Algorithmen und Datenstrukturen I

Mergesort

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 18.03.2018 18:16

Inhaltsverzeichnis

Mergesort

- Sortieren durch Verschmelzen
- eines der ältesten und am Besten untersuchten Sortierverfahren
  - bereits 1945 von John von Neumann vorgeschlagen
- wie Quicksort rekursiv
  - die rekursiv sortierten Teilfolgen werden dann Verschmolzen
- es wird linear viel zusätzlicher Speicherplatz verwendet
- aber Laufzeit kann $\mathcal{O}(NlogN)$ nicht überschreiten

2-Wege-Mergesort

- $F = k_1, ..., k_N$ wird sortiert,
- indem F in zwei möglichst gleich größe Teilfolgen $F_1 = k_1, ..., k_{\lceil \frac{N}{2} \rceil}$ und $F_2 = k_{\lceil \frac{N}{2} \rceil + 1}, ..., k_N$
- dann wird F_1 und F_2 mit Mergesort sortiert
- die sortierte Folge ergibt sich durch Verschmelzen der beiden
- Verschmelzen bedeutet, beide Folge zu durchlaufen und jeweils das kleinere Element in die Resultatliste zu übernehmen

Mergesort in C++

```
1  void mergesort(int *a, int l, int r) {
2      if (l < r) {
3          int m = (l + r) / 2;
4          mergesort(a, l, m);
5          mergesort(a, m + 1, r);
6          merge(a, l, m, r);
7      }
8  }
```

Merge in C++

```
1  void merge(int *a, int l, int m, int r) {
2      int b[r]; // Hilfsfeld
3      int i = l; int j = m + 1; int k = l;
4      while (i <= m && j <= r) {
5          if (a[i] < a[j]) {
6              b[k] = a[i];
7              i++;
8          } else {
9              b[k] = a[j];
10             j++;
11         }
12         k++;
13     }
14     if (i > m)
15         for (int h = j; h <= r; h++) b[k + h - j] = a[h];
16     else
17         for (int h = i; h <= m; h++) b[k + h - i] = a[h];
18     for (int h = l; h <= r; h++) a[h] = b[h];
19 }
```

Reines 2-Wege-Mergesort

- es wird sortiert indem Teilfolgen zu immer längeren Teilfolgen verschmolzen werden
- am Anfang ist jeder Schlüssel eine sortierte Teilfolge
- in einem Durchgang werden jeweils 2 benachbarte Teilfolgen miteinander verschmolzen, d.h. im ersten Schritt k_1 mit k_2, k_3 mit k_4,...

Reines 2-Wege-Mergesort in C++

```
1  void straightmergesort(int *a, int l, int r) {
2      int size = 1;
3      while (size < r - l + 1) {
4          int rr = l - 1;
5          while (rr + size < r) {
6              int ll = rr + 1;
7              int mm = ll + size - 1;
8              rr = mm + size <= r ? mm + size : r;
9              merge(a, ll, mm, rr);
10         }
11         size *= 2;
12     }
13 }
```

- Schlüsselvergleiche und Bewegungen nur in `merge`
- höchstens $logN$ Durchgänge

Natürliches 2-Wege-Mergesort

- ausgehend vom reinen 2-Wege-Mergesort wird nicht mit einelementigen Teilfolgen begonnen,
- sondern gleich möglichst lange, bereits sortierte Teilfolgen genommen

Natürliches 2-Wege-Mergesort in C++

```
1  void naturalmergesort(int *a, int l, int r) {
2    int ll;
3    do {
4      int rr = l - 1;
5      while (rr < r) {
6        ll = rr + 1;
7        int mm = ll;
```

```
8         while (mm < r && a[mm + 1] >= a[mm]) mm++;
9         if (mm < r) {
10          rr = mm + 1;
11          while (rr < r && a[rr + 1] >= a[rr]) rr++;
12          merge(a, ll, mm, rr);
13        } else
14          rr = mm;
15      }
16    } while (ll != l);
17  }
```