Laufzeit- und Speichereffizienz

Der Begriff Effizienz bezeichnet ein Merkmal von Programmen, dass den Ressourcenverbrauch bei der Lösung eines bestimmten Problems beschreibt. Zu den wesentlichen Faktoren gehören dabei die Zeit welche ein Programm zur Ausführung benötigt und die Größe des vom Programm belegten Speichers. Eine hohe Effizienz bedeutet einen niedrigen Aufwand an Ressourcen. Grundsätzlich kann in Laufzeiteffizienz und Speichereffizienz unterschieden werden.

Laufzeiteffizienz beschreibt die Ausführungsdauer eines Programms, also wie schnell ein bestimmtes Problem gelöst wird. Da die benötigte Zeit ja nach verwendeter Hardware und Programmiersprache variieren kann, wird zum Vergleich der Laufzeiten die Anzahl der benötigten Rechenschritte genutzt. Ein Rechenschritt ist beispielsweise die Addition zweier Variablen oder die Prüfung einer if-Bedingung. Eine hohe Laufzeiteffizienz bedeutet eine geringe Laufzeit.

Speichereffizienz beschreibt den Speicherbedarf eines Programms bei der Ausführung. Die Speichereffizienz ist unter anderem abhängig von der Anzahl und Art der verwendeten Variablen, sowie Anzahl der Eingabeparameter. Eine hohe Speichereffizienz bedeutet, dass nur wenig Speicherplatz zur Lösung des Problems verwendet wird.

Zwischen Laufzeiteffizienz und Speichereffizienz besteht eine Wechselbeziehung. So kann zum Beispiel die Laufzeit eines Programms verbessert werden, indem zuvor berechnete Zwischenergebnisse für die spätere Verwendung in Variablen gespeichert werden, statt diese zum entsprechenden Zeitpunkt erneut zu berechnen. Dadurch benötigt das Programm jedoch mehr Speicher, sodass die Verbesserung der Laufzeiteffizienz eine Verschlechterung der Speichereffizienz nach sich zieht. Ebenso kann häufig die Verbesserung der Speichereffizienz eine Verschlechterung der Laufzeiteffizienz bedeuten. Je nach Anwendungsgebiet kann der Laufzeit- oder Speichereffizienz eine höhere Bedeutung beigemessen werden. Ein Programmierer muss dies bei der Entwicklung effizienter Anwendungen bedenken und entsprechend der Anforderungen optimieren.

Im folgenden ist Beispiel für solch ein Abtausch dargestellt, wo durch die wiederholte neue Allokation von Speicherplatz die Laufzeit erhöht wird:

Einmalige Deklaration der Variable specificNumber

```
1 static int addValue(int n) {
2   int specificNumber = 10;
3   for (int i = 0; i < 5; i++) {
4     n = n + specificNumber;
5   }
6 }
```

Mehrmalige Deklaration der Variable specificNumber

```
1 static int addValue(int n) {
2   for (int i = 0; i < 5; i++) {
3     int specificNumber = 10;
4     n = n + specificNumber;
5   }
6 }
```

Im oberen Codebeispiel wird die Variable specificNumber nur einmal vor der for-Schleife deklariert und ist damit im gesamten Kontext der Methode addValue verfügbar. Im unteren Codebeispiel hingegen wird die Variable specificNumber bei jedem Schleifendurchlauf erneut deklariert, was so zu einer erhöhten Laufzeit führt.

Um die Laufzeiteffizienz eines Programms untersuchen zu können wird die O-Notation verwendet. Dabei abstrahiert man von dem tatsächlichen Zeitaufwand, da dieser systemspezifisch ist. Stattdessen wird die Laufzeit in Abhängigkeit der Eingabewerte eines Programms betrachtet und am ungünstigsten Fall orientiert.

Links zur weiteren Vertiefung:

- Wikipedia: Effizienz
- Freie Universität Berlin - Vorlesungsfolien: Effizienz und Komplexität
- Freie Universität Berlin - Vorlesungsfolien: O-Notation
- Wachtler - Definition: Effizienz von Algorithmen