C# Collections

The Array Type

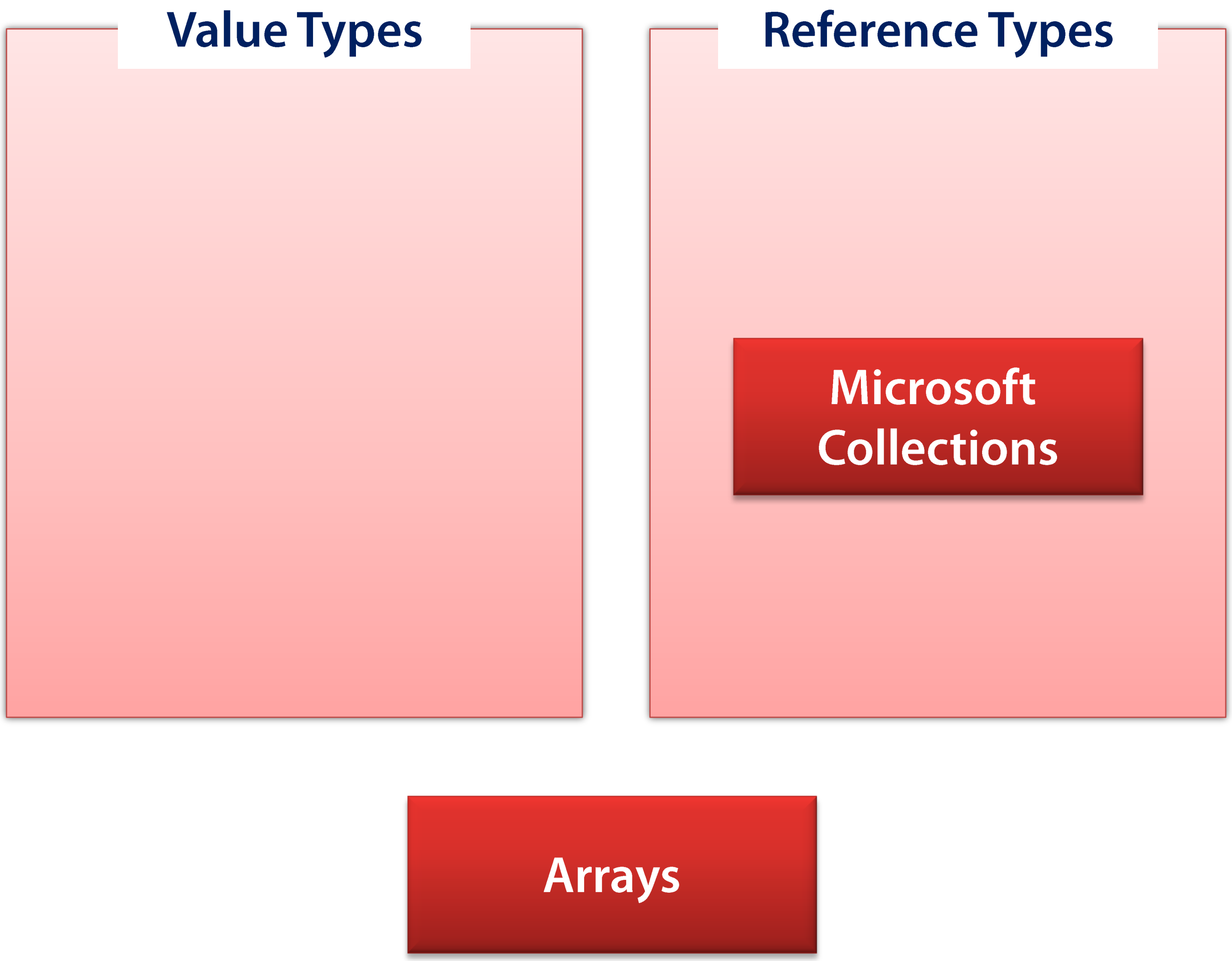
Simon Robinson
http://TechieSimon.com
@TechieSimon

pluralsight
hardcore developer training

Module Overview

- **Arrays and the type system**
  - Reference types
- **Storing Derived Type Instances**
  - Covariance
- **Array capabilities**
  - Copying arrays
  - Sorting elements
  - Finding elements

