

ADS: Algorithmen und Datenstrukturen

Teil XII

Peter F. Stadler & Konstantin Klemm

Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for
Bioinformatics, **University of Leipzig**

19. Januar 2011

Analyse des Hashing: Kostenmaße

- $\beta = n/m$: Belegung von HT mit n Schlüsseln
- $S(\beta)$ = Anzahl Suchschritte für das Auffinden eines Schlüssels
- entspricht den Kosten für erfolgreiche Suche und Löschen
(ohne Reorganisation)
- $U(\beta)$ = Anzahl Suchschritte für die erfolglose Suche — das
Auffinden des ersten freien Platzes — entspricht den
Einfügekosten

		best case:	worst case:
Grenzwerte:	$S(\beta) =$	1	n
	$U(\beta) =$	1	$n + 1$

Analyse: Lineares Sondieren

- Sobald β eine gewisse Größe überschreitet, verschlechtert sich das Zugriffsverhalten sehr stark.
- Je länger eine Liste ist, umso schneller wird sie noch länger.
- Zwei Listen können “zusammenwachsen”, so dass durch neue Schlüssel eine Art Verdopplung der Listenlänge eintreten kann
- Ergebnisse für das lineare Sondieren nach Knuth:

$$S(\beta) \approx \frac{1}{2} \left(1 + \frac{1}{1-\beta} \right), \quad U(\beta) \approx \frac{1}{2} \left(1 + \frac{1}{(1-\beta)^2} \right)$$

mit $0 \leq \beta = \frac{n}{m} < 1$.

Analyse: uniformes Sondieren, erfolglose Suche

Annahme: Sondierungsfolgen zu verschiedenen Schlüsseln sind unkorreliert (zufällig). *Kumulierte* Wahrscheinlichkeit, daß erfolglose Suche *mindestens* i Schritte braucht:

$$q_i = \frac{n}{m} \frac{n-1}{m-1} \frac{n-2}{m-2} \cdots \frac{n-i+2}{m-i+2} \leq \left(\frac{n}{m}\right)^{i-1} = \beta^{i-1}$$

Erwartungswert ist Summe der kumulierten Wahrscheinlichkeiten:

$$\begin{aligned} U(\beta) &= \sum_{i=1}^{\infty} q_i \\ &\leq \sum_{i=1}^{\infty} \beta^{i-1} = \sum_{i=0}^{\infty} \beta^i \\ &= \frac{1}{1-\beta} \end{aligned}$$

Analyse: uniformes Sondieren, erfolgreiche Suche

Bei erfolgreicher Suche wird dieselbe Sondierungsfolge wie beim Einfügen des Schlüssels durchlaufen. Für den als $(i + 1)$ -tes eingefügten Schlüssel entspricht dies einer erfolglosen Suche in einer HT der Länge m mit i Schlüsseln. Kosten hierfür

$$u_i = \frac{1}{1 - \frac{i}{m}} = \frac{m}{m - i}$$

Mittelwert über alle n Schlüssel

$$\begin{aligned} S(\beta) &= \frac{1}{n} \sum_{i=0}^{n-1} u_i = \frac{m}{n} \sum_{i=0}^{n-1} \frac{1}{m - i} = \beta^{-1} \sum_{i=m-n-1}^m \frac{1}{i} \\ &\leq \beta^{-1} \int_{i=m-n}^m \frac{1}{i} di = \beta^{-1} \ln \frac{m}{m - n} \\ &= \frac{1}{\beta} \ln \frac{1}{1 - \beta} \end{aligned}$$

Analyse: Verkettung, erfolglose Suche

- Annahme: n Schlüssel verteilen sich gleichförmig über die m möglichen Ketten.
- Jede Synonymkette hat also im Mittel $n/m = \beta$ Schlüssel.

