



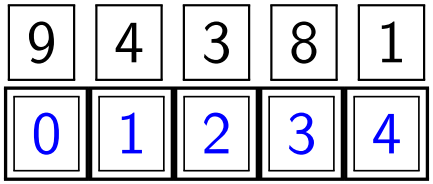
Bubble sort algorithm

Indexed Array A



Bubble sort algorithm

Indexed Array A



Pseudocode Single-Pass BubbleSort(A,n)

```
for i from 0 to n-2

    if (A[i] > A[i + 1])

        swap(A[i],A[i+1])
```

Big O Notation

- "Landau's symbol"
- $\mathcal{O}$ describes a **"simplified"** rate of growth or behavior of a function
- If run-time is a polynomial, $\mathcal{O}$ is the order
- **Ex:**
  $f(n) = 3n^2 + 9n + 2 \implies f = \mathcal{O}(n^2)$

Bubble sort algorithm

Pseudocode BubbleSort(A,n)

```
for k from 0 to n - 1

    set fullsort = 0

    for i from 0 to n - k - 1

        if (A[i] > A[i + 1])

            swap(A[i],A[i+1])

            fullsort = 1

    if fullsort == 0, break
```

Bubble sort time complexity

$$\sum_{k=0}^{n-1} n - k - 1 = (n-1) + (n-2) + \ldots + 0$$

$$= \frac{n(n-1)}{2}$$

$$= \mathcal{O}(n^2)$$

Case Scenarios for Bubble Sort

Best Case:

$$\begin{aligned} n - 1 &\leq n \\ \implies &\mathcal{O}(n) \end{aligned}$$

Worst Case:

$$\begin{aligned} \binom{n}{2} &= \frac{n!}{2!(n-2)!} \\ &= \frac{n(n-1)}{2} \\ &= \frac{n^2 - n}{2} \\ &\leq n^2 \\ \implies &\mathcal{O}(n^2) \end{aligned}$$

