Aufwandsbeurteilung: Wie "schnell" ist ein Algorithmus?

Wenn man ein Element in einer sortierten Liste sucht und den Aufwand vergleicht, je nachdem ob man die einfache Suche (der Reihe nach jedes Element ansehen) oder die binäre Suche verwendet, kann man zu den folgenden Erkenntnissen gelangen.

Angenommen, es dauert 1ms, um ein Element zu überprüfen - wie lange dauert es jeweils, in einer Liste mit 100 Elementen das gesuchte zu finden? Naja, das kommt drauf an - es könnte ja bei beiden Methoden sein, das bereits die erste Betrachtung das gesuchte Element findet, dann dauert es 1ms[1)], es zu finden. Bei der einfachen Suche kann es aber durchaus auch 100ms dauern, wenn das gesuchte Element als letztes überprüft wird.

Darum einigt man sich, dass man bei der Beurteilung von Algorithmen bevorzugt den schlechtesten Fall ("worst case") betrachtet, um einen Eindruck zu bekommen, wie effektiv ein Algorithmus arbeitet

Der schlechteste Fall bei der binären Suche dauert 7ms, also 15 mal so lang. Die binäre Suche ist also 15 mal schneller als die einfache suche?! Stimmt das? Dazu betrachten wir die folgende Tabelle, in der die "worst case"-Zeiten dargestellt sind.

Zahl der Elemente	Einfache Suche	Binäre Suche
100	100ms	7ms
10.000	10 Sekunden	14ms
1.000.000.000	11 Tage	32ms

Man sieht, dass die Laufzeiten **mit der Zunahme der Zahl der Elemente sehr unterschiedlich zunehmen**.

Um diesen Umstand darzustellen, verwendet man die **Landau-Notation** (oder auch "Big-O-Notation). Die Landau Notation ermöglicht es, die Anzahl der Operationen die Algorithmen zur Lösung eines Problems benötigen zu vergleichen - sie teilt dir mit wie schnell die Zahl der Operationen des Algorithmus **anwächst**.

Für unser Problem oben:

Einfache Suche	Binäre Suche
O(n)	O(log n)

1)
wenn man annimmt, dass die anderen Operationen im Vergleich fast keine Zeit beanspruchen