

2.3 Sortieren

- 2.3.1 Einleitung
- 2.3.2 Einfache Sortierverfahren
- 2.3.3 Höhere Sortierverfahren
- 2.3.4 Komplexität von Sortierverfahren
- 2.3.5 Spezielle Sortierverfahren



Sortierproblem

- Bringe eine gegebene Folge von Datenobjekten in eine wohldefinierte Reihenfolge.
- Folge \neq Menge !! (doppelte Objekte)
- Reihenfolge (Ordnungsrelation)



Ordnungsrelationen

- Menge M , Relation \leq
- Partielle Ordnung
 - Reflexivität $x \leq x$
 - Transitivität $x \leq y \wedge y \leq z \Rightarrow x \leq z$
 - Antisymmetrie $x \leq y \wedge y \leq x \Rightarrow x = y$
- Strikter Anteil
 - $x < y :\Leftrightarrow x \leq y \wedge x \neq y$



Ordnungsrelationen

- Menge M , Relation \leq
- Totale Ordnung
 - Trichotomie
 $\forall x, y: x < y \vee x = y \vee x > y$
- Sortierschlüssel
 - $\text{key}[\text{objekt}]$
 - z.B. Adresse nach Nachnamen sortiert



Ordnungsrelationen

- Skalare Typen besitzen in der Regel eine totale Ordnung
- Mengen von Integer 2^N , Mengeninklusion \subseteq
 - Partielle Ordnung ... $\{1,2\}, \{2,3\}$
- Punkte in der Ebene $\mathbb{R} \times \mathbb{R}$
 - Keine Ordnung
 $(x,y) \leq (u,v) \Leftrightarrow x^2 + y^2 \leq u^2 + v^2$
 - Totale Ordnung (Lexikographisch)
 $(x,y) \leq (u,v) \Leftrightarrow x < u \vee (x=u \wedge y \leq v)$



Sortierproblem

- Gegeben eine Folge von Objekten
 R_1, \dots, R_n
- Schreibweise
 $R_i \leq R_j \Leftrightarrow \text{key}[R_i] \leq \text{key}[R_j]$
- Finde eine Permutation π , so dass
 $\forall i=1, \dots, n-1: R_{\pi(i)} \leq R_{\pi(i+1)}$
- Insbesondere
 $\forall i, j: i < j \Rightarrow R_{\pi(i)} \leq R_{\pi(j)}$



Warum sortieren?

- Bringe Objekte mit gleichen oder ähnlichen Schlüsseln zusammen
- Detektiere doppelte Objekte
- Vorbereitung zum Suchen
 - $O(n) \rightarrow O(\log n)$
 - Lohnt sich nur, wenn
$$m \times n \times C_1 > C_{\text{sort}}(n) + m \times \log(n) \times C_2$$



Begriffe

- Direktes / Indirektes Sortieren
 - Direkt: Umkopieren der Objekte R_i
 - Indirekt: Erzeuge Permutationstabelle π
 - Initialisierung $\pi(i) := i$
 - Zugriff per $R_{\pi(i)}$
 - Sortieren bzgl. mehrerer Schlüssel / Relationen
- Stabilität (Reihenfolge von Objekten mit gleichem Schlüssel bleibt erhalten)



Kriterien

- Berechnungsaufwand
 - $O(n^2)$, $O(n \times \log n)$, $O(n)$
- Worst vs. Average Case
 - Ausnutzen von Vorsortierungen
- Speicherbedarf
 - Kopieren
 - In-place



In den meisten Anwendungen relevant: Der Average-Case.