.NET Types


Value Types
Reference Types
Microsoft Collections
Arrays

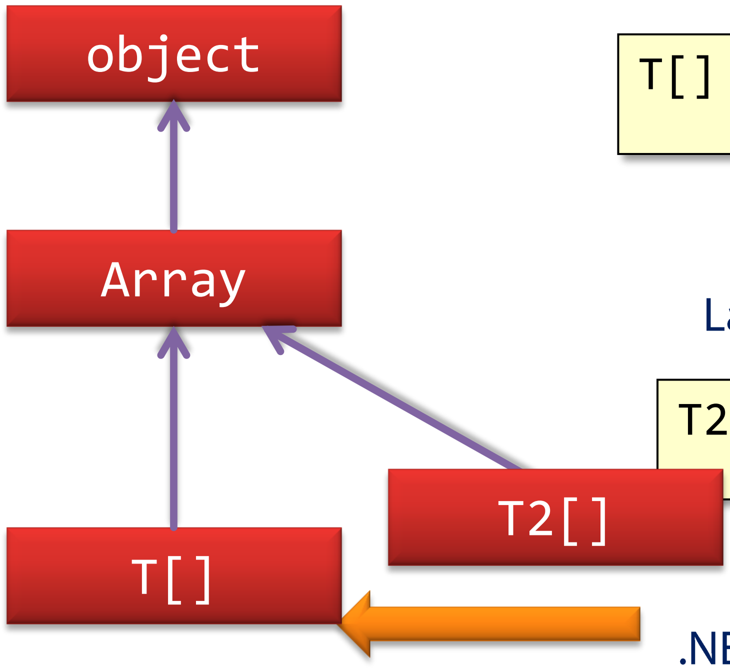
```
class Base
{ … }
```

```
class Derived : Base
{ … }
```

```
Base[] arr = new Base[n];
arr[0] = new Derived();
```

This is fine!

This follows from OOP: A Derived reference can be used in place of a Base reference



Execute this….

```
T[] arr = new T[n];
```

Later execute this….

```
T2[] arr = new T2[n];
```

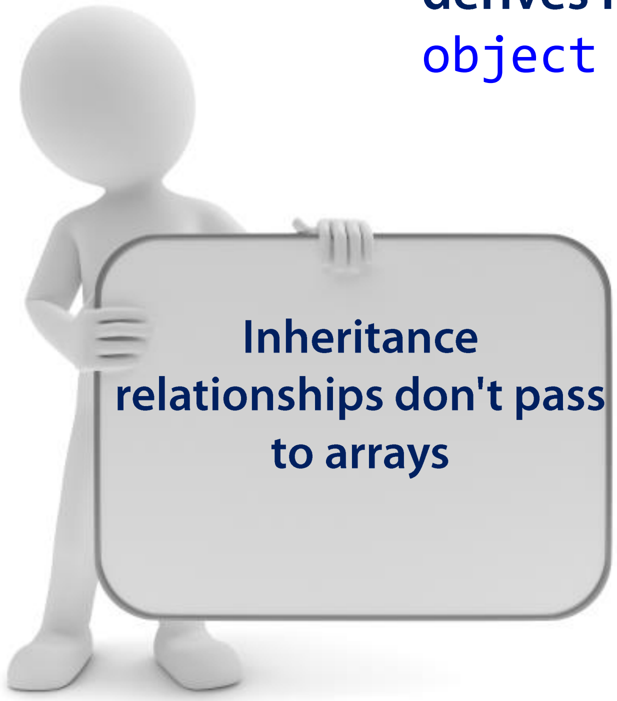
.NET Framework generates this
(if type not already created)

$$\begin{tikzcd}
\texttt{object} \\
\texttt{string} \arrow[u]
\end{tikzcd}$$

`string` **derives from** `object` ✓

$$\begin{tikzcd}
 & \texttt{object} & \\
 & \texttt{Array} \arrow[u] & \\
\texttt{string[]} \arrow[ur] & & \texttt{object[]} \arrow[ul]
\end{tikzcd}$$

