

Sorting II - Advanced

Unsorted Array

9	1	3	2	7	4
---	---	---	---	---	---



sorting algorithm

Sorted Array

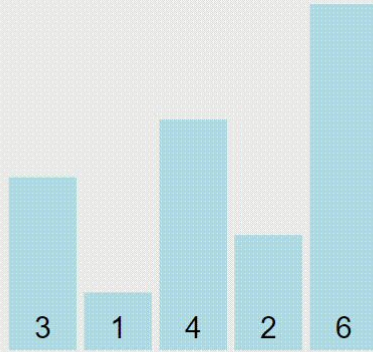
1	2	3	4	7	9
---	---	---	---	---	---

Part II

Quick Sort



Quick Sort



Similar to merge sort, Quicksort follows the **divide-and-conquer approach** that was first introduced.

visualization from: [VisuAlgo](#)

Quick Sort

if **length of array is less than or equal to 1:**

 return array

else:

select an element from the array to use as a pivot

partition the elements of the array into two sub-arrays:

- elements less than or equal to pivot
- elements greater than pivot

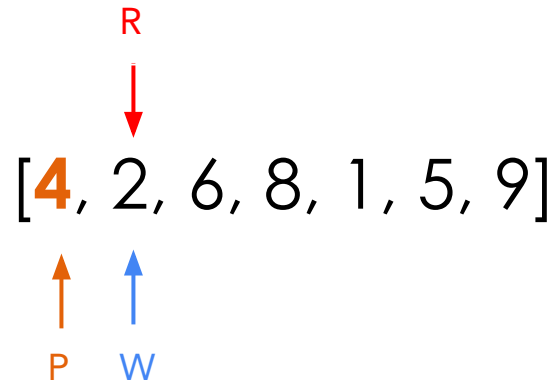
quicksort the sub-array of elements less than or equal to pivot

