Löschungen in zufälligen binären Suchbäumen
Eine Geschichte der Irrungen

Wolfgang Panny
Institut für Informationsverarbeitung und Informationswirtschaft
Wirtschaftsuniversität Wien
Augasse 2-6
A-1090 Wien
wolfgang.panny@wu-wien.ac.at

Abstract: Die üblichen Annahmen für die average-case Analyse von binären Suchbäumen (BSB) sind zufällige Einfügungen (random insertions) und zufällige Löschungen (random deletions). In einem durch zufällige Einfügungen aufgebauten BSB haben die Zugriffsoperationen eine erwartete Zeitkomplexität von $O(\log n)$ und schon die ersten Publikationen über BSB enthalten wesentliche analytische Ergebnisse für solche 'random' BSB. Wenn es allerdings um zufällige Löschungen und um ihre Interaktion mit zufälligen Einfügungen geht, scheint die Analyse um einiges komplizierter zu werden. Das kann man jedenfalls den einschlägigen Publikationen seit 1962 entnehmen, die es berechtigt erscheinen lassen, in diesem Zusammenhang von einer 'Geschichte der Irrungen' zu sprechen. Dieser Beitrag geht etwas näher auf diese Geschichte ein.

1 Einführung und Grundlagen

Binäre Suchbäume (BSB) gehören zu den prominentesten und am häufigsten eingesetzten Datenstrukturen für Symboltabellen-Algorithmen [Knu98, 426]. Die üblichen Such- und Einfügeroutinen für BSB (vgl. etwa [Knu98, 429]) eignen sich sehr gut für diesen Zweck, besonders wenn neben der Suche bzw. dynamischen Einfügung von 'Symbolen' auch die effiziente lineare Abarbeitung der Symbole entsprechend ihrer Sortierordnung möglich sein soll, beispielsweise zur Ausgabe einer sortierten Liste der Symbole. Die ersten Publikationen über binäre Suchbäume stammen von *P. F. Windley* [Win60], *A. D. Booth* und *A. J. T. Colin* [BC60] und *T. N. Hibbard* [Hib62]. Jede dieser (voneinander unabhängigen) Arbeiten enthält eine Beschreibung des Einfügens und Suchens in BSB und der dabei zu erwartenden Anzahl der Vergleiche.[1]

Hibbard hat als erster gezeigt, dass auch Löschungen ohne größere Schwierigkeiten realisiert werden können [Hib62], womit sich der natürliche Anwendungsbereich von BSB beträchtlich erweitert. In [Knu98, 471–475] wird dargestellt, wie durch eine kleine Mo-

[1] *Windley*'s Paper enthält auch eine ausführliche Diskussion von 'Tree Insertion Sorting'. *Booth* und *Colin* schlagen schon eine Methode zur Balancierung vor. Als "Vorläufer" werden auch *A. I. Dumey* [Knu98, 435], *D. J. Wheeler* und *C. M. Berners-Lee* genannt [Dou59, 5], [Win60, 84].

difikation der Datenstruktur bewerkstelligt werden kann, dass die BSB-Operationen nicht nur ‘by key’ sondern auch ‘by rank’ durchgeführt werden können, und wie BSB durch zusätzliche Erweiterungen zu einer äußerst vielseitigen und worst-case robusten Datenstruktur zur effizienten Manipulation von ‘linearen Listen’ ausgebaut werden können.

In diesem Beitrag wollen wir uns aber auf die klassische Datenstruktur mit den Zugriffsfunktionen: Suchen, Einfügen und Löschen beschränken. Die übliche (und durchaus vernünftige) Annahme für die average-case Analyse von BSB stellen *zufällige Einfügungen (random insertions)* dar. In einem durch zufällige Einfügungen aufgebauten BSB haben die Zugriffsoperationen eine erwartete Zeitkomplexität von $O(\log n)$ und wesentliche analytische Ergebnisse dazu waren schon in den ersten Arbeiten [Win60], [BC60], [Hib62] enthalten. Wenn es allerdings um *zufällige Löschungen (random deletions)* und um ihre Interaktion mit zufälligen Einfügungen geht, scheint die Analyse noch komplizierter zu werden. Das kann man jedenfalls den einschlägigen Publikationen seit 1962 entnehmen, die es berechtigt erscheinen lassen, in diesem Zusammenhang von einer ‘Geschichte der Irrungen’ zu sprechen. In diesem Beitrag soll auf diese Geschichte etwas näher eingegangen werden. Bevor das im Teil 2 geschieht, sollen in den folgenden Abschnitten zunächst die notwendigen begrifflichen und notationellen Grundlagen zusammengestellt werden.

1.1 Binäre Suchbäume und Zugriffsoperationen

Ein *Binärbaum (binary tree)* kann folgendermaßen (rekursiv) definiert werden [Knu97]: Ein Binärbaum umfasst eine endliche Menge von Knoten. Diese ist entweder leer (in diesem Fall spricht man von einem leeren Binärbäum) oder sie besteht aus einem Wurzelknoten und den Knoten im linken und im rechten Unterbaum, welche wieder Binärbäume sind.

Aus obiger Definition folgt, dass es in jedem Binärbaum mindestens einen leeren Unterbaum geben muss. Manchmal ist es zweckmäßig diese leeren Unterbäume als spezielle zusätzliche Knoten zu betrachten, die man dann *externe* Knoten nennt. Einen durch externe Knoten ergänzten Binärbaum nennt man einen *erweiterten* Binärbaum [Knu97, 399]. Die “gewöhnlichen” Knoten nennt man dann *interne* Knoten. Ein Binärbaum mit n (internen) Knoten hat $n+1$ externe Knoten. Im erweiterten Binärbaum sind die Blattknoten mit den externen Knoten identisch.

Ein *binärer Suchbaum (binary search tree)* ist ein Binärbaum, bei dem jedem Knoten ein Schlüssel[2] zugeordnet ist, so dass die folgende *Suchbedingung* erfüllt ist: Der leere BSB erfüllt die Suchbedingung. Ansonsten muss für jeden Knoten gelten, dass alle Schlüssel in seinem linken (rechten) Unterbaum kleiner (größer) sind, als der dem Knoten zugeordnete Schlüssel.

