

# Programmiertechnik II

## Radixsort

# Spezialisierte Sortierverfahren

- Annahmen über die Struktur der Schlüssel:
  - Schlüssel selber ist strukturierte Folge von “Ziffern”
  - Beispiel: Zahlen als Schlüssel (Folge von Bits)
  - Beispiel: Strings als Schlüssel (Folge von Zeichen)
  - Sortierung erfolgt “lexikographisch”
  - Elemente des Schlüssels stammen aus “kleiner” Menge (Bits: zwei Werte; Zeichen: z.B. 256 Werte)
    - Basis des Positionssystems sei **R**
- Sortierverfahren mit “stückweiser” Schlüsselverarbeitung:  
*radix sort*

# Radixsort

- Abstrakte Schlüsseloperation: Liefere n-te Ziffer des Schlüssels
- MSD-Sortierung (most-significant digit):
  - Sortiere erst nach vorderster Ziffern, dann nach zweiter Ziffer, usw.
  - Quicksort-Verallgemeinerung: Partitionierung in viele Teilmengen
- LSD-Sortierung (least-significant digit):
  - Sortierung erst nach minderwertigster Ziffer, dann nach vorletzter Ziffer, usw.
  - unintuitiv: Wie hilft die Sortierung nach der letzten Stelle zur Lösung des Gesamtproblems?

# MSD-Sortierung

- Aufteilung der Gesamtmenge in Teilmengen nach der ersten Ziffer:
  - Array von  $\mathbf{R}$  Körben (*bins, buckets*)
  - Elemente werden in passenden Korb eingefügt
- Rekursive Aufteilung jedes Korbs nach zweiter, dritter, usw. Stelle
- Übergang zu alternativem Algorithmus, wenn Zahl der verbleibenden Elemente klein ist (etwa: kleiner als  $\mathbf{R}$ )

# MSD-Sortierung: Speicherung der Körbe

- Idee: Ein Array für alle Körbe
  - Erst alle Elemente mit Ziffer 0, dann alle Elemente mit Ziffer 1 usw.
- Partitionierungsinformation: Ein Array für den Anfangsindex jeder Korbes
- Zwei Pässe:
  - Zählung der Elemente in gleicher Ziffer, um die Korbgrößen zu bestimmen (und damit die Anfangsindizes)
  - Aufteilung der Elemente in die Körbe: Array für die Speicherung der aktuellen Position in jedem Korb

# LSD-Sortierung

- Sortierung zuerst nach letzter Ziffer (Position  $L$ ), dann nach vorletzter, usw.
- historisches Verfahren, verwendet z.B. zur Sortierung von Lochkarten (Nummerierung der Karten in den letzten Spalten)
  - allgemein: praktikabel für Sortierung von ganzen Zahlen, nicht anwendbar auf Strings variabler Länge
- Verfahren sortiert nur, wenn jeder Sortierschritt **stabil** ist:
  - Beweis induktiv: Nach Sortierung von Position  $i$  ist die Menge bezüglich des Teilstrings  $i, i+, \dots, L$  sortiert

# Leistungsbewertung

- Vergleich mit anderen Sortierverfahren schwierig, weil Radixsort explizit Schlüssellänge berücksichtigt
  - bei anderen Algorithmen üblicherweise Annahme, dass Schlüsselvergleiche in konstanter Zeit stattfinden
- LSD: Sortierung von  $N$  Schlüsseln mit Länge  $w$  proportional zu  $Nw$ 
  - asymptotisch im Mittel etwa  $N \log N$ , weil Schlüssellänge von  $N$  Schlüsseln etwa  $\log N$  ist
- MSD: schlechtester Fall: Alle Ziffern aller Schlüssel müssen berücksichtigt werden
  - z.B. falls alle Schlüssel gleich sind
- Sortierung oft *sublinear* bzgl. der Zahl der Schlüsselbits, weil nicht alle Bits aller Schlüssel untersucht werden müssen