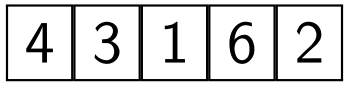
Merge sort algorithm

$n = 5$

4	3	1	6	2

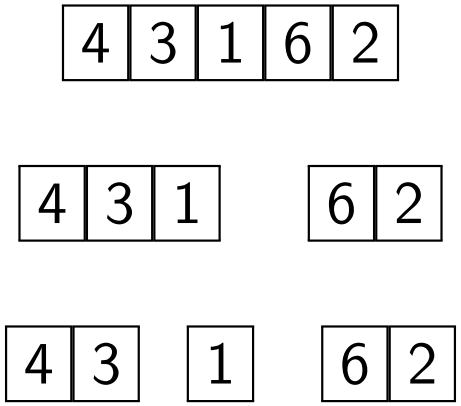
Merge sort algorithm

$n = 5$



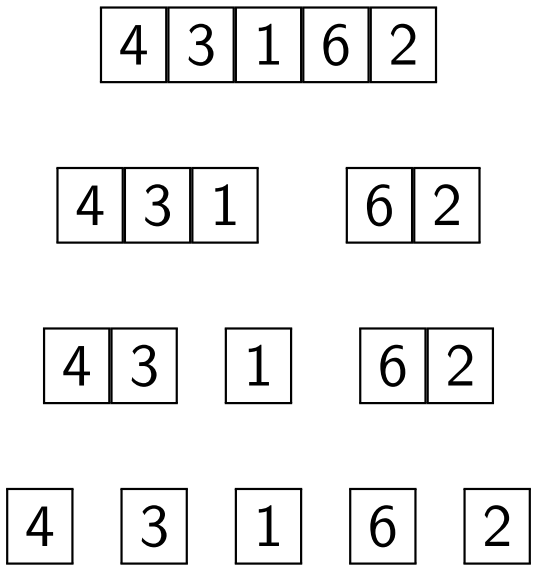
Merge sort algorithm

$n = 5$



Merge sort algorithm

$n = 5$



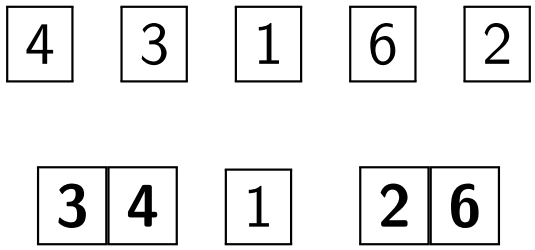
Merge sort algorithm

$$n = 5$$

4	3	1	6	2

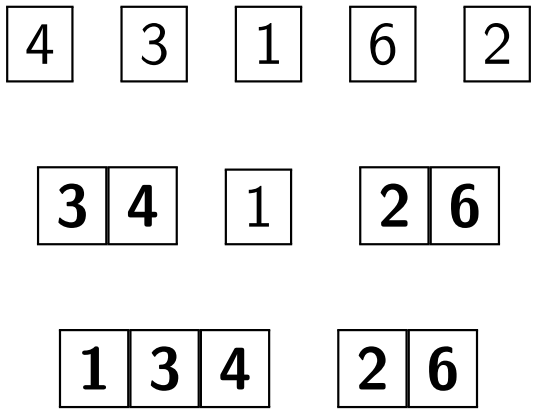
Merge sort algorithm

$$n = 5$$



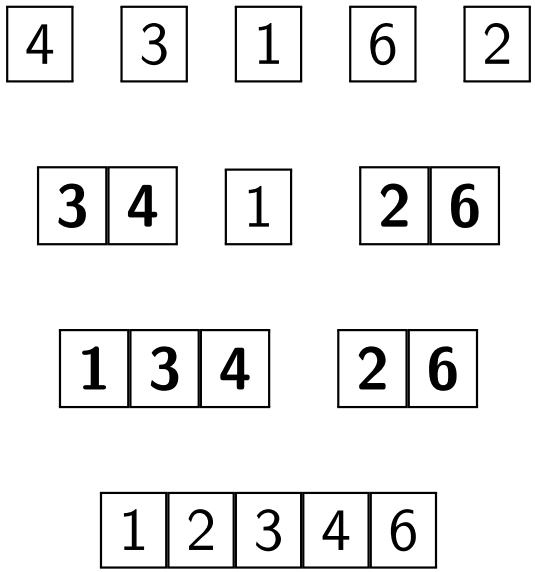
Merge sort algorithm

$n = 5$



Merge sort algorithm

$n = 5$



Merge sort algorithm

Let $M(n)$ be the number of comparisons required to sort a list of size n.

Merge sort algorithm

Let $M(n)$ be the number of comparisons required to sort a list of size n.

Then $M(1) = 0$ and $M(2) = 1$.

Merge sort algorithm

Let $M(n)$ be the number of comparisons required to sort a list of size n.

$$\text{Then } M(1) = 0 \text{ and } M(2) = 1.$$

$$\text{If } n = 2k, \quad M(2k) = 2M(k) + 2k - 1$$

Big O Merge Sort

Let $n = 2^m$.

Big O Merge Sort

Let $n = 2^m$.

$$\begin{aligned} f(m) &= 2f(m-1) + 2^m - 1 \\ &= 2(2f(m-2) + 2^{m-1} - 1) + 2^m - 1 \\ &= 2^2 f(m-2) + 2 \cdot 2^m - 2 - 1 \end{aligned}$$

Big O Merge Sort

Let $n = 2^m$.

$$\begin{aligned} f(m) &= 2f(m-1) + 2^m - 1 \\ &= 2(2f(m-2) + 2^{m-1} - 1) + 2^m - 1 \\ &= 2^2 f(m-2) + 2 \cdot 2^m - 2 - 1 \\ &= 2^3 f(m-3) + 3 \cdot 2^m - 2^2 - 2 - 1 \end{aligned}$$

Big O Merge Sort

Let $n = 2^m$.

$$
\begin{aligned}
f(m) &= 2f(m-1) + 2^m - 1 \\
&= 2(2f(m-2) + 2^{m-1} - 1) + 2^m - 1 \\
&= 2^2 f(m-2) + 2 \cdot 2^m - 2 - 1 \\
&= 2^3 f(m-3) + 3 \cdot 2^m - 2^2 - 2 - 1 \\
&\vdots \\
&= 2^{m-2}(2f(m-m) + 2 - 1) + (m-1)2^m - 2^{m-2} - 2^{m-3} - \ldots - 1 \\
&= m2^m - 2^{m-1} - 2^{m-2} - \ldots - 1 \\
&= m2^m - (2^{m-1} + 2^{m-2} + \ldots + 1) \\
&= m2^m - (2^m - 1) \\
&= (m-1)2^m - 1
\end{aligned}
$$

Big O Merge Sort

Let $n = 2^m$.

$$
\begin{aligned}
f(m) &= 2f(m-1) + 2^m - 1 \\
&= 2(2f(m-2) + 2^{m-1} - 1) + 2^m - 1 \\
&= 2^2 f(m-2) + 2 \cdot 2^m - 2 - 1 \\
&= 2^3 f(m-3) + 3 \cdot 2^m - 2^2 - 2 - 1 \\
&\vdots \\
&= 2^{m-2}(2f(m-m) + 2 - 1) + (m-1)2^m - 2^{m-2} - 2^{m-3} - \ldots - 1 \\
&= m2^m - 2^{m-1} - 2^{m-2} - \ldots - 1 \\
&= m2^m - (2^{m-1} + 2^{m-2} + \ldots + 1) \\
&= m2^m - (2^m - 1) \\
&= (m-1)2^m - 1 \\
&\approx (\log_2 n)n \qquad \text{since } n = 2^m \implies m = \log_2 n \\
\implies & \mathcal{O}(n \log n)
\end{aligned}
$$

References


A Walk Through Combinatorics 3rd Ed., Miklós Bóna, World Scientific 2011


Bubble Sort Algorithm, YouTube, MyCodeSchool, `https://www.youtube.com/watch?v=Jdtq5uKz-w4`


Sorting Algorithm Animations, Website, David R. Martin, `http://www.sorting-algorithms.com/`