Die Realisierung der *Suchoperation* ergibt sich unmittelbar aus der Suchbedingung. Wenn der gesuchte Schlüssel existiert, spricht man von einer *erfolgreichen* Suche, ansonsten von einer *erfolglosen* Suche.

[2] Die Schlüssel sind Elemente einer vollständig geordneten Menge.

Auch die Realisierung der *Einfügeoperation* folgt unmittelbar aus der Suchbedingung: Der eigentlichen Einfügung geht eine Suche nach dem einzufügenden Schlüssel voraus. Diese Suche endet als erfolglose Suche bei einem leeren Unterbaum. Der einzufügende Knoten wird anstelle dieses leeren Unterbaums eingefügt (mit einem leeren linken und einem leeren rechten Unterbaum).

Zur Beschreibung der *Löschoperation* ist die folgende Notation hilfreich: Sei T ein nichtleerer BSB. Sei $v \in T$ ein Knoten von T. Dann symbolisieren wir den linken bzw. rechten Unterbaum von v durch $\ell(v)$ bzw. $r(v)$. $f(v)$ ist der Vaterknoten von v, der durch $f(\ell(v)) = f(r(v)) = v$ charakterisiert werden kann (der Einfachheit halber sei hier angenommen, dass auch T einen Vater w hat, wobei $\ell(w) = T$). Der Schlüssel von v wird durch $k(v)$ symbolisiert. Vor der eigentlichen Löschung muss der zu löschende Knoten gesucht werden. Diese Suche muss erfolgreich enden. Sei v der zu löschende Knoten. Für die Löschoperation gemäß *Hibbard* [Hib62, 24] müssen die folgenden zwei Fälle unterschieden werden:

a) $r(v) = \emptyset$: Dann überschreibe die Referenz auf v in $f(v)$ mit $\ell(v)$.

b) $r(v) \neq \emptyset$: Dann lösche den Knoten v_{min} mit dem kleinsten Schlüssel[3] aus $r(v)$ und kopiere $k(v_{min})$ auf v.

Knuth löst in seiner Weiterentwicklung aus Hibbard's Fall b) den Unterfall a') heraus,[4] der jetzt speziell behandelt wird [Knu97, 431]:

a) $r(v) = \emptyset$: Dann überschreibe die Referenz auf v in $f(v)$ mit $\ell(v)$.

a') $r(v) \neq \emptyset$ und $\ell(v) = \emptyset$: Dann überschreibe die Referenz auf v in $f(v)$ mit $r(v)$.

b') $r(v) \neq \emptyset$ und $\ell(v) \neq \emptyset$: Dann lösche den Knoten v_{min} mit dem kleinsten Schlüssel aus $r(v)$ und kopiere $k(v_{min})$ auf v.

Die beiden Löschverfahren weisen somit nur dann ein verschiedenes Verhalten auf, wenn der linke Unterbaum leer und der rechte Unterbaum nicht leer ist.

1.2 Zufällige Einfügungen, zufällige Löschungen und Randomness

Häufig wird bei *zufälligen Einfügungen (random insertions)* davon ausgegangen, dass die n Schlüssel zufällig und unabhängig voneinander aus einer $[0, 1]$ Gleichverteilung gezogen werden. Tatsächlich kann jede beliebige stetige Verteilung verwendet werden, da es nur darauf ankommt, dass jede der $n!$ Permutationen die gleiche Wahrscheinlichkeit hat (vgl. [Knu77]). Für analytische Zwecke werden deshalb der Einfachheit halber oft Zufallspermutationen der natürlichen Zahlen $\{1, 2, \ldots, n\}$ genommen. *H. M. Mahmoud* [MR03,

[3] Da für v_{min} immer $\ell(v_{min}) = \emptyset$ gelten muss, kann v_{min} — analog zu Fall a) — durch Überschreiben der Referenz auf v_{min} in $f(v_{min})$ durch $r(v_{min})$ gelöscht werden.

[4] Knuth nennt den entsprechenden Abschnitt in seinem Algorithmus D den 'step D1$\frac{1}{2}$'.

254] spricht in diesem Zusammenhang treffend vom '*random permutation model* of randomness'.

Für einen BSB mit n Knoten gibt es insgesamt $\binom{2n}{n}/(n+1)$ verschiedene Gestalten (shapes). Wenn ein BSB durch n zufällige Einfügungen aufgebaut wird (was durch I^n symbolisiert wird), hat jede Gestalt eine bestimmte Wahrscheinlichkeit, es resultiert also eine bestimmte Verteilung F_n der Gestalten. Wenn in einem durch I^n aufgebauten BSB eine weitere zufällige Einfügung vorgenommen wird, resultiert die Verteilung F_{n+1} der Gestalten. Binäre Suchbäume, bei denen die Verteilung der Gestalten F_n entspricht, werden *zufällige binäre Suchbäume (random binary search trees)* genannt. Im Fachjargon sagt man auch 'der BSB ist random'.

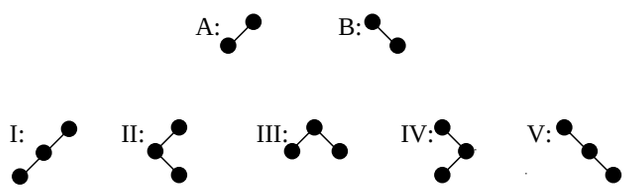

Abbildung 1: Gestalten der BSB mit $n = 2$ und $n = 3$ Knoten

Beispiel: Abbildung 1 enthält alle Gestalten für $n = 2$ und $n = 3$. Unter Randomness betragen die Wahrscheinlichkeiten für diese Gestalten $\mathbf{p}_2 = (1/2,\ 1/2)$ beziehungsweise $\mathbf{p}_3 = (1/6,\ 1/6,\ 1/3,\ 1/6,\ 1/6)$.

