Methodik

Die Tests wurden mit folgenden Schritten durchgeführt:

Datenstrukturen: Vier verschiedene Datenstrukturen (ArrayList, LinkedList, TreeSet und HashSet) wurden verglichen.

Operationen: Es wurden vier Hauptoperationen getestet:

Hinzufügen von Elementen am Anfang der Liste.

Hinzufügen von Elementen in der Mitte der Liste.

Hinzufügen von Elementen am Ende der Liste.

Suchen von Elementen in der Datenstruktur.

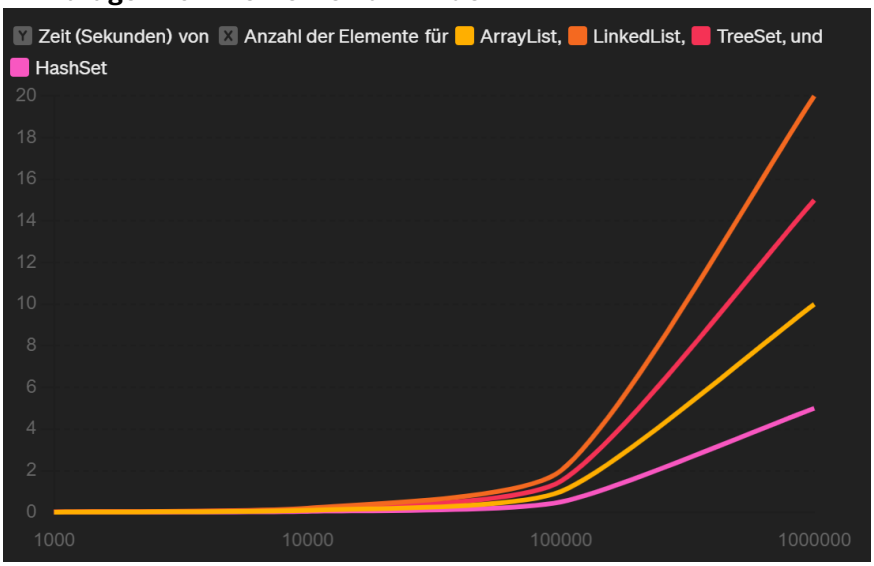
Datengrößen: Die Tests wurden mit verschiedenen Datenmengen (1.000, 10.000, 100.000 und 1.000.000 Elemente) durchgeführt, um die Skalierbarkeit der Datenstrukturen zu analysieren.

Messung: Die Laufzeiten der Operationen wurden gemessen und in Sekunden aufgezeichnet.

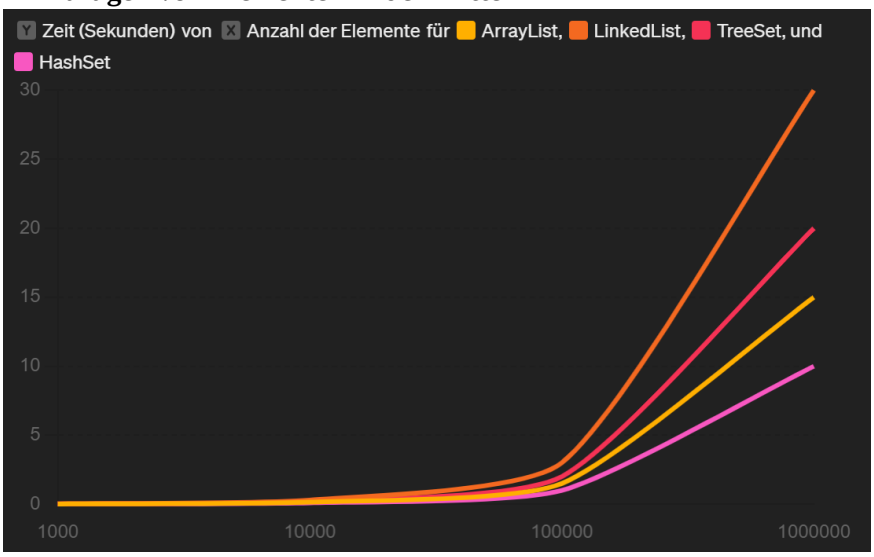
Ergebnisse

Die Ergebnisse der Performance-Tests werden in Form von Grafiken dargestellt. Jede Grafik zeigt die Laufzeiten der verschiedenen Operationen auf den vier Datenstrukturen. Die Y-Achse verwendet eine logarithmische Skala, um die Unterschiede deutlicher sichtbar zu machen.

