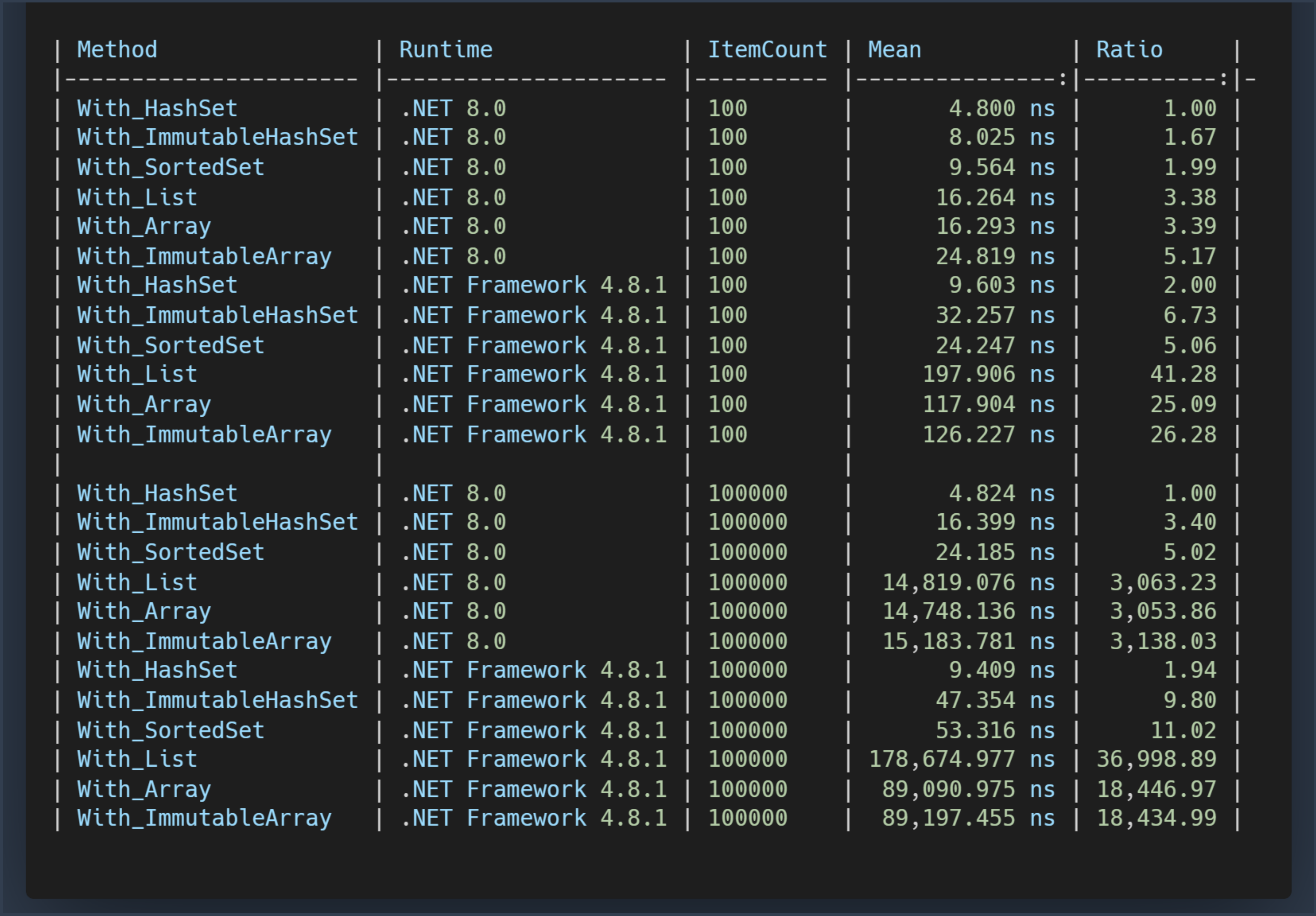
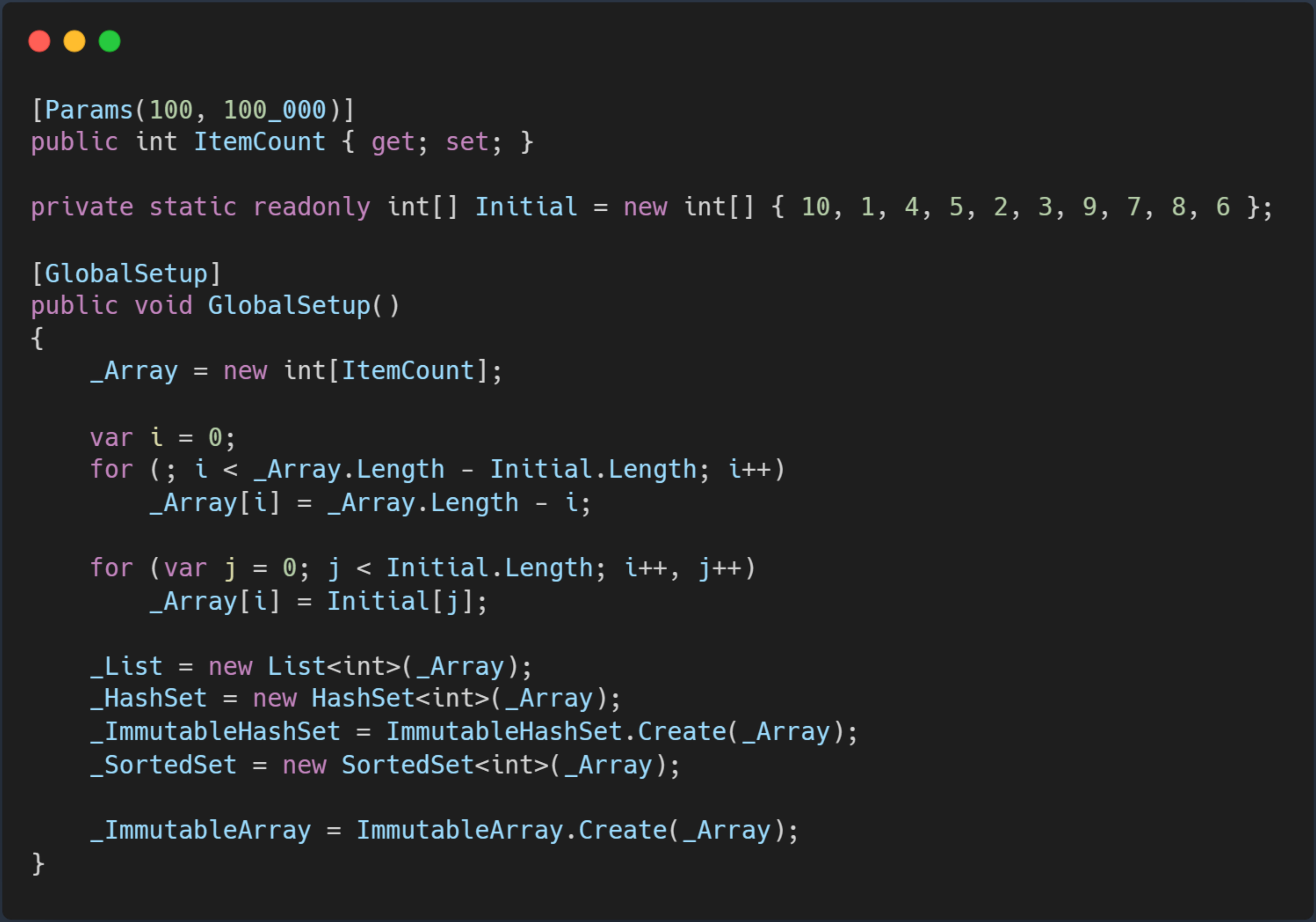
Which collection in C# has fastest “Contains” method?

