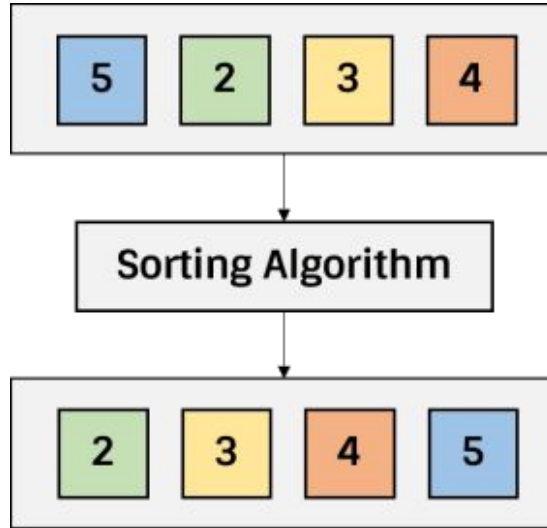


Sorting Algorithms



By: Brendon Lorena,
Ezekiel Velasquez,
Emmanuel Bazile,
& Nathan Campano

What are Sorting Algorithms?

A sorting algorithm is an algorithm that puts elements of a array/list into a specific order. An array or given list of elements are rearranged by the comparison operator on the element.

The comparison operator decides the order of elements in the new list or array given the data structure it is interpreting. There are five common or basic algorithms used, however they each have their own benefits depending on the size and complexity of data being handled.

- Insertion sort
- Merge sort
- Bubble sort
- Selection sort
- Quick sort

Comparison Operators

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Importance of Sorting Algorithms

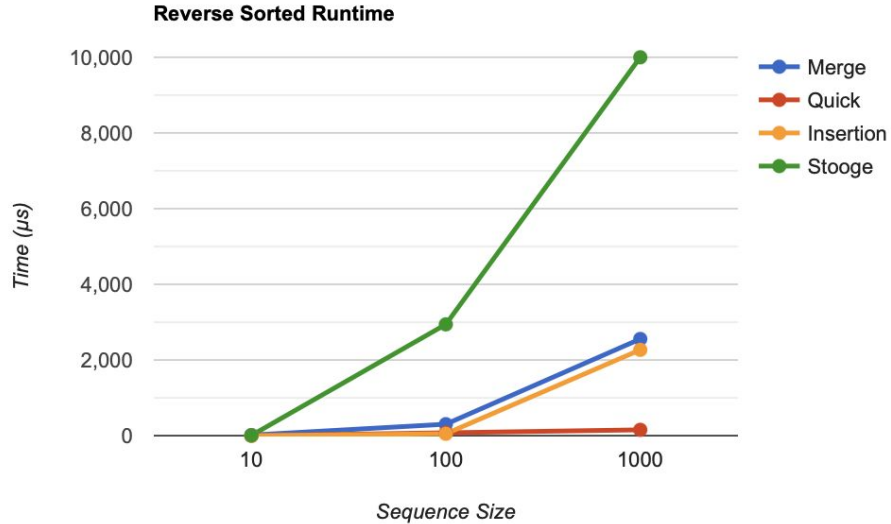
Sorting algorithms are very important in the world of computer science:

- Makes complicated problems simpler and easier to work with.
- Has direct applications to assisting other algorithms such as searching algorithms, and database algorithms.

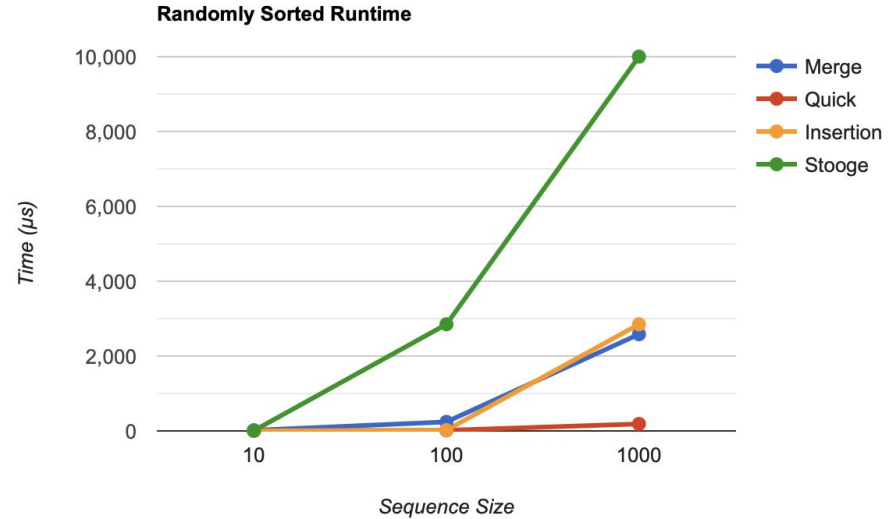
Time and Space Complexities

	<u>Best Case</u>	<u>Average Case</u>	<u>Worst Case</u>	Space Complexity
Mergesort	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	$O(n)$
Quicksort	$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$	$O(\log(n))$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Stooge Sort	$O(n^{\log 3 / \log 1.5})$	$O(n^{\log 3 / \log 1.5})$	$O(n^{\log 3 / \log 1.5})$	$O(n)$