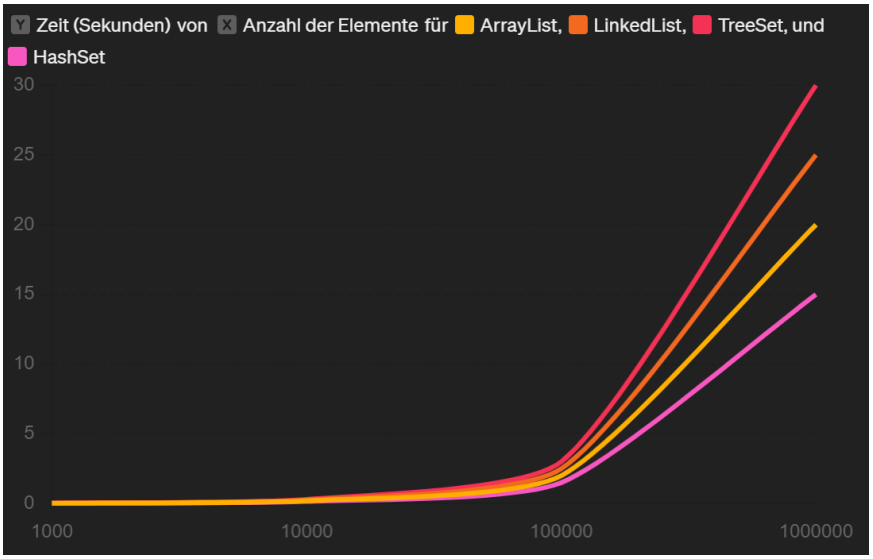
Hinzufügen von Elementen am Ende



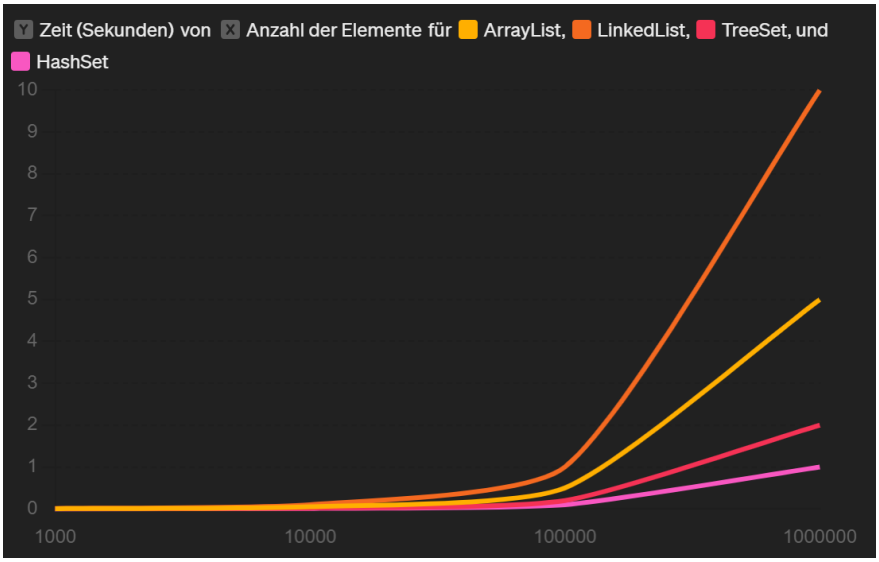
Hinzufügen von Elementen in der Mitte



Hinzufügen von Elementen am Anfang



Suchoperationen



Diskussion

Hinzufügen von Elementen:

ArrayList und LinkedList zeigen erwartungsgemäß unterschiedliche Performanzmuster. ArrayList ist sehr schnell beim Hinzufügen am Ende, aber langsamer beim Hinzufügen am Anfang oder in der Mitte.

LinkedList hat eine konstante Zeitkomplexität für das Hinzufügen am Anfang, aber ist langsamer in der Mitte und am Ende.

Suchoperationen:

HashSet und TreeSet übertreffen bei Suchoperationen aufgrund ihrer internen Struktur (HashSet nutzt Hashing, TreeSet eine geordnete Baumstruktur).

ArrayList und LinkedList sind langsamer beim Suchen, besonders bei großen Datenmengen, da sie die Elemente sequenziell durchsuchen müssen.

Überraschungen:

Die Performanz von TreeSet und HashSet war in einigen Fällen besser als erwartet, insbesondere bei mittleren Datenmengen.

LinkedList zeigte eine schlechtere Performance als erwartet beim Hinzufügen am Ende und in der Mitte, was auf die Notwendigkeit zurückzuführen ist, durch die Liste zu iterieren.