Algorithmen und Datenstrukturen

Die im nachfolgenden benutzte Variable $\overline{S}_e$ bezeichnet die Erfolgreiche Suche und die Variable $\overline{S}_n$ die erfolglose Suche. Ordnungskriterien werden mit **O_i**, Schlüsselwerte mit **k_i** , Vergleiche mit **V** und Bewegungen mit **B** bezeichnet.

Sequentielle (sukzessive) Suche

Mittlere Anzahl von Suchschritte bei erfolgreicher Suche: $\overline{S}_e = \sum_{i=1}^{n} P_i \cdot i$

(P_i := Suchwahrscheinlichkeit nach dem Datenobjekt d_i, n := Anzahl Datenobjekte)

Wenn $P_i = \frac{1}{n}$ → Erfolgreiche Suche: $\overline{S}_e = \frac{1}{n}\sum_{i=1}^{n} i = \frac{n+1}{2}$

Erfolglose Suche: $\overline{S}_n = n$

m-Wege-Suche (Sprungsuche)

$\overline{S}_e = \frac{g+1}{2} + \frac{n_g+1}{2}$ $\quad n = n_g \cdot g$

Optimale Gruppengröße: $n_g = \sqrt{n}$ → $g = \sqrt{n}$ → $\overline{S}_{e_{opt}} = \sqrt{n}+1$

Erfolglose Suche: $2 \le S_n \le g + n_g$

Binäre Suche

Erfolgreiche Suche: $\overline{S}_e \approx ld\,n - 1$

Erfolglose Suche: $\lfloor ld\,n \rfloor \le \overline{S}_n \le \lfloor ld\,n \rfloor + 1$

Interpolierende Suche

Erfolgreiche / Erfolglose Suche: $\overline{S}_e = \overline{S}_n \approx ld(ld(n+1))$

Linear gekettete Liste

Erfolgreiche Suche: $\overline{S}_e = \frac{n+1}{2}$

Erfolglose Suche: $\overline{S}_n = n$

Gleichmäßiges Hashing

(In der Praxis nicht erfüllbar bzw. nur näherungsweise mit Doppel-Hashing)

Erfolgreiche Suche: $\overline{S}_e \approx -\frac{1}{\beta}\ln(1-\beta)$

Erfolglose Suche: $\overline{S}_n \approx \frac{1}{1-\beta}$ $\quad n >> 1$

Lineare Sondierung

Erfolgreiche Suche: $\overline{S}_e \approx \frac{1}{2}(1+\frac{1}{1-\beta})$

Erfolglose Suche: $\overline{S}_e \approx \frac{1}{2}(1+\frac{1}{(1-\beta)^2})$ $\quad n >> 1$

Insertion Sort (Sortieren durch direktes Einfügen)

Beim i-ten Durchlauf ergeben sich max i Vergleiche und minimal 1 Vergleich. Im Mittel ergibt das $\frac{i+1}{2}$ Vergleiche. Die Anzahl der Bewegungen ist jeweils um 2 größer (nämlich für x:= A[i] und A[0]). Insgesamt für 1=2(1)n:

$$V_{\min} = \sum_{i=2}^{n} 1 = n-1 \quad \rightarrow \quad O(n)$$

$$V_{\max} = \sum_{i=2}^{n} i = \sum_{i=1}^{n} i - 1 = \frac{n}{2}(n+1) - 1 = \frac{1}{2}(n^2+n) - 1 = \frac{1}{2}(n^2+n) - 1 \quad \rightarrow \quad O(n^2)$$

$$V_{mitt} = \sum_{i=2}^{n} \frac{i+1}{2} = \frac{1}{2}\sum_{i=2}^{n} i + \frac{1}{2}\sum_{i+2}^{n} 1 = \frac{1}{4}(n^2+n) - \frac{1}{2} + \frac{1}{2}(n-1) = \frac{1}{4}(n^2+3n) - \frac{3}{2} \quad \rightarrow \quad O(n^2)$$

$$B_{\min} = \sum_{i=2}^{n} 3 = 3(n-1)$$

$$B_{\max} = \sum_{i=2}^{n} (i+2) = \sum_{i=2}^{n} i + 2\sum_{i=2}^{n} 1 = \frac{1}{2}(n^2+5n) - 3 \quad \rightarrow \quad O(n^2)$$

$$B_{mitt} = \sum_{i=2}^{n} \left(\frac{i+2}{2} + 2\right) = \frac{1}{4}(n^2+11n) - 3 \quad \rightarrow \quad O(n^2)$$

Selection Sort (Sortieren durch direkte Auswahl)

Unabhängig von einer (teilweisen) gegebene Vorsortierung gilt:

$$V = \frac{1}{2}(n^2-n) \quad \rightarrow \quad O(n^2)$$

$$B = 3(n-1) \quad \rightarrow \quad O(n)$$

Bubble Sort (Sortieren durch direkten Austausch)

$$V = \frac{1}{2}(n^2-n) \quad \rightarrow \quad O(n^2)$$

$$B_{mitt} = \frac{3}{4}(n^2-n) \quad \rightarrow \quad O(n^2)$$

Shellsort (Sortieren durch Einfügen mit abnehmendem Abstand)

Eine mathematische Analyse ist bisher nicht gelungen. Die optimale Abstandsfolge ist auch unbekannt. Für H=3H+1 wird die Laufzeit proportional zu $n^{1,25}$ vermutet.
(H := jeweiliger Abstand)

Quicksort (Direkter Austausch von Elementen)

Wird aus der Mitte gegriffen, so ist es auch bei vorsortierten Fällen unwahrscheinlich, dass der größte bzw. kleinste Wert getroffen wird.

- Vergleiche: $V_{\min} \approx n \cdot ld\, n \quad \rightarrow \quad O(n \cdot ld\ n)$
  $V_{mitt} \approx 2n \cdot ld\, n \quad \rightarrow \quad O(n \cdot ld\ n)$
- Bewegungen: $B_{\min} = \frac{1}{2} n \cdot ld\, n \quad \rightarrow \quad O(n \cdot ld\ n)$
  $B_{\max} = \frac{1}{2} n^2 \quad \rightarrow \quad O(n \cdot ld\ n)$
  $B_{mitt} = n \cdot ld\, n \quad \rightarrow \quad O(n \cdot ld\ n)$

Heap Sort

Für das Erstellen des Heaps am Ort, werden $^{N}/_{2}$ Schritte mit O(ld $^{N}/_{2}$) benötigt. Für das Sortieren selbst werden N-1 Schritte mit maximal O (ld(N-1)) benötigt. Dazu kommen N-1 Bewegungen:

$B \approx \frac{n}{2} ld \left\lfloor \frac{n}{2} \right\rfloor + (n-1) ld(n-1) + n - 1$ → O(n · ld n)

$V = 2n \cdot ld\, n$ → O(n · ld n)

Merge Sort (sortieren durch (direktes) Mischen (mit paarweise gegenläufigen Tupeln))

Bei jedem Durchlauf wird die Tupelgröße verdoppelt, d.h. es werden $\lfloor ld\, N \rfloor + 1$ Durchläufe benötigt. In jedem Durchlauf werden alle Elemente kopiert, d.h.:

$B = O(N \cdot ld\, N)$ bzw. $B \approx N \cdot ld\, N$ → Speicherplatz: 2N

Einfacher binärer Sortierbaum