

Info II Tutorium

– Sommer 2006 –

Joachim Wilke

<http://joachim-wilke.de>

13. Juli 2006



Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Quellennachweis & Dank an:
Joachim Breitner, Felix Brandt, wikipedia.de, Folien der Vorlesung.

Übersicht



- 1 Organisatorisches
- 2 Einstiegsfragen
- 3 Bonusblatt
- 4 Dynamisches Programmieren
- 5 KMP-Algorithmus zur Textsuche

- 1 Organisatorisches
- 2 Einstiegsfragen
- 3 Bonusblatt
- 4 Dynamisches Programmieren
- 5 KMP-Algorithmus zur Textsuche

- Diese Woche ist das letzte bewertete Übungsblatt dran.
- Abgabeschluss ist Mittwoch, 19. Juli 2006!
- Übungsscheine können in der letzten Vorlesungswoche abgeholt werden.
- Am 20. Juli findet das letzte Tutorium statt.
- Nicht vergessen - Anmeldung zur Klausur (06.09.06, 10 Uhr):
 - 24.-28.07.06: Sekretariat (280) / Übungsleitung (244/246)
 - 31.07.-04.08.: im Sekretariat (280), 9-13 Uhr.
 - 21.08.-01.09.: Sekretariat (280) / Übungsleitung (244/246)

- 1 Organisatorisches
- 2 Einstiegsfragen**
- 3 Bonusblatt
- 4 Dynamisches Programmieren
- 5 KMP-Algorithmus zur Textsuche

Einstiegsfragen



- 1 Wann lohnt sich Hashen?
- 2 Wie groß ist der Suchaufwand im O-Kalkül für verkettetes Hashen (worst-case)?
- 3 Wie groß ist der Suchaufwand im O-Kalkül wenn keine Kollisionen auftreten?
- 4 Warum sind AVL-Bäume besser als herkömmliche binäre Suchbäume?
- 5 Was versteht man unter der AVL-Eigenschaft?
- 6 Wann verwendet man einen Doppelrotation, wann eine einfache Rotation?

Einstiegsantworten



- 1 Wenn aus einem großen Schlüsselraum heraus verhältnismäßig wenige Schlüssel gespeichert werden müssen.
- 2 $O(n)$ – linear
- 3 $O(1)$ – konstant
- 4 Der Suchaufwand ist auch im Worst-Case $O(\log n)$
- 5 Für die Balancefaktoren b_i aller Knoten im Baum muss gelten: $|b_i| \leq 1$
- 6 Das kommt auf die Struktur des Teilbaums an in dem das Ungleichgewicht festgestellt wurde (siehe Grafiken im 10. Tutorium).

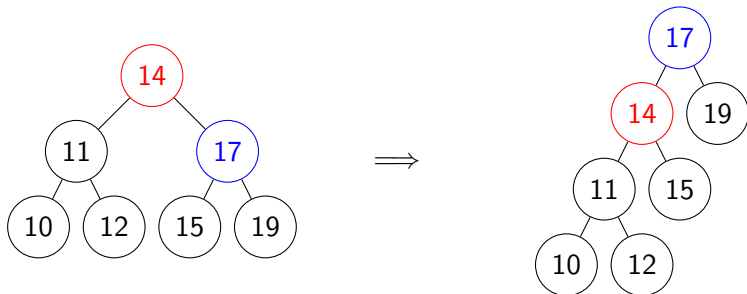
- 1 Organisatorisches
- 2 Einstiegsfragen
- 3 Bonusblatt**
- 4 Dynamisches Programmieren
- 5 KMP-Algorithmus zur Textsuche



- ① Ein Modell für die Formel $F = \neg\forall xP(x)$:
 - zum Beispiel $U = \mathbb{N}$ und $I_{\mathcal{A}}(P) = 1$
- ② Skolemform von $\forall x(\exists zP(z) \wedge \forall zQ(x, z))$:
 - zum Beispiel $\forall x\forall z(P(f(x)) \wedge Q(x, z))$.
- ③ Sind $(F \rightarrow G)$ und $(F \models G)$ gültige prädikatenlogische Formeln?
 - Ja und nein.
- ④ Allgemeinsten Unifikator von $K = \{Q(x, y), Q(x, f(x)), Q(a, y)\}$:
 - $[x/a][y/f(a)]$
- ⑤ $F_1 = \forall xP(x)$ und $F_2 = \exists xP(x)$ erfüllbarkeitsäquivalent?
 - Ja, denn beide Formeln sind erfüllbar.