Eine *zufällige Löschung (random deletion)* ist so definiert, dass für jeden im BSB vorhandenen Knoten die Löschwahrscheinlichkeit gleich ist (wenn der BSB n Knoten hat, beträgt diese Wahrscheinlichleit also $1/n$). Manchmal werden Folgen von zufälligen Einfügungen und Löschungen betrachtet. Eine solche Folge kann durch ein Wort w aus I's und D's symbolisiert werden. Beispielsweise steht $IIIDI$ für 3 zufällige Einfügungen, gefolgt von einer zufälligen Löschung und einer weiteren zufälligen Einfügung. Es versteht sich von selbst, dass die Anzahl der D's in einem solchen Wort w und in jedem Präfix von w höchstens so groß wie die Anzahl der I's sein darf.

1.3 Kennzahlen zur Charakterisierung der Sucheffizienz von BSB

Sei T ein BSB und sei v ein Knoten von T. Wir bezeichnen die Anzahl der Kanten auf dem Pfad von v zum Wurzelknoten von T als *Tiefe*[5] $d(v)$ des Knotens v. Die Größe

$$I(T) = \sum_{i=1}^{n} d(v_i) \tag{1}$$

[5]Knuth würde in diesem Zusammenhang vom '*level*' des Knotens sprechen [Knu97, 308] .

heißt *interne Pfadlänge* von T, wobei über alle internen Knoten v_1, v_2, ..., v_n von T summiert wird. Entsprechend kann man auch eine *externe Pfadlänge* $E(T)$ definieren, wobei nun über die $n+1$ externen Knoten von T zu summieren ist. Es ist nicht schwer, sich zu überlegen, dass der Zusammenhang

$$E(T) = I(T) + 2n \tag{2}$$

besteht. Eine wichtige Kennzahl zur Charakterisierung der Sucheffizienz eines BSB stellt die *mittlere Anzahl von Vergleichen $C(T)$ bei der erfolgreichen Suche* dar. Da die Tiefe eines internen Knotens um genau 1 kleiner ist, als die Anzahl der zu seiner Lokalisierung erforderlichen Vergleiche, ist

$$C(T) = \frac{I(T)}{n} + 1. \tag{3}$$

Da andererseits die Anzahl der Vergleiche bei der erfolglosen Suche der Tiefe des externen Knotens entspricht, in dessen Intervall der gesuchte Schlüssel fällt, erhält man für die *mittlere Anzahl von Vergleichen $C'(T)$ bei der erfolglosen Suche*

$$C'(T) = \frac{E(T)}{n+1}. \tag{4}$$

Mit (2) und (3) erhält man daraus die Beziehung

$$C(T) = \left(1 + \frac{1}{n}\right) C'(T) - 1. \tag{5}$$

Es ist klar, dass die Größen $I(T)$, $E(T)$, $C(T)$ und $C'(T)$ äquivalent sind: Wenn der Wert einer dieser Größen bekannt ist, sind auch die Werte der drei übrigen Größen determiniert.

Eine weitere Kennzahl, die zur Beurteilung des worst-case Verhaltens herangezogen werden kann, ist die *Tiefe* $d(T)$ des gesamten BSB

$$d(T) = \max\left\{d(v_i) \,\middle|\, i = 1, 2, \ldots, n\right\}, \tag{6}$$

die wir als maximale Tiefe der internen Knoten definieren wollen. $d(T)+1$ entspricht der im schlimmsten Fall notwendigen Anzahl von Vergleichen bei der (erfolgreichen oder erfolglosen) Suche in T.[6] Beispiel: Sei T_I ein Binärbaum der Gestalt I aus Abbildung 1. Dann erhält man $I(T_I) = 3$, $E(T_I) = 9$, $C(T_I) = 2$, $C'(T_I) = 9/4$ und $d(T_I) = 2$.

Die Erwartungswerte der eben besprochenen Größen für zufällige BSB sind von großem Interesse. Bezeichnen wir diese Erwartungswerte für random-BSB mit n Knoten durch I_n, E_n, C_n, C'_n und D_n. Unter Verwendung der *harmonischen* Zahlen $H_n = 1 + 1/2 + 1/3 + \ldots + 1/n$ kann man die ersten vier Erwartungswerte folgendermaßen ausdrücken

[6] $d(T)+1$ wird bei Knuth als *Höhe (hight)* von T bezeichnet (vgl. [Knu98, 459]).

(vgl. beispielsweise [Knu98, 431]):

$$I_n = 2(n+1)H_n - 4n = 1.386\, n \log_2 n + O(n) \quad (7)$$

$$E_n = 2(n+1)H_n - 2n = 1.386\, n \log_2 n + O(n) \quad (8)$$

$$C_n = 2\left(1 + \frac{1}{n}\right) H_n - 3 = 1.386 \log_2 n + O(1) \quad (9)$$

$$C'_n = 2H_{n+1} - 2 = 1.386 \log_2 n + O(1) \quad (10)$$

Beispiel: Für $n = 2$ erhält man $I_2 = 1$, $E_2 = 5$, $C_2 = 3/2$, $C'_2 = 5/3$ und für $n = 3$ $I_3 = 8/3$, $E_3 = 26/3$, $C_3 = 17/9$, $C'_3 = 13/6$.

Für die erwartete Tiefe D_n eines random-BSB ist nur das asymptotische Äquivalent

$$D_n = \alpha \ln n + O(\log \log n), \qquad \alpha \approx 4.31107\,04070\ldots \quad (11)$$

bekannt [DR95].

2 Die Geschichte der Irrungen

2.1 Hibbard's Theorem und eine unzulässige Verallgemeinerung

Wie bereits erwähnt war *Hibbard* der erste Autor, der neben der Such- und Einfügeoperation auch eine Löschoperation für binäre Suchbäume angibt. Alle drei Operationen werden in [Hib62] als ALGOL 60-Prozeduren formuliert.

Hibbard motiviert die BSB-Struktur übrigens als Kompromiss zwischen Such- und Update-Effizienz: Während ein geordnetes Array Such-optimal ist (worst- und average-case Zeitkomplexität von $O(\log n)$), hat es eine sehr geringe Update-Effizienz (worst- und average-case Zeitkomplexität von $O(n)$). Umgekehrt ist eine verkettete Liste Update-optimal, weil eine Einfügung bzw. Löschung nur mit geringen fixen Kosten verbunden ist, wobei nun aber die Suche eine worst- und average-case Komplexität von $O(n)$ hat. Der Kompromiss, den die BSB-Struktur demgegenüber bietet, besteht darin, dass der erwartete Aufwand für jede dieser Operationen nur von der Ordnung $O(\log n)$ ist.

