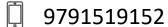# Competitive Programming

In-Place Algorithm

B.Bhuvaneswaran, AP (SG) / CSE

📱 9791519152

✉ bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

In-Place Algorithm

- An in-place function modifies data structures or objects outside of its own stack frame (i.e. stored on the process heap or in the stack frame of a calling function).

- Because of this, the changes made by the function remain after the call completes.

- In-place algorithms are sometimes called destructive, since the original input is "destroyed" (or modified) during the function call.

Note

- "In-place" does *not* mean "without creating any additional variables!"

- Rather, it means "without creating a new copy of the input".

- In *general*, an in-place function will only create additional variables that are *O(1)* space.

Note

- In many languages, primitive values (integers, floating point numbers, or characters) are copied when passed as arguments, and more complex data structures (arrays, heaps, or hash tables) are passed by reference.

- In C, arguments that are pointers can be modified in place.

Advantages

- Working in-place is a good way to save time and space.

- An in-place algorithm avoids the cost of initializing or copying data structures, and it usually has an *O(1)* space cost.

Disadvantages

- But be careful: an in-place algorithm can cause side effects.

- Your input is "destroyed" or "altered," which can affect code *outside* of your function.

Reversing an Array in-place

- The problem is that we want to reverse a T[] array in O(N) linear time complexity and we want the algorithm to be in-place as well!

- For example: input is [1,2,3,4,5] then the output is [5,4,3,2,1]
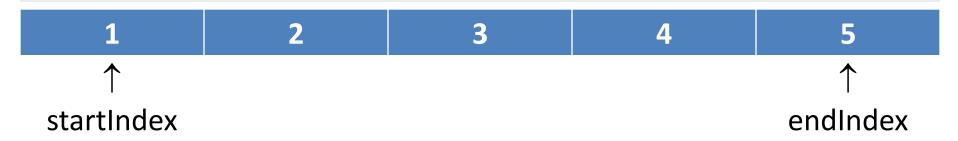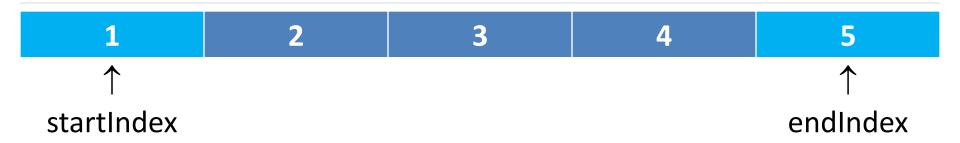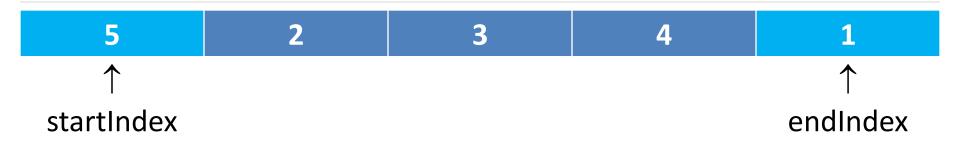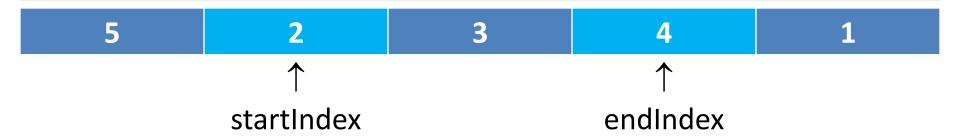
Input

5

1 2 3 4 5

Output

5 4 3 2 1

Example

1	2	3	4	5

5	4	3	2	1

Example

1	2	3	4	5

↑
startIndex

↑
endIndex

Example

1	2	3	4	5

↑ startIndex

↑ endIndex

Example

5	2	3	4	1

↑ startIndex   ↑ endIndex

Example

5	2	3	4	1

↑ startIndex

↑ endIndex

Example

5	4	3	2	1

↑ startIndex   ↑ endIndex

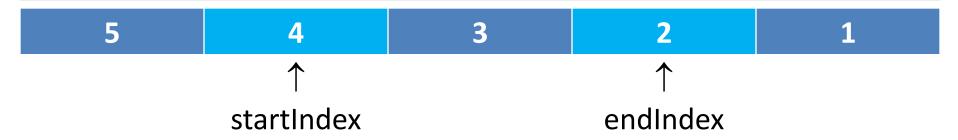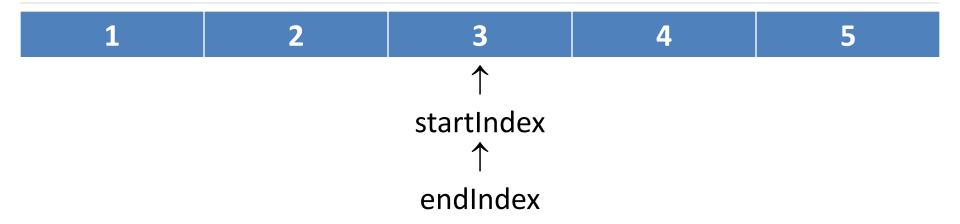# Example

1	2	3	4	5

↑

startIndex

↑

endIndex

Queries?

Thank You...!