HashSet



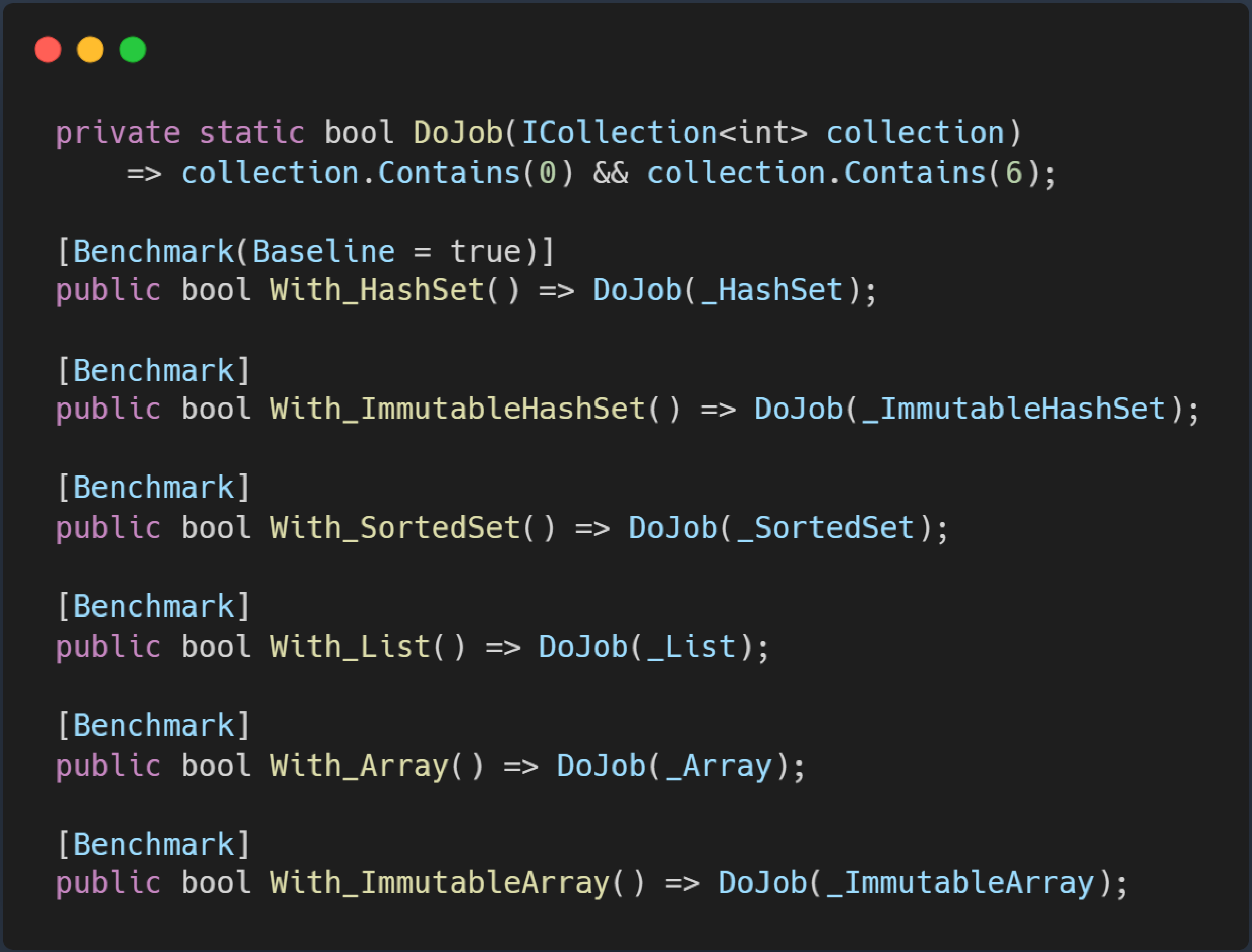
```
| Method                | Runtime              | ItemCount | Mean              | Ratio      |
|---------------------- |--------------------- |---------- |----------------:|----------:|-
| With_HashSet          | .NET 8.0             | 100       |          4.800 ns |       1.00 |
| With_ImmutableHashSet | .NET 8.0             | 100       |          8.025 ns |       1.67 |
| With_SortedSet        | .NET 8.0             | 100       |          9.564 ns |       1.99 |
| With_List             | .NET 8.0             | 100       |         16.264 ns |       3.38 |
| With_Array            | .NET 8.0             | 100       |         16.293 ns |       3.39 |
| With_ImmutableArray   | .NET 8.0             | 100       |         24.819 ns |       5.17 |
| With_HashSet          | .NET Framework 4.8.1 | 100       |          9.603 ns |       2.00 |
| With_ImmutableHashSet | .NET Framework 4.8.1 | 100       |         32.257 ns |       6.73 |
| With_SortedSet        | .NET Framework 4.8.1 | 100       |         24.247 ns |       5.06 |
| With_List             | .NET Framework 4.8.1 | 100       |        197.906 ns |      41.28 |
| With_Array            | .NET Framework 4.8.1 | 100       |        117.904 ns |      25.09 |
| With_ImmutableArray   | .NET Framework 4.8.1 | 100       |        126.227 ns |      26.28 |
|                       |                      |           |                   |            |
| With_HashSet          | .NET 8.0             | 100000    |          4.824 ns |       1.00 |
| With_ImmutableHashSet | .NET 8.0             | 100000    |         16.399 ns |       3.40 |
| With_SortedSet        | .NET 8.0             | 100000    |         24.185 ns |       5.02 |
| With_List             | .NET 8.0             | 100000    |     14,819.076 ns |   3,063.23 |
| With_Array            | .NET 8.0             | 100000    |     14,748.136 ns |   3,053.86 |
| With_ImmutableArray   | .NET 8.0             | 100000    |     15,183.781 ns |   3,138.03 |