Um das zu präzisieren führt Hibbard in seiner Arbeit die übliche (und vernünftige) Grundannahme für die *average-case* Analyse von binären Suchbäumen ein, nämlich die, dass jede Permutation der einzufügenden Schlüsselwerte die gleiche Wahrscheinlichkeit hat.[7] Ausgehend von diesem 'random permutation model of randomness' (vgl. Abschnitt 1.2) gibt Hibbard in seinem Theorem 1 die Resultate (9) und (10) für C_n bzw. C'_n an, wobei er diese Größen als 'mean internal search length' $\overline{l'}(n)$ beziehungsweise 'mean open search length' $\overline{l}(n)$ bezeichnet.

Hibbard untersucht auch die Frage, wie sich zufällige Löschungen auf die 'Randomness' eines BSB auswirken. Dabei sind *zufällige Löschungen* wie in Abschnitt 1.2 definiert: die

[7] Hibbard spricht in diesem Zusammenhang auch von *randomly constructed (binary) search trees.*

Löschwahrscheinlichkeit für jeden im BSB vorhandenen Knoten ist also gleich groß. Das entsprechende Ergebnis ist erstaunlich. In einer an [Knu98, 432] angelehnten Formulierung lautet *Hibbard's Theorem* folgendermaßen:

H: Wenn aus einem random-BSB ein Element mit dem Löschverfahren von Hibbard gelöscht wird, ergibt sich wieder ein random-BSB. ■

Für das richtige Verständnis von H sollte man daran denken, dass die 'Randomness' eines BSB eine Aussage über die Verteilung der Baumgestalten ist (vgl. Abschnitt 1.2). Wenn wir also eine Folge von n zufälligen Einfügungen gefolgt von einer zufälligen Löschung durch $I^n D$ symbolisieren, besagt H somit, dass die Verteilung der Gestalten der aus $I^n D$ resultierenden BSB identisch mit der aus I^{n-1} resultierenden Verteilung ist. Als unmittelbare Folgerung aus H erhält man die folgende Verallgemeinerung:

H': Wenn aus einem random-BSB mit m_1 Elementen m_2 Elemente ($m_2 < m_1$) mit dem Löschverfahren von Hibbard gelöscht werden, resultiert wieder ein random-BSB. ■

Das bedeutet, dass die Verteilung der Gestalten der aus $I^{m_1} D^{m_2}$ resultierenden BSB identisch mit der aus $I^{m_1 - m_2}$ resultierenden Verteilung ist. Diese Verallgemeinerung von H ist vollkommen in Ordnung. Vgl. dazu auch [Knu77, 355]. Daher ist es zumindest intuitiv einigermaßen verblüffend, dass die folgende über H' hinausgehende Verallgemeinerung (mit der die "Geschichte der Irrungen" beginnt) unzulässig ist:

H": *In general, therefore, we assert that, starting with an empty list and performing m_1 random insertions and m_2 random deletions in any order, if $m_1 - m_2 = n$ then*

$$C'_n = 2\left(\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n+1}\right) \qquad \textit{and} \qquad C_n = \frac{n+1}{n} C'_n - 1 \approx C'_n - 1.$$

Die obige Formulierung entspricht dem Original. Nur die dort verwendeten Symbole t und t' wurden durch die entsprechenden Symbole C_n und C'_n ersetzt. Hibbard spricht übrigens von einer *empty list* (statt eines *empty (binary search) tree*, wie man erwarten würde), weil er seine Implementierung der BSB-Datenstruktur wegen der jeweils zwei Pointer zum linken bzw. rechten Teilbaum als *doubly linked list* bezeichnet. Bezüglich der 'm_1 *random insertions and* m_2 *random deletions in any order*' versteht es sich von selbst, dass die Anzahl der Löschungen natürlich zu keinem Zeitpunkt die Anzahl der Einfügungen übersteigen darf. Im Grunde ist H" nur eine Konsequenz der als erwiesen angenommenen "Tatsache", dass eine solche *Folge von beliebig gemischten zufälligen Einfügungen und Löschungen wieder zu einem random-BSB führt.* Das ist aber nicht korrekt, was angesichts der (wahren) Prämissen, dass sowohl zufällige Updates der Bauart I^n als auch solche der Bauart $I^{m_1} D^{m_2}$ zu random-BSB führen, nicht leicht zu erkennen ist.

2.2 Knott deckt Hibbard's Irrtum auf, die Knott'sche Vermutung

Nach dem Erscheinen von Hibbard's Arbeit [Hib62] im Jahr 1962 hat die gesamte 'scientific community' die unzulässige Verallgemeinerung Hibbard's übersehen und sich der Illusion hingegeben, dass jede mögliche Folge von zufälligen Einfügungen und Löschungen

zu einem random-BSB führt. Selbst *D. E. Knuth* war sich beim Erscheinen der ersten Auflage von [Knu73a] dieser unzulässigen Verallgemeinerung noch nicht bewusst. [8] So kann man in [Knu73a, 429] lesen:

K.1: *Since Algorithm D*[9] *is quite unsymmetrical between left and right, it stands to reason that a long sequence of random deletions and insertions will make the tree get way out of balance, so that the efficiency estimates we have made will be invalid. But actually the trees do not degenerate at all!*

Der Grund warum (leider nur vermeintlich) '*the trees do not degenerate at all*', besteht natürlich darin, dass auch Knuth zu diesem Zeitpunkt noch an den Erhalt der 'Randomness' glaubte. Wie bereits in Abschnitt 1.1 dargestellt, hat Knuth das ursprüngliche Löschverfahren von Hibbard modifiziert. In [Knu73a, ex.6.2.2-14] wird gezeigt, dass das Löschverfahren von Knuth dem von Hibbard in dem folgenden Sinn überlegen ist: Sei T ein beliebiger BSB und seien T'_H bzw. T'_K die nach Löschung eines Knotens v gemäß Hibbard bzw. Knuth resultierenden BSB. Dann gilt für die internen Pfadlängen (vgl. 1.3) immer $I(T'_K) \leq I(T'_H)$. Tatsächlich gibt es genügend Fälle, in denen die strikte Ungleichung $I(T'_K) < I(T'_H)$ erfüllt ist. Das bedeutet, dass das Knuth'sche Verfahren bezüglich der Sucheffizienz der resultierenden BSB überlegen ist. Diese Lage der Dinge führt zu einer weiteren nicht zutreffenden Bemerkung in [Knu73a, 431f], die gewissermaßen wieder eine Auswirkung der unzulässigen Verallgemeinerung H" darstellt:

K.2: *Exercise 14 shows that Algorithm D with this extra step*[10] *always leaves a tree that is at least as good as the original Algorithm D, in the path-length sense, and sometimes the result is even better. Thus, a sequence of insertions and deletions using this modification of algorithm D will result in trees which are actually better than the theory of random trees would predict: the average computation time for search and insertion will tend to decrease as time goes on.*

Gemäß Knuth kommt *Gary D. Knott* das Verdienst zu, die nicht zulässige Verallgemeinerung von Hibbard als erster entdeckt zu haben. Knott war Ph.D. Student in Stanford. Der Titel seiner Doktorarbeit lautet *Deletions in Binary Storage Trees* (vgl. [Kno75]). Diese Arbeit wurde von Knuth betreut und ist im Jahre 1975 fertiggestellt worden. Knott hat diese Entdeckung schon im Jahr 1972 bei Vorarbeiten für seine Doktorarbeit gemacht (vgl. [Knu73b, 431], [Knu98, S.435]). Publiziert wurde seine Entdeckung aber erst im Jahr 1975 im Rahmen seiner Doktorarbeit [Kno75, 35] und — immer unter Hinweis auf die Urheberschaft von Knott — in [Knu73b, 2nd printing (1975), 431], [Knu77, 353] und [JK78, 302]. Beispielsweise findet man in [Knu73b, 2nd printing (1975), 431] die folgende Bemerkung:

K.3a: *Although Theorem H*[11] *is rigorously true, in the precise form we have stated it, it* cannot *be applied, as we might expect, to a sequence of deletions followed by insertions. The shape of the tree is random after deletions, but the relative distribution of values in a given tree shape may change, and it turns out that the first random insertion after deletion*

[8] Allerdings kommt Knuth, wie wir gleich sehen werden, auch ein entscheidender Anteil an der Aufdeckung des Hibbard'schen Irrtums zu.

[9] Das ist der Algorithmus des Hibbard'schen Löschverfahrens (vgl. Abschnitt 1.1).

[10] Nämlich mit dem Schritt D1$\frac{1}{2}$ der Knuth'schen Modifikation (vgl. Abschnitt 1.1).

[11] Das ist *Hibbard's Theorem*, also Theorem H aus Abschnitt 2.1 dieses Beitrags.

actually destroys *the randomness property on the shapes. This startling fact, first observed by Gary Knott in 1972, must be seen to be believed.*

In [Knu77, 353] gesteht Knuth freimütig ein, dass auch er nicht gegen den Hibbard'schen Irrtum gefeit war:

K.3b: *The I^*D_r property*[12] *might seem to be all that one needs to guarantee insensitivity to any number of deletions, when they are intermixed with insertions in any order. At least, many people (including the present author when writing the first edition of [Knu73a]) believed this, and the subtle fallacy in this reasoning was apparently first pointed out by G. D. Knott in his thesis [Kno75].*

In dieselbe Kerbe schlägt die folgende Passage aus [JK78, 302]:

K.3c: *However, Knott also discovered a surprising paradox: Although Hibbard's theorem establishes that $n+1$ random insertions followed by a random deletion produce a tree whose shape has the distribution of n random insertions, it does* not *follow that a subsequent random insertion yields a tree whose shape has the distribution of $n+1$ random insertions! For ten years it had been believed that Hibbard's theorem proved the stability of the algorithms under repeated insertions and deletions (cf. [Hib62, p.25], [Knu73a, first printing, pp 429–432]; the discovery of a subtle fallacy in this reasoning therefore came as a shock.*

Zur Widerlegung der unzulässigen Hibbard'schen Verallgemeinerung H" genügt ein Gegenbeispiel. Ein solches wird in [Knu77, 353], [Knu73b, 2nd printing (1975), ex. 6.2.2-15] und in [JK78, 302f] gegeben. Dazu wird die Verteilung der BSB-Gestalten für $n = 3$ (vgl. Abbildung 1) betrachtet, wie sie durch eine Sequenz von zufälligen Einfügungen und Löschungen der Bauart $IIIDI$ entsteht. Was die zufälligen Einfügungen anlangt, gibt es insgesamt $4!$ gleichwahrscheinliche Möglichkeiten, wie die einzufügenden Schlüssel permutiert sein können. Was die zufällige Löschung anlangt, gibt es jeweils drei gleichwahrscheinliche Möglichkeiten, welcher der drei vorhandenen Schlüssel gelöscht werden soll. Insgesamt gibt es also 72 gleichwahrscheinliche *Historien*, von denen eine jede genau einem Tupel (i_1, i_2, i_3, d, i_4) entspricht, wobei $d \in \{1, 2, 3\}$ und (i_1, i_2, i_3, i_4) eine Permutation der Elemente $\{1, 2, 3, 4\}$ ist. Wenn $d = 1, 2, 3$ wird das kleinste, mittlere, größte Element gelöscht. Während der aus $IIID$ resultierende BSB gemäß Hibbard's Theorem H random ist (für die entsprechenden 18 Historien resultiert je 9 mal die Gestalt A bzw. B aus Abbildung 1), zerstört die erste Einfügung nach der Löschung die Randomness. Die 5 Gestalten I,II,. . .,V treten nämlich mit den Wahrscheinlichkeiten

$$\mathbf{p}_{3,1} = (11/72,\ 13/72,\ 25/72,\ 11/72,\ 12/72)$$

auf, was nicht der random-Verteilung

$$\mathbf{p}_3 = (1/6,\ 1/6,\ 1/3,\ 1/6,\ 1/6)$$

entspricht (vgl. Abschnitt 1.2).

