Algorithmen und Datenstrukturen (Master)
WiSe 19/20

Benedikt Lüken-Winkels

November 21, 2019

Contents

Wörterbuchproblem

Menge S mit n Schlüssln aus einem Universum U. Operationen: INSERT (darauf achten, dass die Balance nicht verloren geht), DELETE, LOOKUP (Im Baum runterlaufen, bis das Element gefunden wurde)

Situationen

1. U linear geordnet, also existiert ein $\leq$-Test $\Rightarrow$ Suchbäume
2. U ist ein Intervall $\{0, ..., N-1\}$ der gesamten Zahlen $\Rightarrow$ Hashing

<u>**zu 1:**</u>

Randomisierte Suchbäume Idee: Benutze Zufallszahlen zur Balancierung eines binären Suchbaums

Binärer Suchbaum (Knoten-Orientiert) Schlüssel werden in den n Knoten eines binären Baums gespeichert, sodass im linken Unterbaum des Knotens mit Schlüssel x alle Schlüssel $< x$ **und** im rechten Unterbaum alle $> x$. Balanciert $\Rightarrow H\ddot{o}he(T) \leq logn$. Degeneriert $\Rightarrow H\ddot{o}he(T) = O(n)$

Definition: Randomized Search Tree (RST)

Sei $S = \{x_1, ..., x_n\}$ eine Menge von n Schlüsseln. Jedem x_i wird eine zusätzlich eine Zufallszahl (auch Priorität genannt) $prio(x_i)$ zugeordnet. $prio(x_i)$ sind gleichverteilte reelle Zufallszahlen $\in [0, 1]$ (Implementierung wären int-Zahlen, zB 32-bit).
Ein RST für S ist eine binärer Suchbaum für die Paare $(x_i, prio(x_i)), 1 \leq i \leq n$, sodass

1. normaler Knoten-orientierter Suchbaum für die Schlüssel $x_i, ..., x_n$
2. Maximumsheap bzgl der Prioritäten. dh $prio(v) \geq prio(u)$, falls v Parent. ((u,v) sind Knoten in einem Baum). $\Rightarrow$ Wurzel enthält maximale Priorität.

Existenz durch Algorithmus zum Aufbau (rekursiv).

- Wurzel einthält (x_i, p_i) mit $p_i = prio(x_i)$ maximal
- Linker Unterbaum: RST für $\{(x_i, p_i) | x_j < x_i\}$
- Rechter Unterbaum: RST für $\{(x_k, p_k) | x_k > x_i\}$

Beispiel: $S = \{1, ..., 10\}$

- Schreibe Tabelle mit Prioriäten und Werten.
- Teile die Tabelle beim Maximum und schreibe es in die Wurzel. Wiederhole, bis alle Elemente geschrieben.

$\Rightarrow$ Wenn sich die Prioritäten genauso oder umgekehrt, wie die Schlüssel verhalten, erhält man einen degenrierten Baum. (bzgl $\leq$). zB $prio(x_i) = x_i$. Dieser Fall ist sehr unwahrscheinlich, wenn sich bei der Priorität um gleichverteilte Zufallszahlen handelt.

Operationen

- Lookup(x): normale suche in binärem Baum. Kosten $O(Höhe(T))$
- Insert(x): Füge einen neuen Knoten v als Blatt $(x, prio(x))$ gemäß des Schlüssels in den binären Baum ein, wobei $prio(x)$ neue Zufallszahl (kann die Prio-Ordnung zerstören). Dann: Rotiere v nach oben, bis die Heap-Eigenschaft gilt, also $prio(v) \leq prio(parent(v))$. Kosten: O(#Rotationen) = O(Höhe(T)). Alternativ: normales einfügen in binären Baum in absteigender Reihenfolge der Prioritäten.
- DELETE(x): Sei v der knoten mit Schlüssel x (v = Lookup(x)). Kosten: O(#Rotationen) = O(1 + |L| + |R|)
  1. Rotiere v nach unten, bis v ein Blatt ist. R = linkes Rückgrat des rechten Unterbaums von v. L = rechtes Rückgrat des linken Unterbaums.
  2. Entferne das Blatt.
- Split(y) $\rightarrow S_1 = \{x \in S | x \leq y\}, S_2 = \{x \in S | x \geq y\}$ (Teile den Baum, indem y mit maximaler Priorität zur Wurzel rotiert wird)
  1. Insert($y + \epsilon$) mit Priorität ∞
  2. Entferne die Wurzel
- Join(T_1, T_2): $S \leftarrow S_1 \cup S_2$. T_1 RST für S_1 und T_2 RST für S_2
  1. Konstruiere T (Füge y zwischen Max(S_1) und Min(S_2) ein. Voraussetzung: Max(S_1) < Min(S_2)
  2. Lösche die Wurzel (Durch runterrotieren des eingefügten Knotens y)

Analyse des RST

Wir analysieren die erwarteten Kosten einer Delete-Operation (Insert $\rightarrow$ umgekehrtes Delete). Seit T ein RST für die Menge $\{x_1, ..., x_n\} mit x_1 < x_2 < ... < x_n$ der durch Inserts aufgebaut wurde. Bertrachte die Operation Delete(x_k) für eine $k, 1 \leq k \leq n$. Für einen Knoten x_k im Baum T mit Suchpfad P_k, L_k rechtes Rückgrad von T_l und R_k linkes Rückgrad von T_r. Kosten $O(|P_k|+|L_k|+|R_k|)$. Wir schätzen die Erwartungswerte

Lemma 1:

- a) E($|P_k|$) $= H_k + H_{n-k+1} - 1$

$$k - te\ HarmonischeZahl = H_k = \sum_{i_1}^{k} \frac{1}{i}\ H_k \leq ln(x) + 1$$

- b) E($|L_k|$) $= 1 - \frac{1}{k}$
- c) E($|R_k|$) $= 1 - \frac{1}{n-k+1}$

Beweis Betrachte eine Permutation $\pi : [1..n] \rightarrow [1..n]$ (bijektive Abbildung), die die Schlüssel absteigend nach ihren Prio Werten sortiert. Dann gilt:

1. Jede Permutation π ist gleichwahrscheinlich (Wahrscheinlichkeit $\frac{1}{n!}$), da die Prioritäten gleichverteilte Zufallszahlen sind.

2. Man erhält den selben binären Baum durh einfügen der Schlüssel in einen unbalancierten Baum in der Reihenfolge, die π angibt. $\rightarrow$ gleiches Vehalten, wie ein zufälliger binärer Baum.

3. Baum wächst nur an den Blättern.

Trick: arbeite ab jetzt mit zufälliger Permutation statt den Prioritäten. $\rightarrow$ normaler Binärbaum mit zufälliger Einfügereihenfolge.

Teil a) des Lemmas P_k ist Suchpfad für Knoten x_k. Seien P'_k und P''_k Teilfolgen von P_k mit: $\forall v \in P'_k key(v) \leq x_k$