`string[]` **does NOT derive from** `object[]` ✗



Inheritance relationships don't pass to arrays

Code Demo

Covariance

Supported for

`arrays`

(but broken) ✗

`IEnumerable<T>`

`IEnumerator<T>`

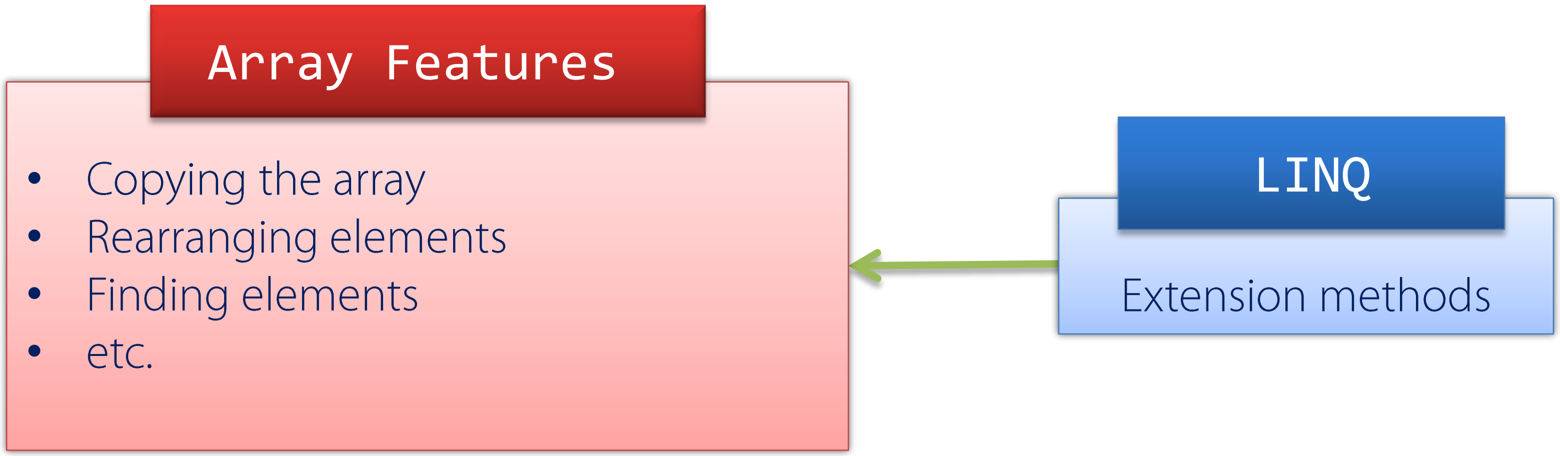
(good) ✓

Not supported for

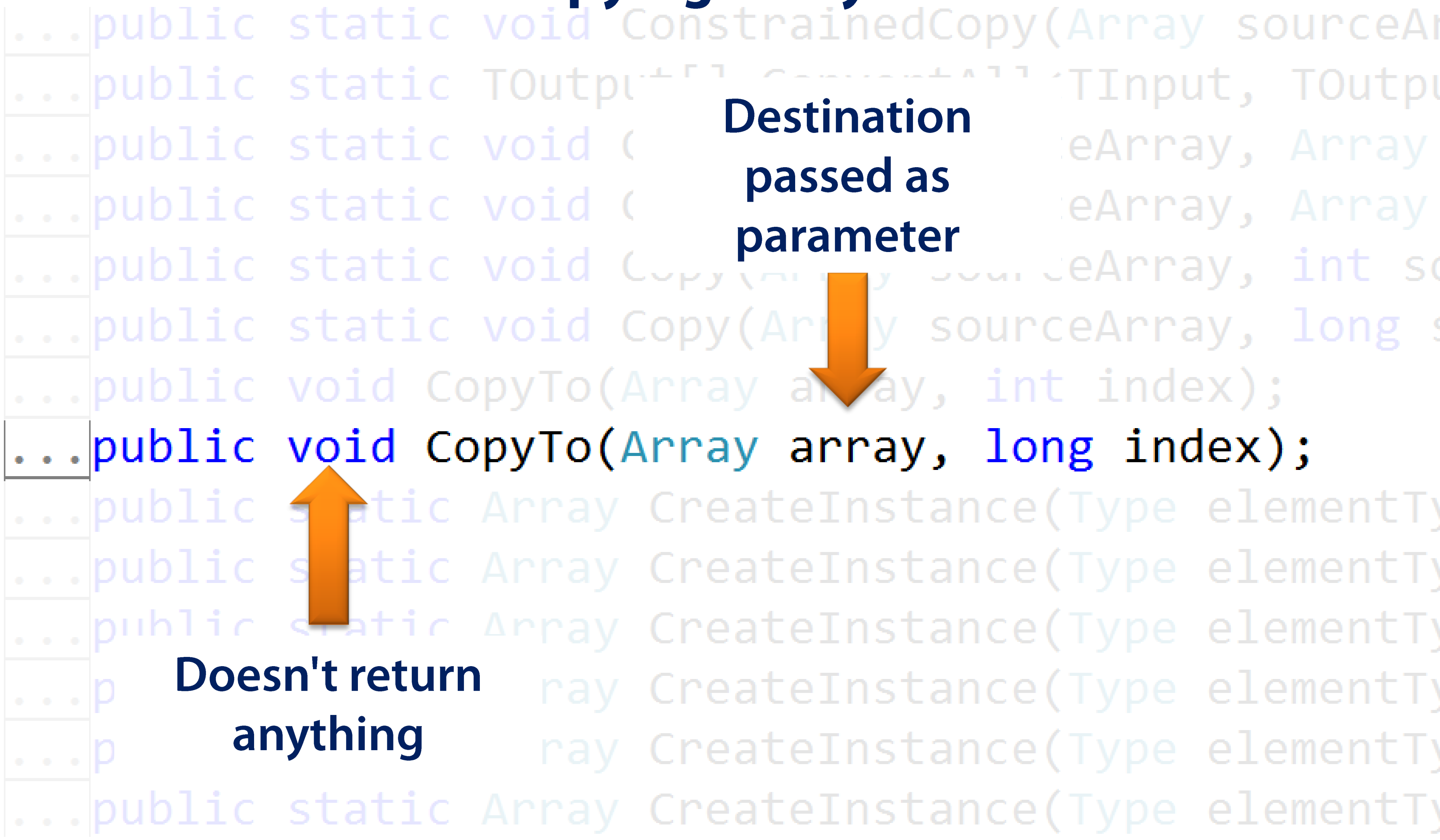
`Any other collection type`

(thankfully)