[12] In [Knu77] untersucht Knuth vielerlei Löschdisziplinen, wobei er für zufällige Löschungen (vgl. Abschnitt 1.2) das Symbol D_r verwendet. Die I^*D_r-Notation steht somit für 'beliebig viele zufällige Einfügungen gefolgt von einer zufälligen Löschung'.

Wenn man die interne Pfadlänge für BSB aus dieser Verteilung berechnet, erhält man $191/72 = 2.652777\ldots$, was besser als der random-Wert $I_3 = 8/3 = 2.666\ldots$ ist.[13]

Gary Knott hat im Rahmen seiner Arbeit [Kno75] auch umfangreiche empirische Läufe mit zufälligen Einfügungen und Löschungen der Form $I^n(ID)^m$ vorgenommen ($m = 24$ und $n = 2\ldots 9, 11, 14, 19, 49, 98$). Es wurden dabei sowohl für das Hibbard'sche als auch für das Knuth'sche Löschverfahren für jedes n jeweils 1600 BSB generiert. Die Ergebnisse dieser Simulationen wiesen eindeutig in die Richtung, dass zufällige Löschungen zu einer geringeren internen Pfadlänge führen, als unter Randomness zu erwarten, und dass dieses Verhalten beim Knuth'schen Löschverfahren noch ausgeprägter ist. Knuth präzisiert die '*Knott'sche Vermutung*' in [JK78, 304] folgendermaßen: *More precisely, Knott's conjecture is this: Consider a pattern of $m + k$ insertions and m deletions, in some order, where the number of deletions never exceeds the number of insertions. For example, one of the patterns with $m = 4$ and $k = 4$ is $IIIDIIDIIIDD$. To do each insertion, put a new random element into the tree, say a uniform random number between 0 and 1; to do each deletion, choose a random element uniformly from among those present. All of these random choices are to be independent. Then for each fixed pattern of I's and D's, the average path length of the resulting tree is conjectured to be at most equal to the average path length of the pattern consisting solely of k I's.*

Um neben den empirischen Befunden weitere Indizien für die Knott'sche Vermutung zu bekommen, gehen *Jonassen* und *Knuth* in besagter Arbeit so vor, dass sie für zufällige Update-Folgen der Bauart

$$III,\ IIIDI,\ IIIDIDI,\ \ldots,\ III\,(DI)^m,\ \ldots$$

nach der Verteilung $\mathbf{p}_{3,m}$ der BSB-Gestalten I,II,...,V fragen (für $m = 0, m = 1$ sind uns diese ja bereits bekannt). Insbesondere konnte nachgewiesen werden, dass $\lim_{m\to\infty} \mathbf{p}_{3,m}$ existiert, und dass die entsprechende Grenzverteilung

$$\mathbf{p}_{3,\infty} = (0.150,\ 0.196,\ 0.353,\ 0.137,\ 0.164)$$

beträgt (auf 3 Stellen gerundet). Die dazugehörige interne Pfadlänge beträgt $2.64749\ldots$, was wieder geringer als der random-Wert $I_3 = 8/3 = 2.666\ldots$ ist.[14] Es sei noch darauf hingewiesen, dass dieses Problem einfacher aussieht, als es ist. Das kommt auch durch den Titel der Arbeit '*A Trivial Algorithm Whose Analysis Isn't*' zum Ausdruck. Zur Ableitung der analytischen Ergebnisse war die Verwendung von Bessel-Funktionen und die Lösung von bivariaten Integralgleichungen notwendig.

In der gegenständlichen Arbeit wird dann auch noch $\mathbf{p}'_{3,m}$ unter dem gleichen Wahrscheinlichkeitsmodell aber unter Zugrundelegung des Knuth'schen Löschverfahrens untersucht. Das etwas verblüffende Resultat besteht darin, dass hier für alle $m \geq 0$ die interne Pfadlänge $8/3$ beträgt, also immer dem random-Wert entspricht. Das heißt, dass der verbesserte Löschalgorithmus von Knuth für $n = 3$ zu einer schlechteren internen

[13] Wenn man die gleiche Enumeration der Historien für die Knuth'sche Löschdisziplin vornimmt, ergibt sich die Verteilung $\mathbf{p'}_{3,1} = (13/72,\ 14/72,\ 24/72,\ 10/72,\ 11/72)$, was ebenfalls nicht der random-Verteilung entspricht. Die interne Pfadlänge entspricht aber mit $192/72 = 8/3$ "zufällig" dem random-Wert.

[14] Eine analoge Untersuchung von *R. A. Baeza-Yates* [BY89] kommt auch für $n = 4$ zu dem Ergebnis, dass die unter der stationären Verteilung $\mathbf{p}_{4,\infty}$ zu erwartende interne Pfadlänge geringer als der random-Wert I_4 ist.

Pfadlänge führt. *Jonassen* und *Knuth* kommentieren dieses Ergebnis wie folgt: ... *the average internal path length actually turns out to be worse when we use the "improved" algorithm. On the other hand, Knott's empirical data in [Kno75] indicate that the modified algorithm does indeed lead to an improvement when the trees are larger.*

Auf Grund der empirischen Befunde von *Knott* und der mit *Jonassen* erarbeiteten analytischen Resultate setzt *Knuth* die Bemerkung K.3a aus [Knu73b, 431] folgendermaßen fort:

K.3a': *Empirical evidence suggests strongly that the path length tends to* decrease *after repeated deletions and insertions, so the departure from randomness seems to be in the right direction; a theoretical explanation for this behavior is still lacking.*

Obwohl nun dank *Knott* aufgedeckt worden ist, dass die Verallgemeinerung H" von Hibbard nicht zulässig war, begeht er mit der *Knott'sche Vermutung*, die Abweichung ginge in die richtige Richtung, den nächsten Irrtum.