$$U(\beta) = 1 + \beta$$

Analyse: Verkettung, erfolgreiche Suche

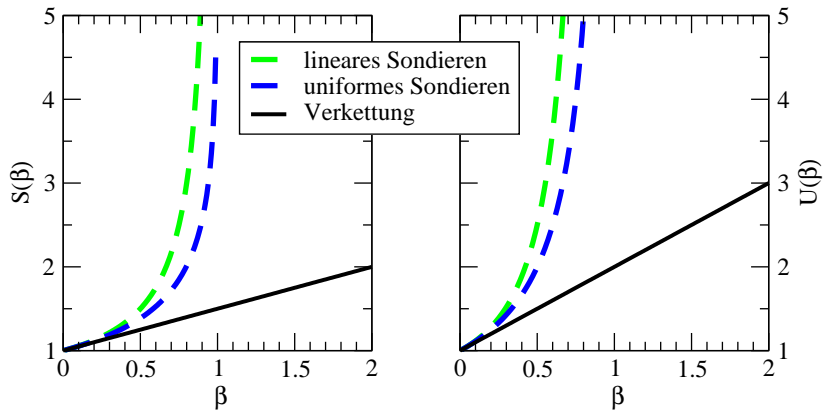
- Sondierungsfolge bei erfolgreicher Suche von $k =$
Sondierungsfolge beim Einfügen von k
- Beim Einfügen des $(i + 1)$ -ten Schlüssels sind i Schlüssel in der HT. Von diesen im Mittel i/m in derselben Kette wie k .

$$S(\beta) = \frac{1}{n} \sum_{i=1}^n \left[1 + \frac{i-1}{m} \right] = 1 + \frac{1}{nm} \frac{n(n-1)}{2} \quad (1)$$

$$= 1 + \frac{n-1}{2m} \quad (2)$$

$$< 1 + \frac{\beta}{2} \quad (3)$$

Analyse des Hashing: Vergleich



⇒ Verkettung hat deutlich geringeren Zeitaufwand als offenes Hashing für Belegungsfaktor β nahe 1

Hashing auf Externspeichern I

Hash-Adresse bezeichnet Bucket (hier: Seite)

- Kollisionsproblem wird entschärft, da mehr als ein Satz auf seiner Hausadresse gespeichert werden kann
- Bucket-Kapazität $b \rightarrow$ Primärbereich kann bis zu bm Sätze aufnehmen.

Überlaufbehandlung

- Überlauf tritt erst beim $(b + 1)$ -ten Synonym auf
- alle bekannten Verfahren sind möglich, aber lange Sondierungsfolgen im Primärbereich sollten vermieden werden
- häufig Wahl eines separaten Überlaufbereichs mit dynamischer Zuordnung der Buckets

Speicherungsreihenfolge im Bucket

- ohne Ordnung (Einfügefølge)
- nach der Sortierfolge des Schlüssels: aufwendiger, jedoch Vorteile beim Suchen (sortierte Liste!)

Hashing auf Externspeichern II

Bucket-Größe meist Seitengröße (Alternative: mehrere Seiten / Spur einer Magnetplatte)

- Zugriff auf die Hausadresse bedeutet 1 physische E/A
- jeder Zugriff auf ein Überlauf-Bucket löst weiteren physischen E/A-Vorgang aus

Bucket-Adressierung mit separaten Überlauf-Buckets

- weithin eingesetztes Hash- Verfahren für Externspeicher
- jede Kollisionsklasse hat eine separate Überlaufkette.

Hashing auf Externspeichern III

Grundoperationen

- direkte Suche: nur in der Bucket-Kette
- sequentielle Suche ?
- Einfügen: ungeordnet oder sortiert
- Löschen: keine Reorganisation in der Bucket-Kette - leere Überlauf-Buckets werden entfernt

Dynamische Hash-Verfahren I

Wachstumsproblem bei statischen Verfahren

- Statische Allokation von Speicherbereichen:
Speicherausnutzung?
- Bei Erweiterung des Adressraumes: Re-Hashing \Rightarrow Alle Sätze erhalten eine neue Adresse
- Probleme: Kosten, Verfügbarkeit, Adressierbarkeit

Dynamische Hash-Verfahren II

Entwurfsziele

- Eine im Vergleich zum statischen Hashing dynamische Struktur, die Wachstum und Schrumpfung des Hash-Bereichs (Datei) erlaubt
- Keine Überlauftechniken

Erweiterbares Hashing

- Die einzelnen Bits eines Schlüssels steuern der Reihe nach den Weg durch den zur Adressierung benutzten Digitalbaum
 $K_i = (b_0, b_1, b_2, \dots)$
- Verwendung der Schlüsselwerte kann bei Ungleichverteilung zu unausgewogenem Digitalbaum führen (Digitalbäume kennen keine Höhenbalancierung!)
- Verwendung von $h(K_i)$ als sog. Pseudoschlüssel (PS) soll bessere Gleichverteilung gewährleisten. $h(K_i) = (b_0, b_1, b_2, \dots)$
- Digitalbaum-Adressierung bricht ab, sobald ein Bucket den ganzen Teilbaum aufnehmen kann.

Beispiel → Tafel