Array Functionality



Copying Arrays

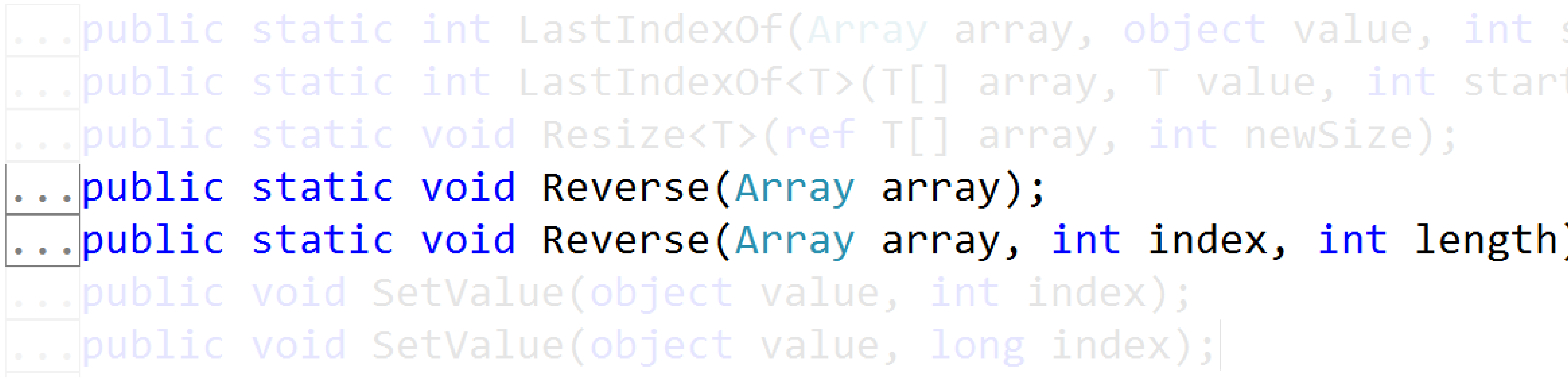


```
...public static void ConstrainedCopy(Array sourceAr
...public static TOutpu          TInput, TOutpu
...public static void (          eArray, Array
...public static void (          eArray, Array
...public static void C          eArray, int so
...public static void Copy(Ar  y sourceArray, long s
...public void CopyTo(Array a   ay, int index);
...public void CopyTo(Array array, long index);
...public s  tic Array CreateInstance(Type elementTy
...public s  atic Array CreateInstance(Type elementTy
...public   atic Array CreateInstance(Type elementTy
...p          ray CreateInstance(Type elementTy
...p          ray CreateInstance(Type elementTy
...public static Array CreateInstance(Type elementTy
```