2.3 Eppinger widerlegt die Knott'sche Vermutung

J. L. Eppinger [Epp83] hat umfangreiche empirische Läufe durchgeführt, um zu überprüfen, ob sich die interne Pfadlänge bei gemischten zufälligen Einfügungen und Löschungen (mit dem Hibbard'schen Löschverfahren) tatsächlich entsprechend der *Knott'schen Vermutung* verhält. Er orientiert sich dabei grundsätzlich an der Versuchsanordnung von Knott, indem er von Folgen der Form $I^n(ID)^m$ ausgeht. Allerdings sind die untersuchten Werte von n (nämlich $n = 2^6, 2^7, \ldots, 2^{11}$) deutlich größer, als bei Knott (dessen maximales n nur 98 beträgt). Auch die Anzahl m der durchgeführten (ID)-Updates ist bei Eppinger weitaus größer: Während Knott immer nur $m = 24$ solcher Updates betrachtet, orientiert sich Eppinger an einem maximalen m-Wert der Größenordnung $2n^2$. Eppinger beobachtet nun den Verlauf der mittleren internen Pfadlänge $\overline{IPL}_{n,m}$ für zunehmende Werte von m. Der Mittelwert bezieht sich dabei auf Stichprobengrößen zwischen 750 und 6800. Für den größten Wert von $n = 2048$ beträgt die Stichprobengröße 5840.

Eppinger selbst beschreibt seine Beobachtungen folgendermaßen [Epp83, 667f]: *Initially, $\overline{IPL}_{n,m}$ decreases, as Knott observed. After some critical point, though, $\overline{IPL}_{n,m}$ starts to increase, eventually levelling off after approximately n^2 I/D pairs. ... Perhaps the most significant observation is that as n increases so does the asymptotic value for $\overline{IPL}_{n,m}/I_n$. Binary tree operation, such as insertion and deletion, can be modeled by Markov chains (but the state space would be quite large). Since any binary tree may be obtained by applying some combination of I/D pairs to any other binary tree, the $\lim_{m\to\infty} \overline{IPL}_{n,m}$ exists. Figure 7 suggests that*

$$\lim_{m\to\infty} \overline{IPL}_{n,m} > I_n$$

for sufficiently large values of n (roughly greater than 128). Thus binary trees seem to become "worse than random" after many insertions and deletions.

Eppinger setzt Methoden der Regressionsanalyse ein, deren Ergebnisse auf den folgenden Verlauf des stationären Werts der internen Pfadlänge in Abhängigkeit von n hinzuweisen

scheinen:

$$\lim_{m\to\infty} \overline{IPL}_{n,m} = 0.0280n\log_2^3 n - 0.392n\log_2^2 n + 3.03n\log_2 n - 4.81n.$$

Jedenfalls ist die *Knott'sche Vermutung* durch die Ergebnisse von *Eppinger* eindeutig widerlegt. Auf Grund der Ergebnisse von Eppinger erkennt man auch, dass die Versuchsanordnung von Knott in zweierlei Hinsicht unzureichend war, um den tatsächlichen Verlauf der internen Pfadlänge zu entdecken:[15]

a) Die von Knott untersuchten BSB lagen alle im Bereich $n \leq 98$. In diesem Bereich stabilisiert sich $\lim_{m\to\infty} \overline{IPL}_{n,m}$ aber noch unterhalb I_n.

b) Selbst wenn Knott größere BSB untersucht hätte, wäre die Anzahl m der von ihm beobachteten (ID)-Updates (nämlich bloß $m = 24$) viel zu gering gewesen, um nach der von Eppinger beobachteten anfänglichen Verringerung der internen Pfadlänge in den Bereich zu kommen, in dem sie sich wieder verschlechtert.

In einer weiteren Staffel von Experimenten weist Eppinger nach, dass der Grund für die deutliche Verschlechterung der internen Pfadlänge gegenüber random offenbar in der starken Asymmetrie der Hibbard'schen Löschoperation liegt. Bei Verwendung einer symmetrisierten Variante[16] beobachtet Eppinger nämlich ein ständiges Sinken der mittleren internen Pfadlänge, bis sie sich bei etwa 88% des random-Wertes stabilisiert.

2.4 Culberson und Munro vermuten, dass es noch schlimmer ist, als Eppinger annimmt, und entwickeln ein theoretisches Modell für diese Vermutung

Nach *Eppinger* setzen sich auch *J. Culberson* und *J. I. Munro* in einigen Arbeiten [Cul84], [Cul85], [Cul86], [CM89], [CM90] mit der *Knott'schen Vermutung* auseinander, wobei wieder von zufälligen Folgen der Bauart $I^n(DI)^m$ ausgegangen wird. Alle in diesen Arbeiten enthaltenen empirischen Untersuchungen bestätigen die Ergebnisse von Eppinger. Sie legen sogar nahe, dass sich die mittlere interne Pfadlänge noch nachteiliger entwickelt, als von Eppinger vermutet (der von einer Größenordnung von $\Theta(n\log_2^3 n)$ ausgeht). Culberson und Munro's Untersuchungen bekräftigen die Annahme, dass sich die mittlere interne Pfadlänge erst bei einem Wert von $\Theta(n^{3/2})$ stabilisiert. Dieses Verhalten wurde nicht nur für das Hibbardsche Löschverfahren beobachtet, sondern auch für Löschungen gemäß Knuth, die nur zu einer unwesentlichen Verbesserung führen.

Auch was die "symmetrisierten" Varianten der Hibbard'schen und Knuth'schen Löschalgorithmen anlangt, werden die Ergebnisse von Eppinger bestätigt bzw. ergänzt: In beiden Fällen stellt sich ein Gleichgewichtszustand ein, der deutlich besser ist als der random-Wert I_n. Das Verhältnis $\lim_{m\to\infty} \overline{IPL}_{n,m}/I_n$ scheint für großes n ($n = 1024$) bei

[15] Man muss Knott dabei aber sicher zu Gute halten, dass die für ihn (1972) verfügbare Hardware in ihrer Leistungsfähigkeit um ca. 2 Zehnerpotenzen unter der von Eppinger lag.

[16] Die originale Variante von Hibbard ist nach rechts orientiert. Man kann eine spiegelbildliche, nach links orientierte Variante schreiben. Die symmetrisierte Version ergibt sich durch starres oder randomisiertes Umschalten zwischen den beiden spiegelbildlichen Varianten, wobei Eppinger zwischen starrem und randomisiertem Umschalten keine signifikanten Unterschiede feststellen konnte.