Runtimes

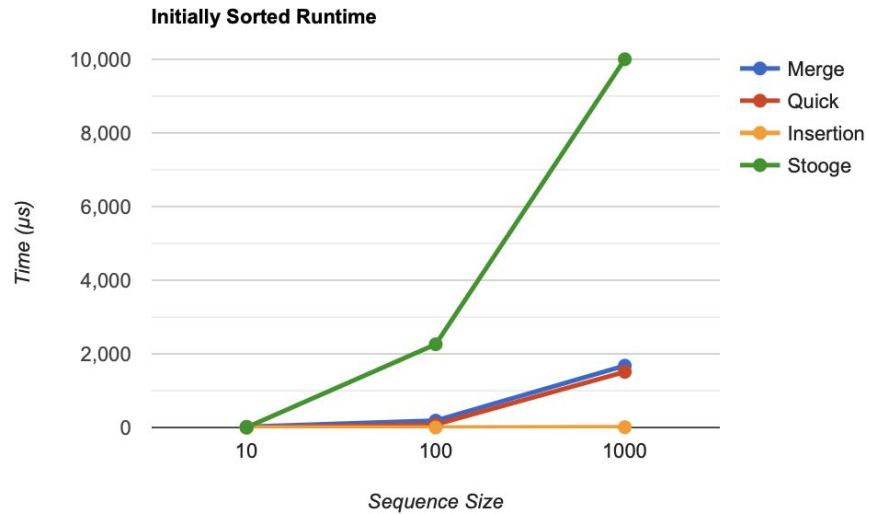


Note: Stooge actually goes to 439652 μ s for sequence of 1000



Note: Stooge actually goes to 442640 μ s for sequence of 1000

Runtimes cont.



*Note: Stooge actually goes to 587776 μs
for sequence size 1000*

Insertion Sort

Initial Array Input:


3	6	4	5	2	1
---	---	---	---	---	---

Step 1:

3	6	4	5	2	1
---	---	---	---	---	---

 \longrightarrow

3	6	4	5	2	1
---	---	---	---	---	---




Step 2:

3	6	4	5	2	1
---	---	---	---	---	---

 \longrightarrow

3	4	6	5	2	1
---	---	---	---	---	---




Step 3:

3	4	6	5	2	1
---	---	---	---	---	---

 \longrightarrow

3	4	5	6	2	1
---	---	---	---	---	---




Step 4:

3	4	5	6	2	1
---	---	---	---	---	---

 \longrightarrow

2	3	4	5	6	1
---	---	---	---	---	---




Step 5:

2	3	4	5	6	1
---	---	---	---	---	---

 \longrightarrow

1	2	3	4	5	6
---	---	---	---	---	---



Sorted Array Output:

1	2	3	4	5	6
---	---	---	---	---	---

Quick Sort

Initial Array Input:

	0	1	2	3	4	5
3	6	4	5	2	1	

Step 1:

3	6	<u>4</u>	5	2	1
---	---	----------	---	---	---

 Randomly Selecting 4 as pivot

Step 2:

3	6	4	5	2	1
---	---	---	---	---	---

 \rightarrow

3	6	1	5	2	<u>4</u>
---	---	---	---	---	----------

Step 3:

3	6	1	5	2	<u>4</u>
---	---	---	---	---	----------

 \rightarrow

3	2	1	5	6	<u>4</u>
---	---	---	---	---	----------

Step 4:

3	2	1	5	6	<u>4</u>
---	---	---	---	---	----------

 \rightarrow

3	2	1	<u>4</u>	6	5
---	---	---	----------	---	---

 $A[2] \leftarrow A[3]$

Step 5:

3	2	1
---	---	---

 \rightarrow

1	2	3
---	---	---

6	5
---	---

 \rightarrow

5	6
---	---

=

1	2	3	<u>4</u>	5	6
---	---	---	----------	---	---

 ✓

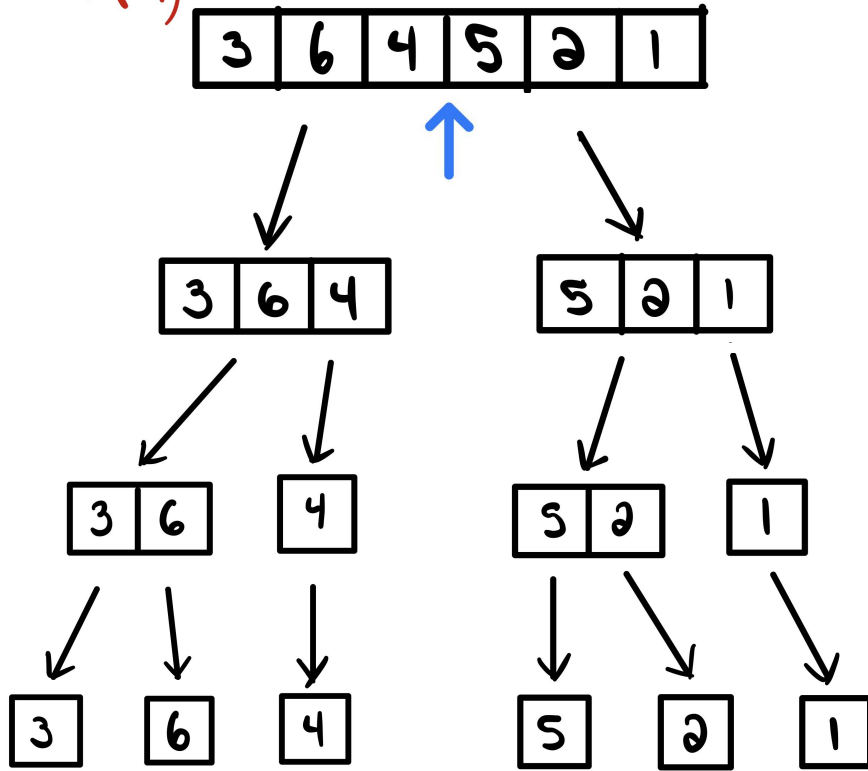
Sorted Array Output:

1	2	3	4	5	6
---	---	---	---	---	---

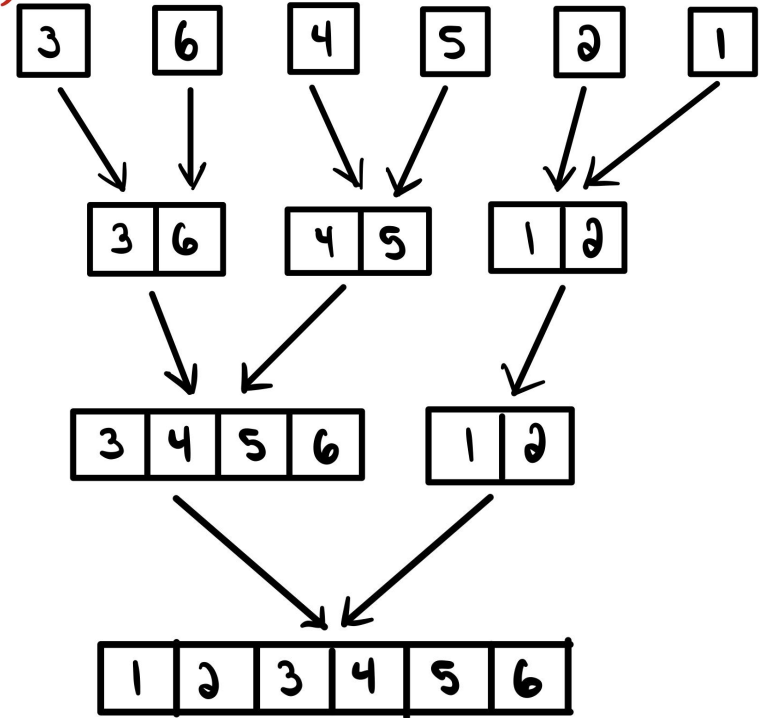
Merge Sort

Step 1)