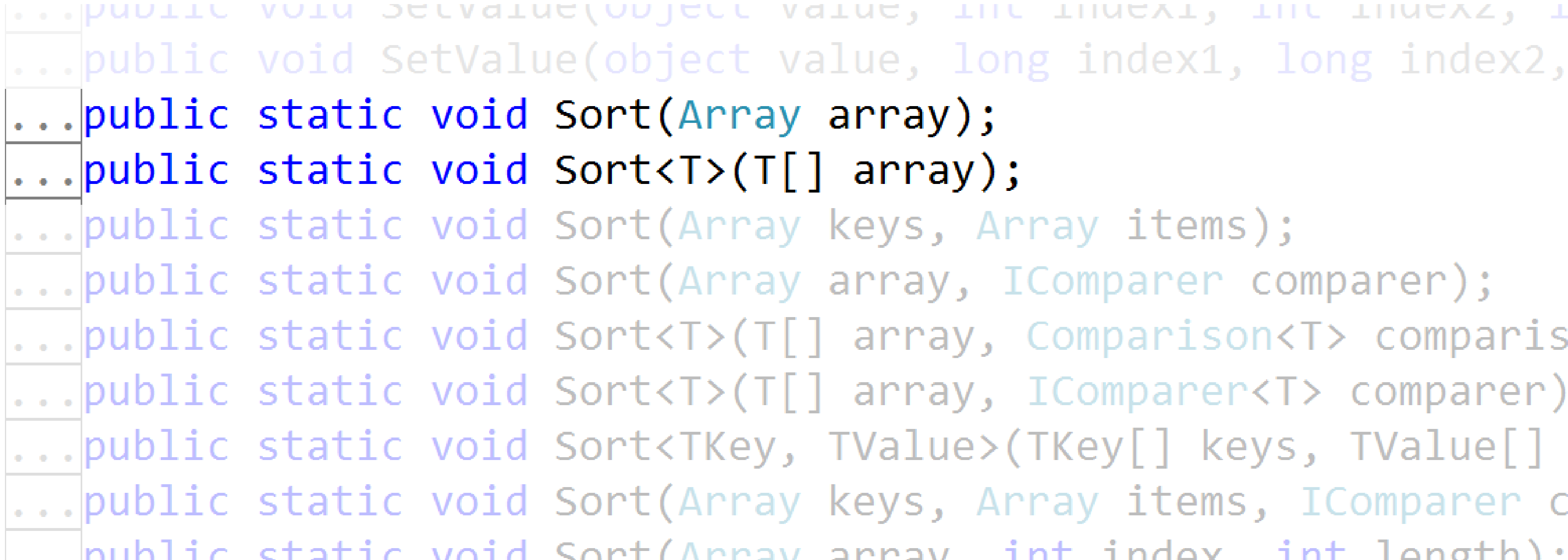
Destination passed as parameter

Doesn't return anything

Reordering Elements

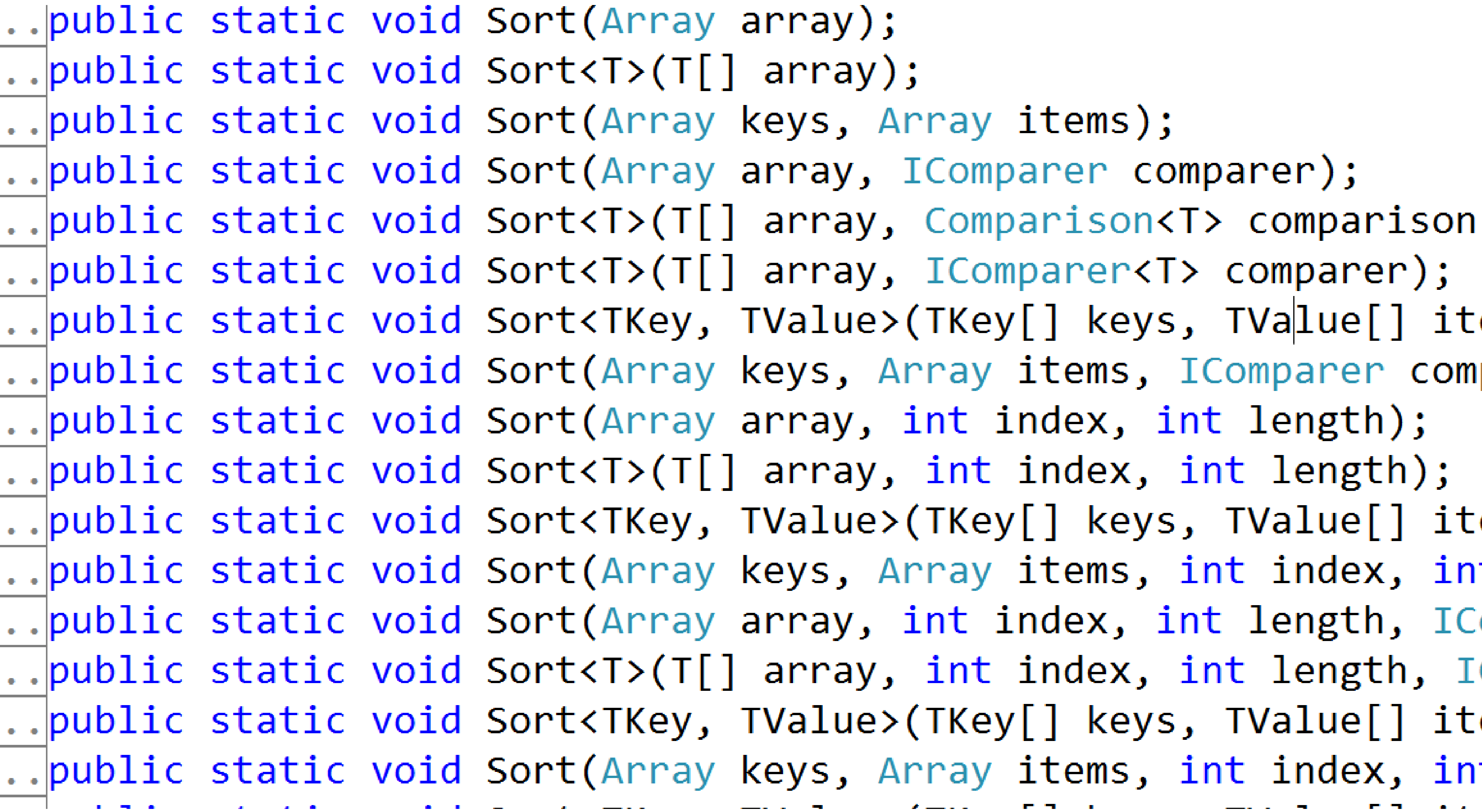


```
...public static int LastIndexOf(Array array, object value, int s
...public static int LastIndexOf<T>(T[] array, T value, int start
...public static void Resize<T>(ref T[] array, int newSize);
...public static void Reverse(Array array);
...public static void Reverse(Array array, int index, int length)
...public void SetValue(object value, int index);
...public void SetValue(object value, long index);
```



```
...public void SetValue(object value, int index1, int index2, i
...public void SetValue(object value, long index1, long index2,
...public static void Sort(Array array);
...public static void Sort<T>(T[] array);
...public static void Sort(Array keys, Array items);
...public static void Sort(Array array, IComparer comparer);
...public static void Sort<T>(T[] array, Comparison<T> comparis
...public static void Sort<T>(T[] array, IComparer<T> comparer)
...public static void Sort<TKey, TValue>(TKey[] keys, TValue[]
...public static void Sort(Array keys, Array items, IComparer c
...public static void Sort(Array array, int index, int length);
```

