

Algorithmen und Datenstrukturen

Die im nachfolgenden benutzte Variable \bar{S}_e bezeichnet die Erfolgreiche Suche und die Variable \bar{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 Suchschritten bei erfolgreicher Suche: $\bar{S}_e = \sum_{i=1}^n P_i \cdot i$

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

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

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

m-Wege-Suche (Sprungsuche)

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

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

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

Binäre Suche

Erfolgreiche Suche: $\bar{S}_e \approx \lg n - 1$

Erfolglose Suche: $\lfloor \lg n \rfloor \leq \bar{S}_n \leq \lfloor \lg n \rfloor + 1$

Interpolierende Suche

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

Linear gekettete Liste

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

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

Gleichmäßiges Hashing

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

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

Erfolglose Suche: $\bar{S}_n \approx \frac{1}{1-\beta}$ $n \gg 1$

Lineare Sondierung

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

Erfolglose Suche: $\bar{S}_n \approx \frac{1}{2} \left(1 + \frac{1}{(1-\beta)^2}\right)$ $n \gg 1$

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 $1 \leq i \leq n$:

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

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

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

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

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

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

Selection Sort (Sortieren durch direkte Auswahl)

Unabhängig von einer (teilweisen) gegebenen 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)

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

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

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: $V_{\min} \approx n \cdot \ln n \quad \rightarrow \quad O(n \cdot \ln n)$

$$V_{\text{mitt}} \approx 2n \cdot \ln n \quad \rightarrow \quad O(n \cdot \ln n)$$

○ Bewegungen: $B_{\min} = \frac{1}{2}n \cdot \ln n \quad \rightarrow \quad O(n \cdot \ln n)$

$$B_{\max} = \frac{1}{2}n^2 \quad \rightarrow \quad O(n \cdot \ln n)$$

$$B_{\text{mitt}} = n \cdot \ln n \quad \rightarrow \quad O(n \cdot \ln n)$$

Heap Sort

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

$$B \approx \frac{n}{2} \text{ld} \left\lfloor \frac{n}{2} \right\rfloor + (n-1) \text{ld}(n-1) + n - 1 \quad \rightarrow \quad O(n \cdot \text{ld } n)$$

$$V = 2n \cdot \text{ld } n \quad \rightarrow \quad O(n \cdot \text{ld } n)$$

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

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

$$B = O(N \cdot \text{ld } N) \text{ bzw. } B \approx N \cdot \text{ld } N \quad \rightarrow \quad \text{Speicherplatz: } 2N$$

Einfacher binärer Sortierbaum