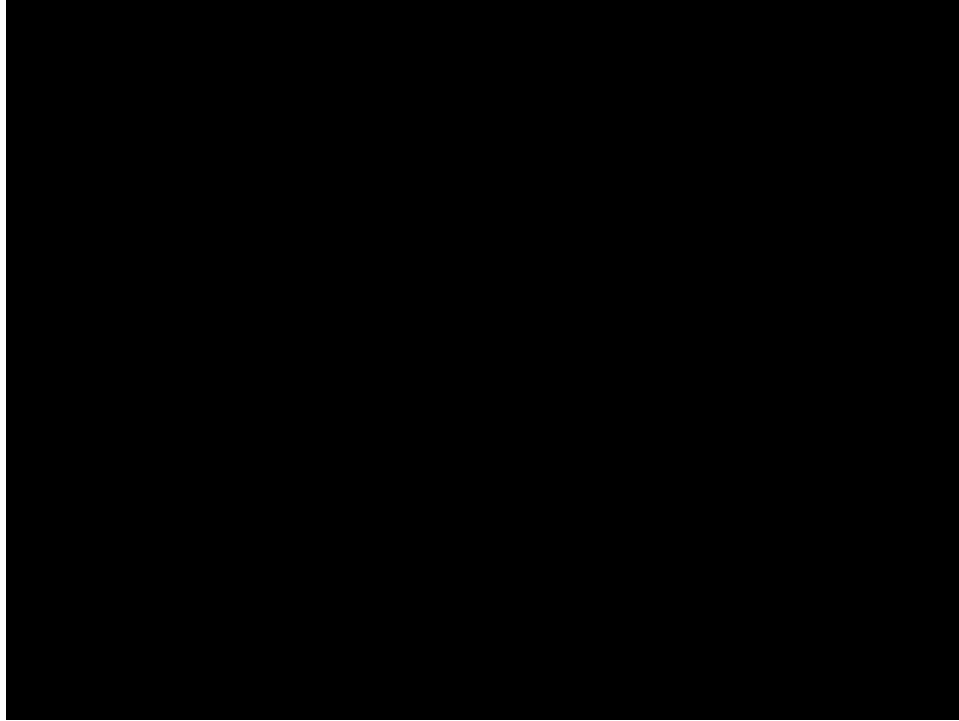
Median



Step 2)



Stooge Sort



Stooge Sort

Initial Array Input:

3	6	4	5	2	1
---	---	---	---	---	---

Step 1

3	6	4	5	2	1
---	---	---	---	---	---

 →

1	6	4	5	2	3
---	---	---	---	---	---

Step 2

a

1	6	4	5	2	3
---	---	---	---	---	---

 →

1	6	4	5
---	---	---	---

 →

1	4	5	6
---	---	---	---

b

1	4	5	6	2	3
---	---	---	---	---	---

 →

5	6	2	3
---	---	---	---

 →

2	3	5	6
---	---	---	---

c

1	4	2	3	5	6
---	---	---	---	---	---

 →

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

 →

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

Sorted Array Output:

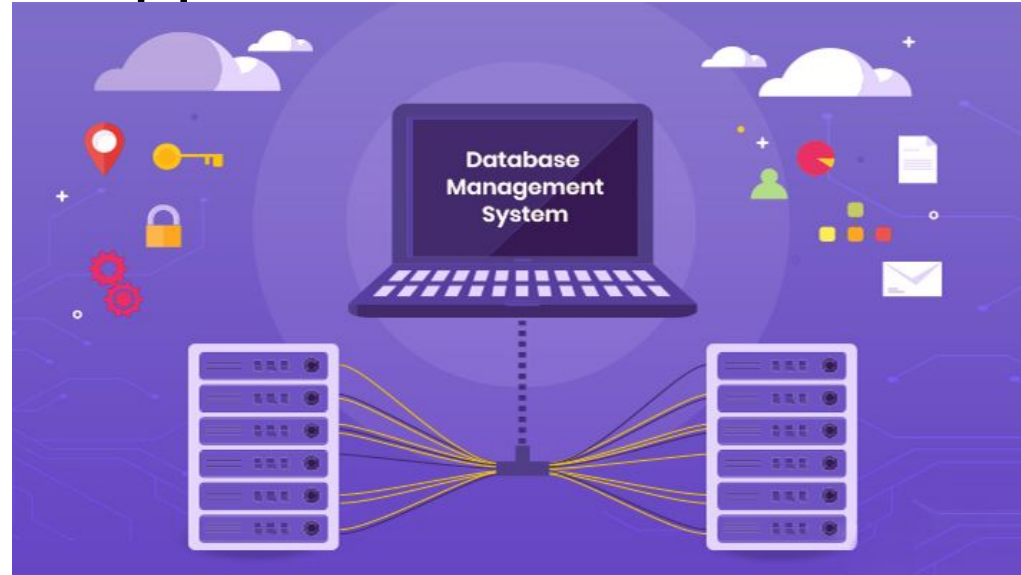
1	2	3	4	5	6
---	---	---	---	---	---

Real-World Applications

Sorting Is Everywhere !!

 PHI 21	 WSH
 NYG 27	 CIN
3RD - 9:24 1ST & 10	4TH - 1:07 🔥

• 22	 KC TOUCH DOWN 10
36	 IND 0
1ST & 10	2ND - 12:58 XP



Coding Demonstration