quicksort the sub-array of elements greater than pivot

concatenate the sorted sub-arrays and return the result

Quick Sort

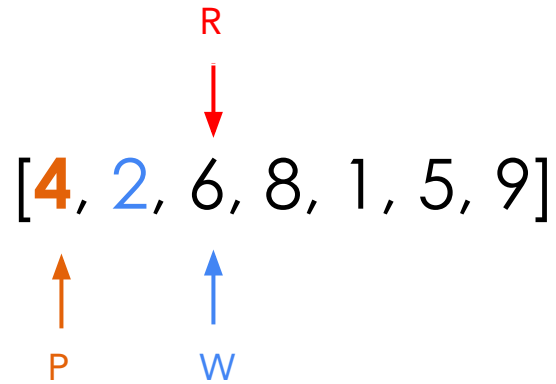
Example input



P: pivot
W: write index
R: read index

Quick Sort

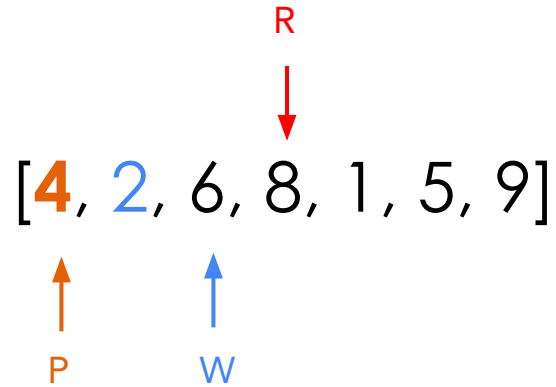
Example input



P: pivot
W: write index
R: read index

Quick Sort

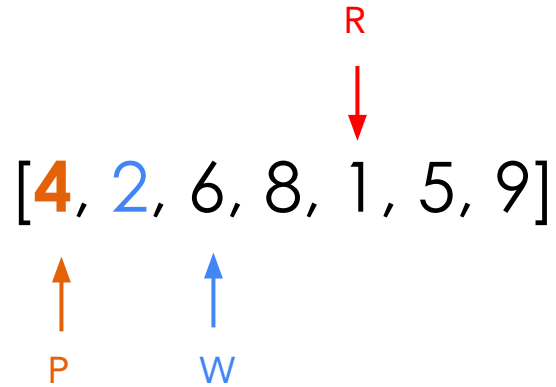
Example input



P: pivot
W: write index
R: read index

Quick Sort

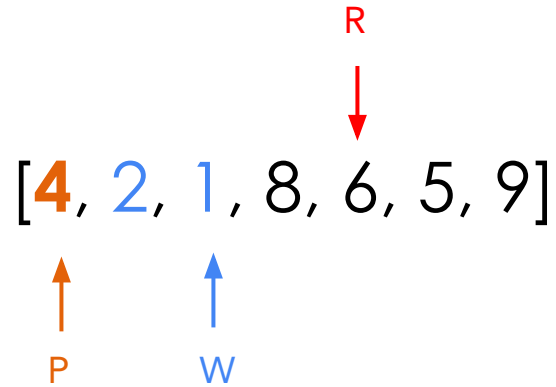
Example input



P: pivot
W: write index
R: read index

Quick Sort

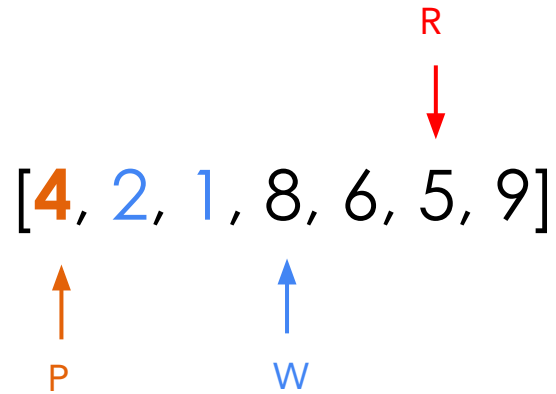
Example input



P: pivot
W: write index
R: read index

Quick Sort

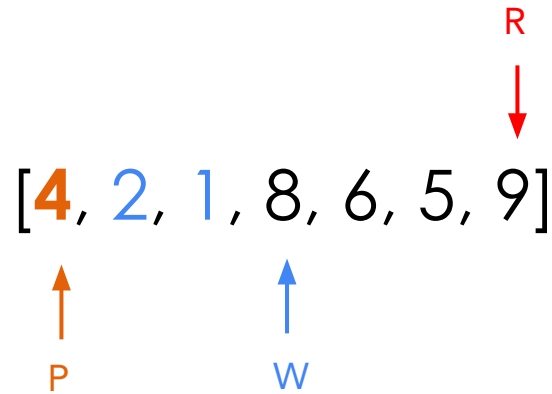
Example input



P: pivot
W: write index
R: read index

Quick Sort

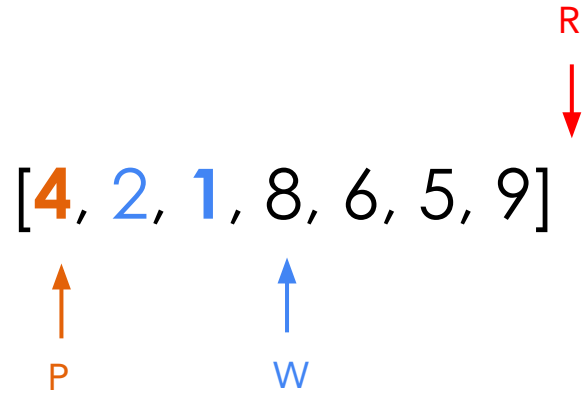
Example input



P: pivot
W: write index
R: read index

Quick Sort

Example input



P: pivot
W: write index
R: read index

Quick Sort

Example input

[1, 2, 4, 8, 6, 5, 9]



P



W

P: pivot
W: write index
R: read index

Quick Sort

Example input

[1, 2, 4, 8, 6, 5, 9]

quicksort([1, 2])

quicksort([8, 6, 5, 9])

Combine the answer

Visualization Link



Can you implement the function **partition** ?

Implement Here

Q: What do you think is the time complexity for the aforementioned sorting Algorithm?



