Romain Ottonelli Dabadie • Following
Author of Mastering Visual Studio | 👫 Proud husband & dad | 💻 Senior Software Engine...
Visit my website
13h •

Is size matters?

Using collection it could be.

In c# a List<T> are defined with a default initial capacity, when the capacity is exceeded, the list automatically resizes to double its current capacity.

Each resize operation requires:

- Allocating a new, larger array
- Copying all existing elements to the new array
- Discarding the old array

For dictionaries and Hashset the resizes is launch when elements reach the threshold of (capacity x load factor). The resizing involves rehashing all existing elements into a larger internal array.

This automatic resizing is convenient but can cause performance issues with large collections, as each resize operation has O(n) time complexity.

That's why specifying a capacity is beneficial when you can reasonably estimate the collection size.

Advantages of specifying collection size:

1. **Performance improvements** - Pre-allocating memory reduces the need for resizing operations, which can be costly
2. **Memory efficiency** - You can avoid over-allocation when you know exactly how many elements you'll need
3. **Reduced fragmentation** - Fewer resize operations means less memory fragmentation
4. **Predictable performance** - Your code's performance becomes more consistent without unexpected resize operations

However, it's can be dangerous and offers some disisadvantages:

1. **Memory waste** if you overestimate the needed size

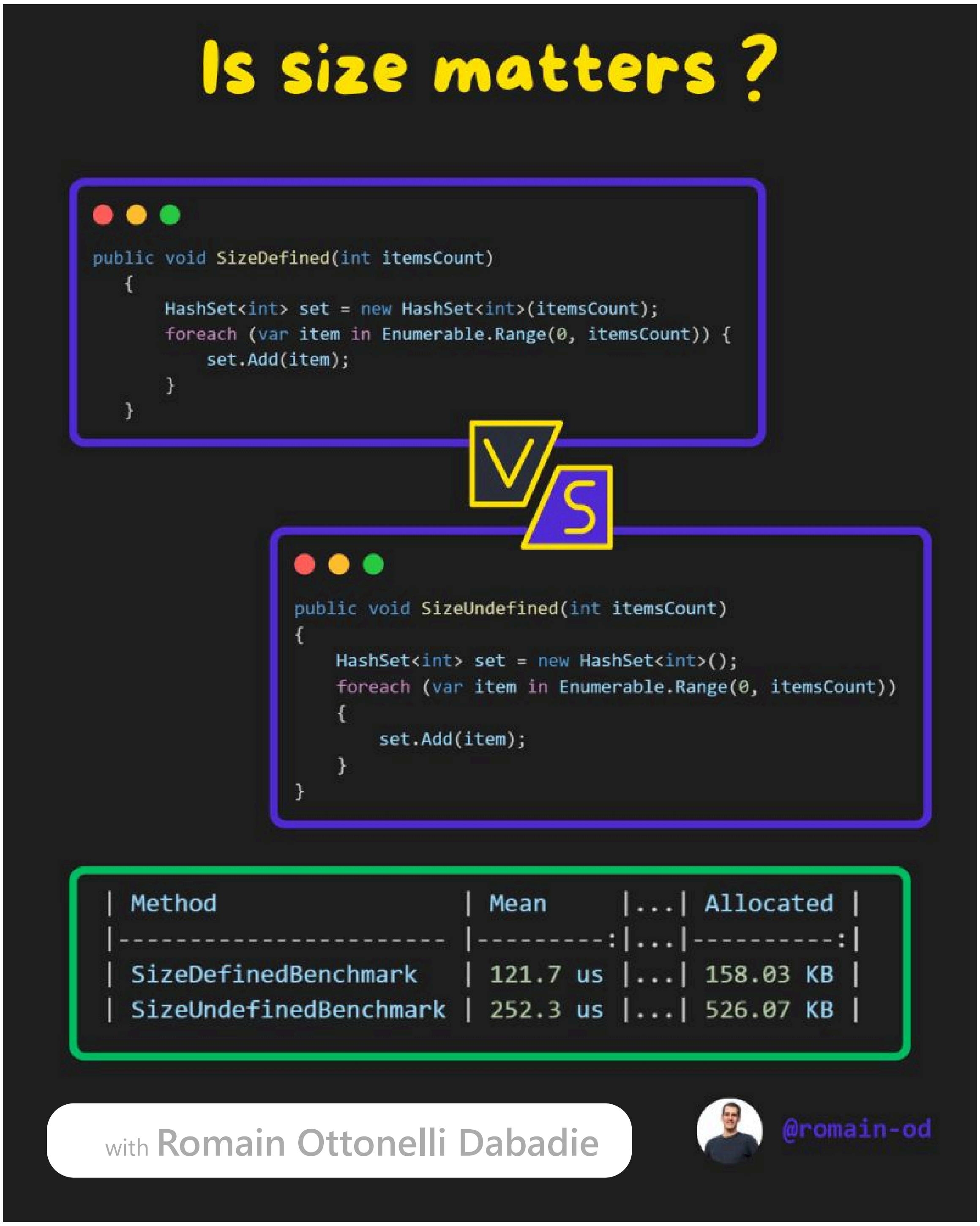
2. **Extra complexity** in your code when the exact size isn't known in advance

3. **Potential inefficiency** if collection needs to grow beyond initial capacity anyway

4. **Less readable code** in some cases where size calculation is complex

Have you ever specified the size of your collection? Share your experience?

#csharp #dotnet #perfomance #collection #softwareengineer



```
public void SizeDefined(int itemsCount)
  {
      HashSet<int> set = new HashSet<int>(itemsCount);
      foreach (var item in Enumerable.Range(0, itemsCount)) {
          set.Add(item);
      }
  }
```

```
public void SizeUndefined(int itemsCount)
{
    HashSet<int> set = new HashSet<int>();
    foreach (var item in Enumerable.Range(0, itemsCount))
    {
        set.Add(item);
    }
}
```

```
| Method                 | Mean      |...| Allocated |
|----------------------- |----------:|...|----------:|
| SizeDefinedBenchmark   | 121.7 us  |...| 158.03 KB |
| SizeUndefinedBenchmark | 252.3 us  |...| 526.07 KB |
```

SERKUT YILDIRIM and 149 others   47 comments 8 reposts