| With_HashSet          | .NET Framework 4.8.1 | 100000    |          9.409 ns |       1.94 |
| With_ImmutableHashSet | .NET Framework 4.8.1 | 100000    |         47.354 ns |       9.80 |
| With_SortedSet        | .NET Framework 4.8.1 | 100000    |         53.316 ns |      11.02 |
| With_List             | .NET Framework 4.8.1 | 100000    |    178,674.977 ns |  36,998.89 |
| With_Array            | .NET Framework 4.8.1 | 100000    |     89,090.975 ns |  18,446.97 |
| With_ImmutableArray   | .NET Framework 4.8.1 | 100000    |     89,197.455 ns |  18,434.99 |
```

Data is important! An array with “ItemCount” size. Array filled with “data[i] = length-i”, last 10 item isn’t sorted and ended with number 6



```
[Params(100, 100_000)]
public int ItemCount { get; set; }

private static readonly int[] Initial = new int[] { 10, 1, 4, 5, 2, 3, 9, 7, 8, 6 };

[GlobalSetup]
public void GlobalSetup()
{
    _Array = new int[ItemCount];

    var i = 0;
    for (; i < _Array.Length - Initial.Length; i++)
        _Array[i] = _Array.Length - i;

    for (var j = 0; j < Initial.Length; i++, j++)
        _Array[i] = Initial[j];

    _List = new List<int>(_Array);
    _HashSet = new HashSet<int>(_Array);
    _ImmutableHashSet = ImmutableHashSet.Create(_Array);
    _SortedSet = new SortedSet<int>(_Array);

    _ImmutableArray = ImmutableArray.Create(_Array);
}
```

We just use Contains method
0 -> not existed 1 -> last item



```
private static bool DoJob(ICollection<int> collection)
    => collection.Contains(0) && collection.Contains(6);

[Benchmark(Baseline = true)]
public bool With_HashSet() => DoJob(_HashSet);

[Benchmark]
public bool With_ImmutableHashSet() => DoJob(_ImmutableHashSet);

[Benchmark]
public bool With_SortedSet() => DoJob(_SortedSet);

[Benchmark]
public bool With_List() => DoJob(_List);

[Benchmark]
public bool With_Array() => DoJob(_Array);

[Benchmark]
public bool With_ImmutableArray() => DoJob(_ImmutableArray);
```