Time & Space Complexity

Worst case ? _____

Best case ? _____

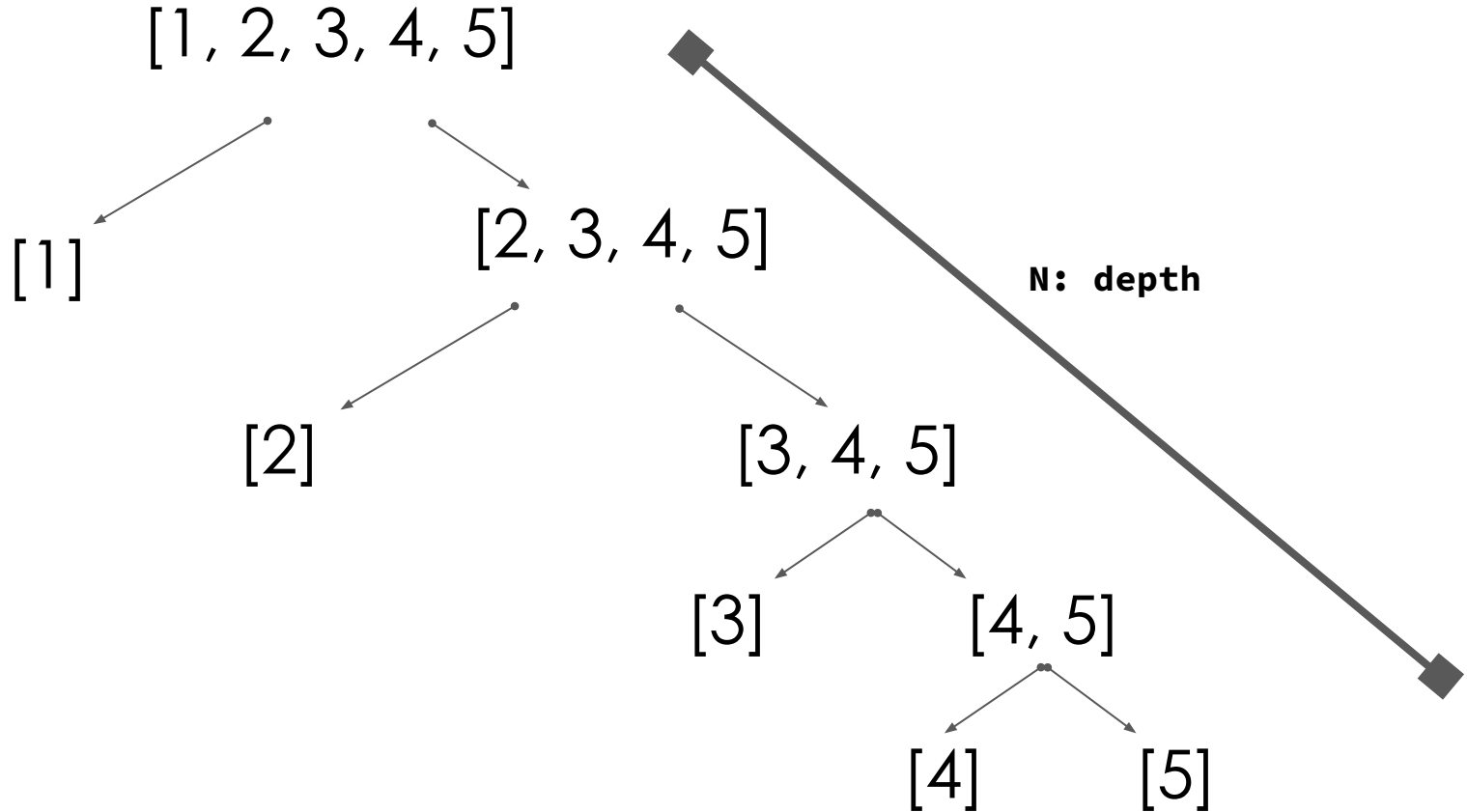
Average case ? _____

Q: what kind of input would result in the **worst case**?



What happens in the worst case?

Sorted
array



Q: Can we do better ? How ?



How can we avoid the worst case?

- Pick pivot randomly
- Median value of `arr[0]`, `arr[len//2]` and `arr[len - 1]`

Time & Space Complexity

Time complexity: **$O(n^2)$**

Space complexity: **$O(1)$**

Worst case **$O(n^2)$**

Best case **$O(n \log n)$**

Average case **$O(n \log n)$**

Stable **NO**

In-place **YES**

Cycle/ Cyclic Sort



Cycle Sort

It is known that all comparison-based sorting algorithms have a lower bound time complexity of $\Omega(N \log N)$.

However, we can achieve faster sorting algorithm — i.e., in $O(N)$ — if certain **assumptions** of the **input array exist**

Cycle Sort

Problem:

You are given an array of **size n** that only includes numbers in the range **[1, n]**, sort the array in a **single pass** in **$O(N)$** runtime.