Sorting Arrays



```
..public static void Sort(Array array);
..public static void Sort<T>(T[] array);
..public static void Sort(Array keys, Array items);
..public static void Sort(Array array, IComparer comparer);
..public static void Sort<T>(T[] array, Comparison<T> comparison
..public static void Sort<T>(T[] array, IComparer<T> comparer);
..public static void Sort<TKey, TValue>(TKey[] keys, TValue[] ite
..public static void Sort(Array keys, Array items, IComparer com
..public static void Sort(Array array, int index, int length);
..public static void Sort<T>(T[] array, int index, int length);
..public static void Sort<TKey, TValue>(TKey[] keys, TValue[] ite
..public static void Sort(Array keys, Array items, int index, in
..public static void Sort(Array array, int index, int length, IC
..public static void Sort<T>(T[] array, int index, int length, I
..public static void Sort<TKey, TValue>(TKey[] keys, TValue[] ite
..public static void Sort(Array keys, Array items, int index, in
```

Sorting Arrays

```
public static void Sort(Array array);
publi
publi                                        items);
public static void Sort(Array array, IComparer comparer);
public static void Sort<T>(T[] array, Comparison<T> comparison
public static void Sort<T>(T[] array, IComparer<T> comparer);
public static void Sort<TKey, TValue>(TKey[] keys, TValue[] ite
```

Example: We will sort days of the week by length of the name

`IComparer<T>:`

I know how to compare instances of T

```
public static void Sort<T>(T[] array, int index, int length, I
public static void Sort<TKey, TValue>(TKey[] keys, TValue[] ite
public static void Sort(Array keys, Array items, int index, in
```

Finding Elements

```
string[] days = {
                       "Monday",
                       "Tuesday",
                       "Wednesday",
                       "Thursday",
                       "Friday",
                       "Saturday",
                       "Sunday" };
```

Find where an element is

Where is 'Tuesday'?

Find an element that satisfies some condition

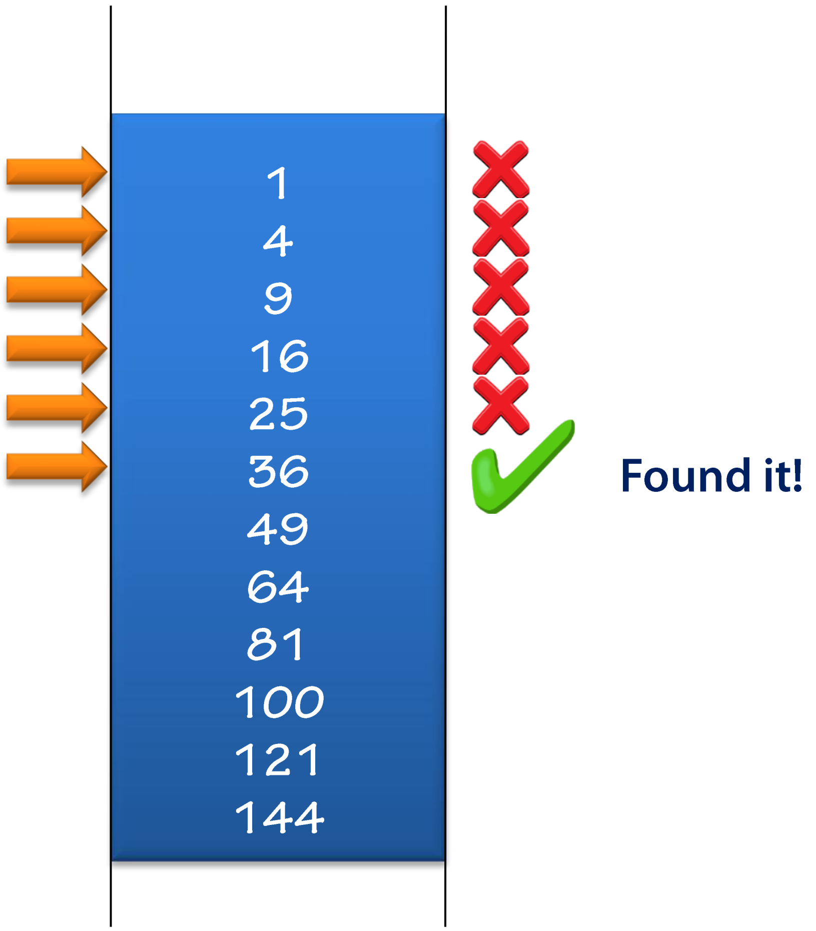
What day begins with 'W'?

