Lernnachweis zu Kompetenz B1F

Aufteilung von Algorithmen in funktionale Teilstücke

Einleitung

Die Fähigkeit, Algorithmen in funktionale Teilstücke aufzuteilen, indem Rekursion verwendet wird, ist eine wichtige Technik in der Softwareentwicklung. Rekursion ermöglicht es, komplexe Aufgaben in kleinere, wiederholbare Teilaufgaben zu zerlegen, was die Lesbarkeit und Wartbarkeit des Codes verbessern kann. In dieser Dokumentation werden wir das Konzept der Aufteilung von Algorithmen mit Rekursion näher erläutern und dies anhand von Python-Codebeispielen verdeutlichen.

Warum ist die Aufteilung von Algorithmen mit Rekursion wichtig?

Die Verwendung von Rekursion ermöglicht eine elegante und klare Aufteilung von Aufgaben in der Programmierung. Sie führt oft zu kürzerem und verständlicherem Code, der leichter gewartet werden kann. Die Verwendung von Rekursion kann auch in vielen problemorientierten Algorithmen wie dem Teilen und Erobern (Divide and Conquer) äußerst nützlich sein.

1. Rekursive Sortierfunktion (Bubble Sort)

In dieser Version verwenden wir Rekursion, um den Bubble-Sort-Algorithmus aufzuteilen:

```
def bubble_sort_recursive(arr, n=None):
    if n is None:
        n = len(arr)
    if n == 1:
        return arr

    for i in range(n - 1):
        if arr[i] > arr[i + 1]:
            arr[i], arr[i + 1] = arr[i + 1], arr[i]

    return bubble_sort_recursive(arr, n - 1)
```

Die Funktion «bubble_sort_recursive» sortiert das übergebene Array «arr» rekursiv, indem sie in jedem Durchlauf das größte Element an das Ende des Arrays bewegt. Dies geschieht, bis das gesamte Array sortiert ist.

2. Rekursive lineare Suche

Hier ist eine rekursive Version der linearen Suchfunktion:

```
def linear_search_recursive(arr, target, index=0):
    if index == len(arr):
        return -1
    if arr[index] == target:
        return index
    return linear_search_recursive(arr, target, index + 1)
```

Die Funktion «linear_search_recursive» sucht das «target» in «arr» rekursiv, indem sie das Array schrittweise durchläuft und das gefundene Element zurückgibt oder -1 zurückgibt, wenn das Element nicht gefunden wird.

Diese rekursiven Versionen der Beispiele veranschaulichen, wie Rekursion verwendet werden kann, um Algorithmen in funktionale Teilstücke aufzuteilen, ohne Schleifen (Loops) zu verwenden.

Fazit

Die Aufteilung von Algorithmen in funktionale Teilstücke mit Rekursion ist eine wichtige Technik in der Softwareentwicklung. Sie führt zu sauberem und verständlichem Code und ermöglicht die Lösung komplexer Probleme in handhabbare Teilaufgaben. Die vorgestellten Python-Beispiele zeigen, wie diese Technik in der Praxis angewendet werden kann. Dies ist eine wertvolle Kompetenz für Softwareentwickler, um effiziente und gut wartbare Programme zu schreiben.