AuD - Zusammenfassung

Moritz Gerhardt

Contents

Sektion 1 Was ist ein Algorithmus?

Ein Algorithmus beschreibt eine Handlungsvorschrift zur Umwandlung von Eingaben in eine Ausgabe.
Dabei sollte ein Algorithmus im allgemeinen folgende Vorraussetzungen erfüllen:

1. Bestimmt:
   - Determiniert: Bei gleicher Eingabe liefert der Algortihmus gleiche Ausgabe.
     $\implies$ Ausgabe nur von Eingabe abhängig, keine äußeren Faktoren.
   - Determinismus: Bei gleicher Eingabe läuft der Algorithmus immer gleich durch die Eingabe.
     $\implies$ Gleiche Schritte, Gleiche Zwischenstände.
2. Berechenbar:
   - Finit: Der Algorithmus ist als endlich definiert. (Theoretisch)
   - Terminierbar: Der Algorithmus stoppt in endlicher Zeit. (Praktisch)
   - Effektiv: Der Algorithmus ist auf Maschine ausführbar.
3. Andwendbar:
   - Allgemein: Der Algorithmus ist für alle Eingaben einer Klasse anwendbar, nicht nur für speziellen Fall.
   - Korrekt: Wenn der Algorithmus ohne Fehler terminiert, ist die Ausgabe korrekt.

Sektion 2 Sortieren

2.1 Sortierproblem

Sortieralgorithmen sind die wohl am häufigsten verwendeten Algorithmen. Hierbei wird als Eingabe eine Folge von Objekten gegeben, die nach einer bestimmten Eigenschaft sortiert werden. Der Algorithmus soll die Eingabe in der richtigen Reihenfolge (nach einer bestimmten Eigenschaft) zur Ausgabe umwandeln. Es wird hierbei meist von einer total geordneten Menge ausgegangen. (Alle Elemente sind miteinander vergleichbar).
Eine Totale Ordnung wie folgt definiert:

Eine Relation $\leq$ auf M ist eine totale Ordnung, wenn:

- Reflexiv: $\forall x \in M : x \leq x$
  (x steht in Relation zu x)
- Transitiv: $\forall x, y, z \in M : x \leq y \land y \leq z \implies x \leq z$
  (Wenn x in Relation zu y steht und y in Relation zu z steht, so folgt, dass x in Relation zu z steht)
- Antisymmetrisch: $\forall x, y \in M : x \leq y \land y \leq x \implies x = y$
  (Wenn x in Relation zu y steht und y in Relation zu x steht, so folgt, dass x = y)
- Totalität: $\forall x, y \in M : x \leq y \lor y \leq x$
  (Alle Elemente müssen in einer Relation zueinander stehen)

2.2 Insertion Sort

```
Function insertion_sort(A):
    for i = 1 to A.length - 1 do
        key = A[i]                          ▷ Element zum Sortieren
        j = i - 1                           ▷ Einfügepunkt wird von hinten gesucht
        while j >= 0 and A[j] > key do
            A[j + 1] = A[j]                 ▷ Elemente nach Rechts verschieben
            j = j - 1
        end
        A[j + 1] = key                      ▷ Element wird in den Einfügepunkt geschoben
    end
end
```