0	1	2	3	4
[3,	5,	2,	1,	4]

Cycle Sort

Approach:

Let's imagine the array was already sorted, what would be the relationship between the values and the indices ?

0	1	2	3	4
[1	, 2	, 3	, 4	, 5]

Cycle Sort

Approach:

$$\text{Index} = \text{value} - 1$$

0 1 2 3 4
[1, 2, 3, 4, 5]

Cycle Sort

Approach:

This means, we can use the **values** to know where **exactly** in the array they should be **placed**.

0	1	2	3	4
[3,	5,	2,	1,	4]

Where should **3** be placed at ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[3	, 5,	2	, 1,	4]

The value 3 should be placed at index 2. So we swap

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[2,	5,	3,	1,	4]

The value **2** is **not** in the correct place either, **where should it be ?**

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[2,	5,	3,	1,	4]

Swap

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[5,	2,	3,	1,	4]

Where should **5** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[5	2	3	1	4]

Swap

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[4	2	3	1	5]

Where should **4** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[4	2	3	1	5]

Swap

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[1	2	3	4	5]

Where should **1** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[1	2	3	4	5]

|

It is finally in its correct position, so we move our pointer

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[1	, 2	, 3	, 4	, 5]

Where should **2** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[1	, 2	, 3	, 4	, 5]

|

Where should **3** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[1	, 2	, 3	, 4	, 5]

Where should **4** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0	1	2	3	4
[1	2	3	4	5]

Where should **5** be ?

Cycle Sort

This means, we can use the values to know where exactly in the array they should be placed.

0 1 2 3 4
[1, 2, 3, 4, 5]



Array is sorted.

Can you implement the function **cycleSort** ?

Implement Here

Q: What do you think is the time complexity for the aforementioned sorting Algorithm?



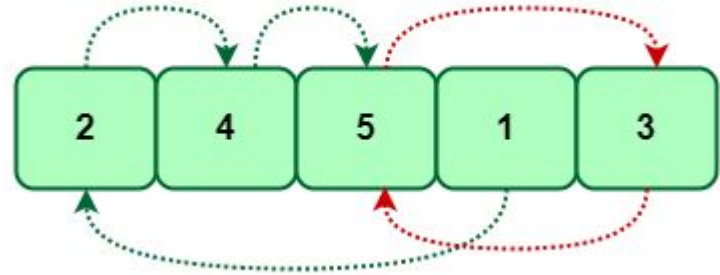
Time & Space Complexity

Cycle Sort

Worst case

Best case

Average case



Time & Space Complexity

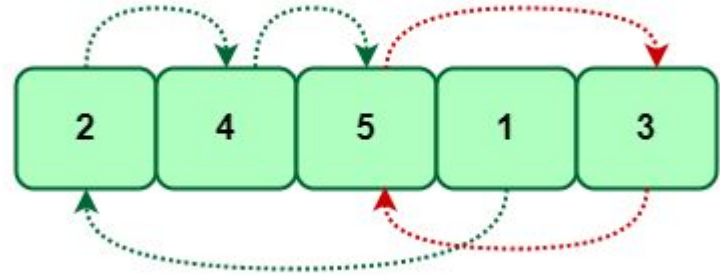
Cycle Sort

Worst case $O(N)$

Best case $O(N)$

Average case $O(N)$

Only applies to a **constrained range**
of values



Further Reading

- There are also other popular sorting algorithms such as **Radix Sort, Binary Insertion Sort...**
- Feel free to explore and share with your teammates.

Pair Programming

Practice Problems

Missing Number

Find All Numbers Disappeared in an Array

Find all duplicates in an array

Set Mismatch

Find the Duplicate Number

First Missing Positive

Kth Largest Element in an Array

Resources

- [Visualgo.com](https://visualgo.com): is great for visualizing sorting algorithms in general
- [Chatgpt](#): is great at re-writing and generating pseudocode
- [Geeks for Geeks \(GFG\)](#): has clear explanations
- [A2SV Slides repo](#): good reference for which topics to cover and good quotes.
- [Leetcode learn card Recursion II](#): good reference for merge sort, quick sort, and other recursion concepts
- [Leetcode learn card Sorting](#): good reference for bucket sort, and other sorting algorithms like radix sort.

Quote of the Day

"The first law of success is concentration — to bend all the energies to one point, and to go directly to that point, looking neither to the right nor to the left."

- **William Matthews**