- 16 Einfache Rotation nach links an Knoten 14.



Rekurrenzen



- 17 $T(1) = 1, T(n) = 2T(n-1) + 3$
- $T(1) = 1, T(2) = 5, T(3) = 13, T(4) = 29, \dots$
 - $T(2) - T(1) = 4, T(3) - T(2) = 8, T(4) - T(3) = 16, \dots$
 - $T(n) = 2^{n+1} - 3$

- 21 Rekurrenz für die Zahl der Additionen (+,-) in folgendem Haskell-Programm

```

1 | f :: Integer → Integer
2 | f n | n < 0 = 0
3 |     | n ≥ 0 = 1 + 3*(f (n-2)) - 2*(f (n-4))

```

- für $n < 0 : T(n) = 0$
- sonst: $T(n) = 4 + T(n-2) + T(n-4)$

- 1 Organisatorisches
- 2 Einstiegsfragen
- 3 Bonusblatt
- 4 Dynamisches Programmieren**
- 5 KMP-Algorithmus zur Textsuche

Dynamisches Programmieren



Idee

- Lösung eines Optimierungsproblems aus den Lösungen kleinerer Teilprobleme zusammensetzen
- Zwischenergebnisse nur einmal berechnen (dann in Tabelle speichern)
- Teile-und-Herrsche-Algorithmus

Anwendungen

- Spracherkennung
- Plagiatsanalyse
- Lösung kombinatorischer Probleme

- 1 Organisatorisches
- 2 Einstiegsfragen
- 3 Bonusblatt
- 4 Dynamisches Programmieren
- 5 KMP-Algorithmus zur Textsuche**



Worum geht es?

- String-Matching (Suchwort in Text finden)
- Naiver Ansatz $O(m \cdot n)$
- Beschleunigen auf $O(n)$

Idee

- Berechne Präfixe für das „autokorrelierte“ Suchwort
- Lineares Suchen (kein Schritt zurück), da bei jedem Mismatch nur in einen anderen Zustand gesprungen wird (vergleichbar mit endl. Automaten).

Berechnung der Präfixfunktion π 

COMPUTE-PREFIX-FUNCTION(P)

```
1   $m \leftarrow \text{laenge}(P)$ 
2   $\pi[1] \leftarrow 0$ 
3   $k \leftarrow 0$ 
4  for  $q \leftarrow 2$  to  $m$ 
5      do while  $k > 0$  und  $P[k + 1] \neq P[q]$ 
6          do  $k \leftarrow \pi[k]$ 
7          if  $P[k + 1] = P[q]$ 
8              then  $k \leftarrow k + 1$ 
9           $\pi[q] = k$ 
10 return  $\pi$ 
```

Aufgaben



Aufgabe 1

Berechnet für das Muster ababaca die Funktion π :

x	0	1	2	3	4	5	6	7
$P[x]$		a	b	a	b	a	c	a
$\pi[x]$	-1	0	0	1	2	3	0	1

Hinweise zur Berechnung

- Gesucht ist für jedes Präfix des Musters x der Rand r
- Beispiel: Für das Muster abcd müssen die Ränder für ϵ , a, ab, abc und abcd berechnet werden.
- Die Länge des Randes von ϵ ist -1.
- Ein Rand eines Wortes muss ein echtes Präfix dieses Wortes sein.

Aufgaben



Aufgabe 2

Sucht das Muster abrakadabra in folgendem Text:

ababrakabrakadabradabaka

- Es gilt: $\text{shift} = \text{len_match} - \pi[\text{len_match}]$

Lösung:

```

a b a b r a k a b r a k a d a b r a d a b a k a
a b r a k a d a b r a
  a b r a k a d a b r a
    a b r a k a d a b r a

```

Ende