Find **all** elements that satisfy some condition

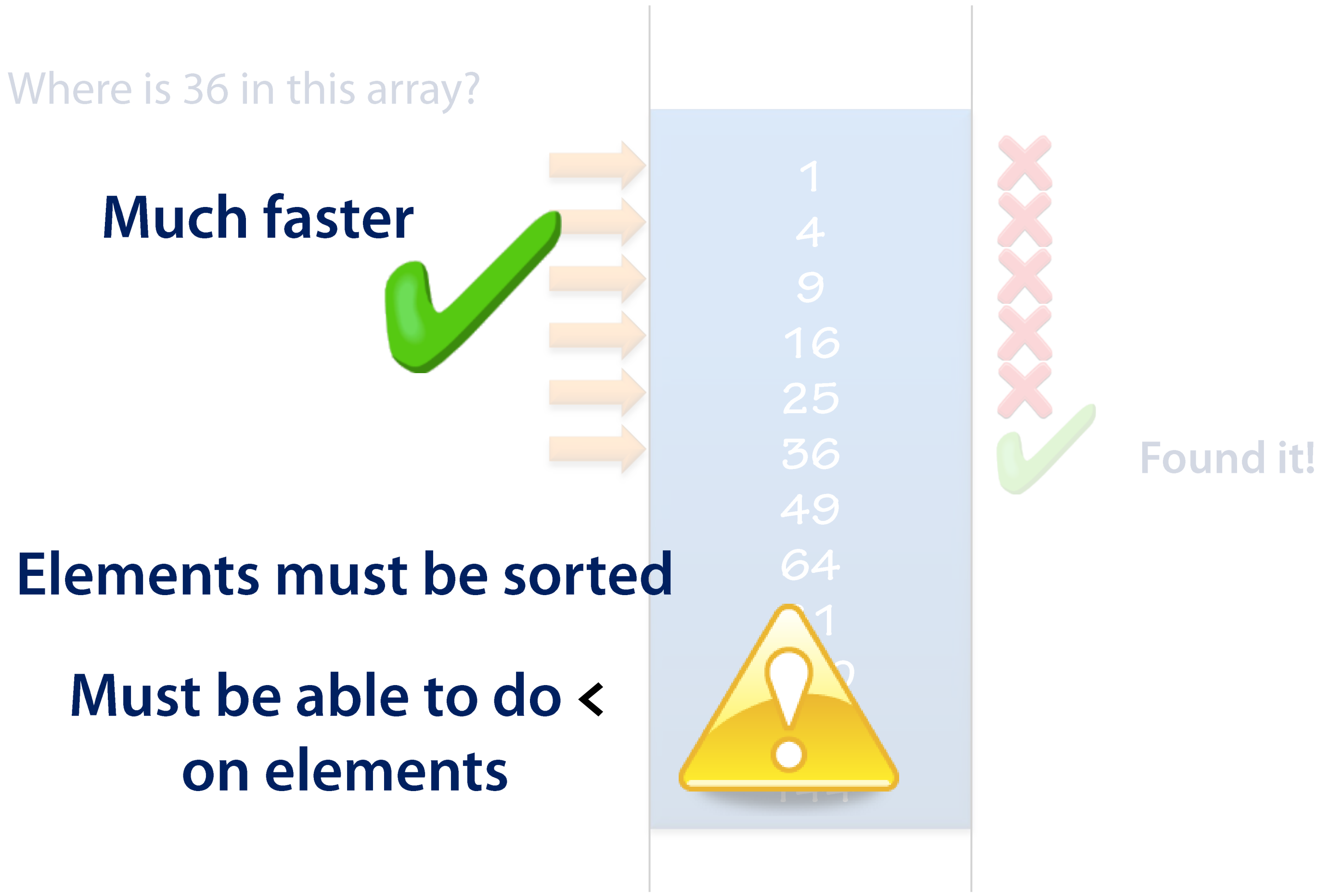
What days have 6 letters?

