In-Place Algorithm

An **in-place** algorithm operates *directly* on its input and *changes* it, instead of creating and returning a *new* object. This is sometimes called **destructive**, since the original input is "destroyed" when it's edited to create the new output.

Careful: "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.

Here are two functions that do the same operation, except one is in-place and the other is out-of-place:

C# (beta)

```
public int[] SquareArrayInPlace(int[] intArray)
{
    for (int i = 0; i < intArray.Length; i++)
    {
        intArray[i] *= intArray[i];
    }

    // NOTE: we don't *need* to return anything
    // This is just a convenience
    return intArray;
}

public int[] SquareArrayOutOfPlace(int[] intArray)
{
    // We allocate a new array with the length of the input array
    var squaredArray = new int[intArray.Length];

    for (int i = 0; i < intArray.Length; i++)
    {
        squaredArray[i] = intArray[i] * intArray[i];
    }

    return squaredArray;
}
```

Working in-place is a good way to save space. An in-place algorithm will generally have $O(1)$ space cost.

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. For example:

C# (beta) ▾

```
var originalArray = new[] { 2, 3, 4, 5 };
var squaredArray = SquareArrayInPlace(originalArray);

// Prints: squared: [4, 9, 16, 25]
Console.WriteLine($"squared: [{string.Join(", ", squaredArray)}]");

// Prints: original array: [4, 9, 16, 25], confusingly!
Console.WriteLine($"original array: [{string.Join(", ", originalArray)}]");

// And if squareArrayInPlace() didn't return anything,
// which it could reasonably do, we just could not initialize and use squaredArray!
```

Generally, out-of-place algorithms are considered safer because they avoid side effects. You should only use an in-place algorithm if you're very space constrained or you're *positive* you don't need the original input anymore, even for debugging.

In-Place Algorithm Coding Interview Questions

24✔ Reverse A Linked List »

Write a function to reverse a linked list in-place. keep reading »

(/question/reverse-linked-list)

26✔ Reverse String in Place »

Write a function to reverse a string in-place. keep reading »

(/question/reverse-string-in-place)

27✔ Reverse Words »

Write a function to reverse the word order of a string, in-place. It's to decipher a supersecret message and win the war. keep reading »

(/question/reverse-words)

40✓ **Find Repeat, Space Edition »**

Figure out which number is repeated. But here's the catch: optimize for space. keep reading »

(/question/find-duplicate-optimize-for-space)

All Questions ➜ (/all-questions)