0.88 bzw. 0.87 zu liegen.[17] Offenbar kommt es in erster Linie auf die Symmetrie des Löschverfahrens an. Demgegenüber wirken sich die Vorteile des Knuth'schen Verfahrens nur marginal aus. Dementsprechend bestätigen auch Culberson und Munro die Empfehlung von Eppinger, eine symmetrisierte Version des Knuth'schen Verfahrens zu verwenden (siehe beispielsweise [CM90, 311]).

Die Arbeiten von Culberson und Munro widmen sich aber auch den analytischen Aspekten des Problems. Es wird nämlich ein Modell entwickelt, mit dessen Hilfe die Autoren das von ihnen vermutete $\Theta(n^{3/2})$ Verhalten von $\lim_{m\to\infty} \overline{IPL}_{n,m}$ für die beiden asymmetrischen Löschvarianten erklären wollen. Dieses Modell wurde in [CM90][18] für '*Exact Fit Domains*' (EFD) entwickelt. Dabei werden (nachdem zunächst n Knoten zufällig eingefügt worden sind) Paare (DI) betrachtet, bei denen nach der Löschung D eines zufälligen Knotens der Schlüssel des gelöschten Knotens durch I sofort wieder eingefügt wird. Unter diesem EFD-Modell wird bewiesen, dass $\lim_{m\to\infty} \overline{IPL}_{n,m} = \Theta\left(n^{3/2}\right)$, und zwar sowohl für die Hibbard'sche als auch für die Knuth'sche Löschoperation.

Die Autoren glauben, dass diese Gesetzmäßigkeiten auch für die sonst betrachteten allgemeineren (DI)-Updates gelten, wozu sie in [CM89] ein entsprechendes Modell entwickeln. Entsprechend diesem Modell wäre $\lim_{m\to\infty} \overline{IPL}_{n,m}$ auch im allgemeinen Fall von der Ordnung $\Theta\left(n^{3/2}\right)$. Auf Grund dieses Modells erscheint es plausibel, dass sich für den führenden Koeffizienten der asymptotische Wert

$$\lim_{m\to\infty} \overline{IPL}_{n,m} \sim \frac{1}{3}\left(\frac{2}{\pi}\right)^{1/2} n^{3/2} = 0.266\ldots n^{3/2}$$

ergibt, was auch gut zu den empirischen Befunden passt.

Jedenfalls erscheint *Knuth* die obige Vermutung und ihre Begründung durch die Autoren ernsthaft und glaubwürdig genug, um sie wie folgt zu kommentieren [Knu98, 435]: *Further study by Culberson and Munro [CM89], [CM90] has led to a plausible conjecture that the average search time in the steady state is asymptotically* $\sqrt{2n/9\pi}$. Wir können nur hoffen, daß die Geschichte der Irrungen damit ein Ende hat.

Literatur

[BC60] A.D. Booth and A.J.T. Colin. On the Efficiency of a New Method of Dictionary Construction. *Information and Control*, Vol. 3:327–334, 1960.

[BY89] R.A. Baeza-Yates. A Trivial Algorithm Whose Analysis Isn't: A Continuation. *BIT*, Vol. 29:378–394, 1989.

[CM89] J. Culberson and J.I. Munro. Explaining the Behaviour of Binary Search Trees under Prolonged Updates: A Model and Simulations. *The Computer Journal*, Vol. 32(1):68–75, 1989.

[17] In [Knu98, 435] wird sogar ein Verhältnis von 0.86 angegeben.

[18] Dieses paper wurde vor [CM89] eingereicht (July 28, 1986 versus Dec. 1987) aber erst später (August 15, 1988 versus May 1988) angenommen. Obwohl es also erst später erschienen ist, ist es doch ein Vorgänger von [CM89].

[CM90] J. Culberson and J.I. Munro. Analysis of the Standard Algorithms in Exact Fit Domain Binary Search Trees. *Algorithmica*, Vol. 5(3):295–311, 1990.

[Cul84] J. Culberson. *Updating Binary Trees*, Technical Report CS-84-08. University of Waterloo, Canada, 1984.

[Cul85] J. Culberson. The effect of updates in binary search trees. *Proceedings of the 17th ACM Symposium on the Theory of Computing (STOC)*, pages 205–212, 1985.

[Cul86] J. Culberson. *The Effect of Asymmetric Updates in Binary Search Trees*. PhD thesis, Computer Science Dept., University of Waterloo, Waterloo, Ontario, 1986.

[Dou59] A.S. Douglas. Techniques for the Recording of, and Reference to Data in a Computer. *The Computer Journal*, Vol. 2:1–9, 1959.

[DR95] L. Devroye and B. Reed. On the Variance of the Height of Random Binary Search Trees. *SIAM Journal on Computing*, Vol. 24(6):1157–1162, 1995.

[Epp83] J.L. Eppinger. An empirical study of insertion and deletion in binary search trees. *Commun. ACM*, Vol. 26(9):663–669, 1983.

[Hib62] T.N. Hibbard. Some Combinatorial Properties of Certain Trees with Applications to Searching and Sorting. *JACM*, Vol. 9:13–28, 1962.

[JK78] A. Jonassen and D.E. Knuth. A trivial algorithm whose analysis isn't. *J. Comput. Syst. Sci.*, Vol. 16:301–322, 1978.

[Kno75] G.D. Knott. *Deletions in Binary Storage Trees*. PhD thesis, Computer Science Dept., Stanford University, Stanford, Calif., 1975.

[Knu73a] D.E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, MA, 1973.

[Knu73b] D.E. Knuth. *The Art of Computer Programming*, volume 3, 2nd printing (1975). Addison-Wesley, Reading, MA, 1973.

[Knu77] D.E. Knuth. Deletions That Preserve Randomness. *IEEE Transactions on Software Engineering*, Vol. SE-3(5):351–359, 1977.

[Knu97] D.E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, Reading, MA, 3 edition, 1997.

[Knu98] D.E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, MA, 2 edition, 1998.

[MR03] H.M. Mahmoud and Neininger R. Distribution of Distances in Random Binary Search Trees. *The Annals of Applied Probability*, Vol. 13(1):253–276, 2003.

[Win60] P.F. Windley. Trees, Forests, and Rearranging. *The Computer Journal*, Vol. 3:84–88, 1960.