Finding Elements

Where is 36 in this array?



Binary Searching

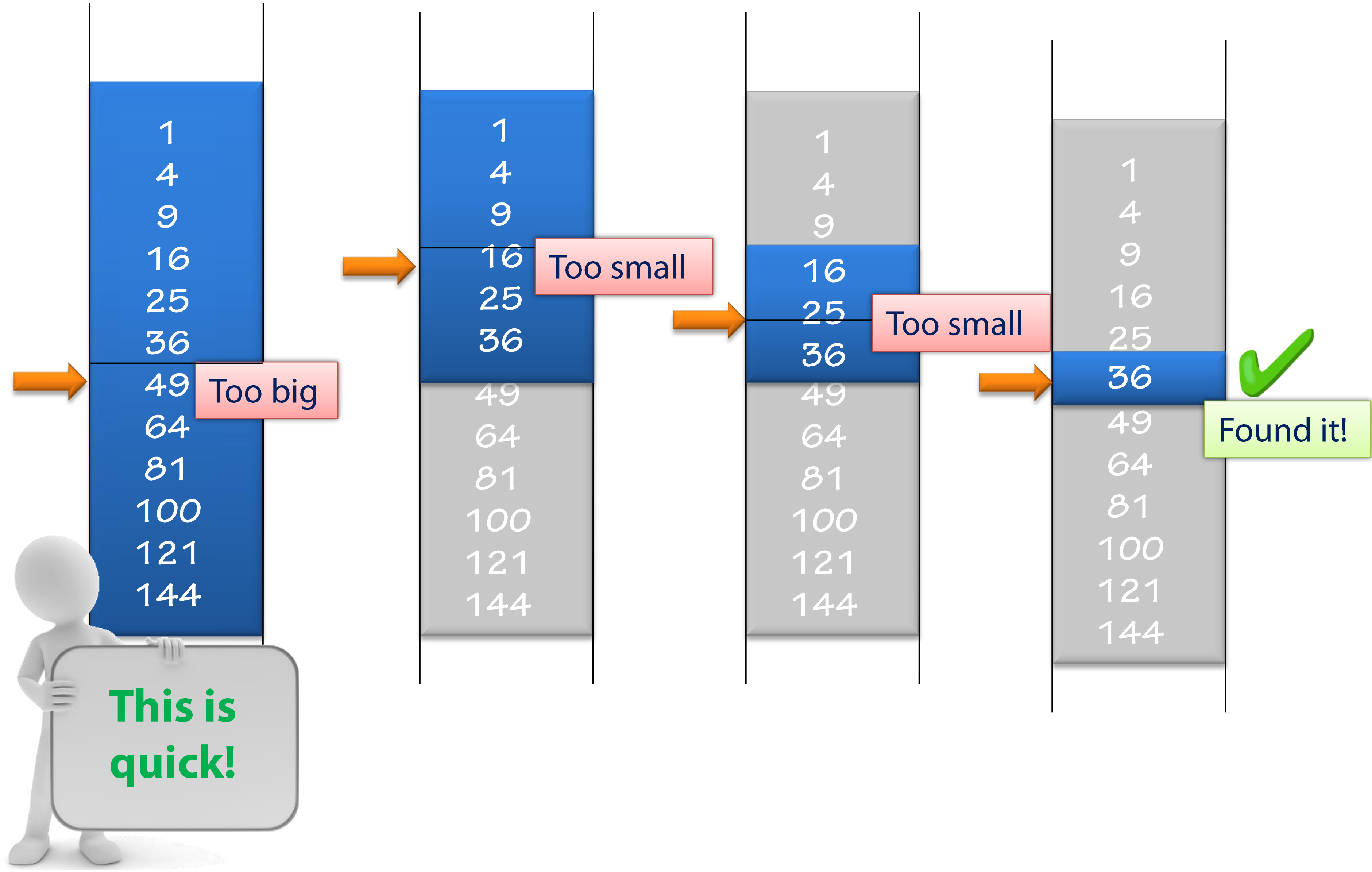


Much faster ✓

Elements must be sorted

Must be able to do < on elements

Where is 36 in this array?



LINQ vs Array Methods?

LINQ	Array members
More suited to interfaces (Good for best practices)	Performance (no LINQ overhead!)
Consistent for all collections	Only arrays (and `List<T>`)
Return new objects	Modify arrays inline

Summary

- **Arrays are reference types**
- **Arrays are covariant**
  - (But it's often a bad idea to use this)
- **Lots of array capabilities out of the box:**
  - Copying arrays
  - Sorting elements
  - Finding elements
- **Many capabilities duplicated by LINQ methods**
  - Usually: LINQ copies, array methods modify inline