12. Weitere Beispiele zum dynamischen Programmieren

Bisher dynamisches Programmieren bei:

- Kürzester einfacher Weg (bei negativen Kanten): $O(n^2 \cdot 2^n)$
- TSP: $O(n^2 \cdot 2^n)$
- Hamilton Kreis: $O(n^2 \cdot 2^n)$
- Kürzester einfacher Weg (bei Graphen ohne negative Kreise): $O(n^3)$

Floyd Warshall

Hier noch einige Beispiele, die zu polynomialer Zeit führen, also wie Floyd Warshall.

Nun Suchen.

Zunächst das Problem: Optimaler, statischer binärer Suchbaum.

Gegeben: n (Schlüssel-)Werte,

$$a_1 \lneq a_2 \lneq \cdots \lneq a_n$$

mit n Zugriffswahrscheinlichkeiten

a_1 mit Ws-keit p_1, a_2 mit p_2, ..., a_n mit p_n, dabei

$$\sum_{i=1}^{n} p_i = 1 .$$

Gesucht: Binärer Suchbaum T für $a_1, \ldots, a_n$. Die Kosten für das Suchen in T mit Häufigkeiten p_i sollen minimal sein.

(12.8)

Was sind die Kosten von T?

<u>Definition</u>

(a) $K(T) := \sum_{i=1}^{m} p_i \cdot \underbrace{(\text{Tiefe}_T(a_i)+1)}_{=\#\text{Baumknoten, Kosten bei der Suche nach } a_i}$

Wo-keit von a_i

(b) T ist optimal, gdw.

$K(T) = \min\{K(S) \mid S \text{ bin. Sb. für } a_1, \dots, a_m\}$ $\square$

Beachte: Ein optimaler Sb. existiert immer, da es nur endlich viele Suchbäume zu $a_1, \dots, a_m$ gibt.

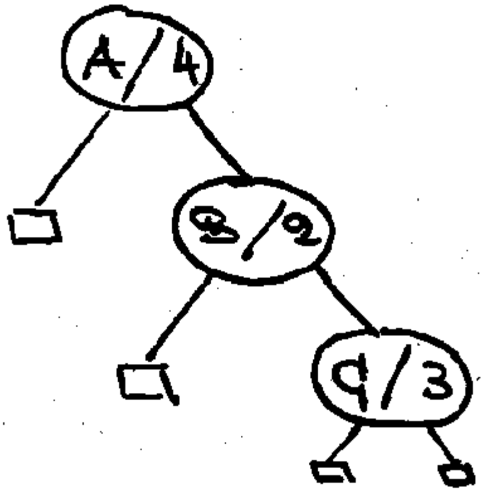
BEISPIELE

B W-KEIT 2 $(2/9)$

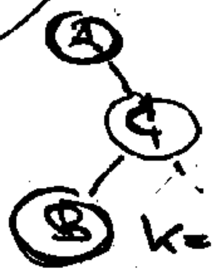
C W-KEIT 3 $(3/9)$

A W-KEIT 4 $(4/9)$

$A \leq B \leq C$



ABER

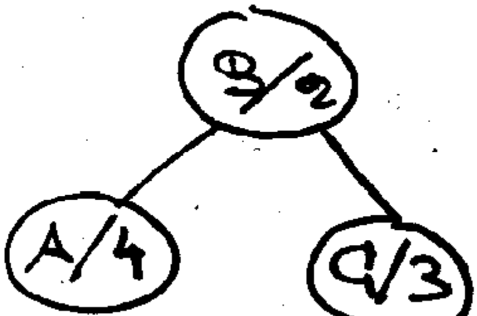


$k = 16.$

$k = 4 + 4 + 9 = 17$

A B C

$K = 9 + 5 + 3 = 17$ $(17/9)$



$k = 8 + 2 + 6 = 16 \leq 17$

A B C

$K = 2 + 4 + 3 = 16$ $(16/9)$

NICHT IMMER: WURZEL = HÄUFIGSTES

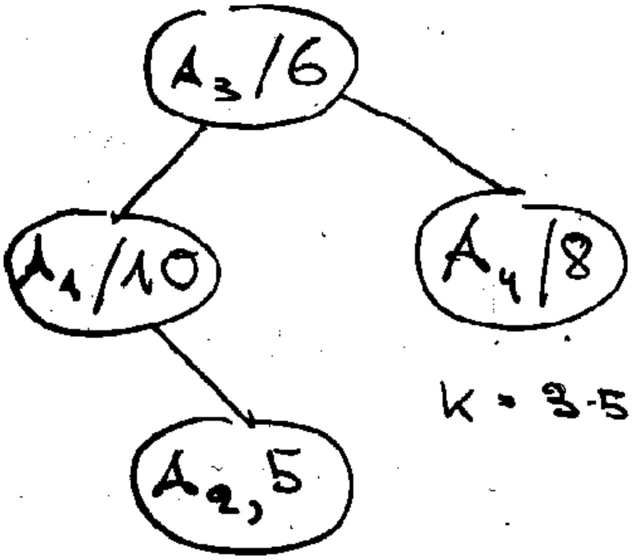
IM OPTIMALEN BAUM →a

12.4a

WEITERES BEISPIEL

$A_1 \leq A_2 \leq A_3 \leq A_4$

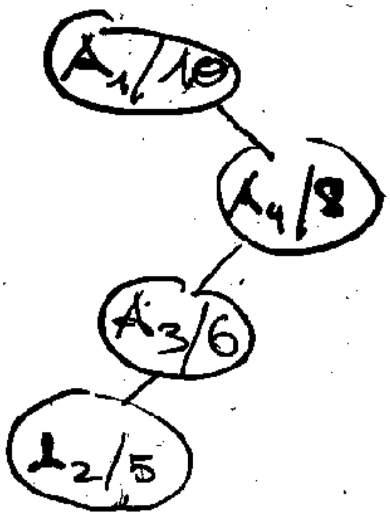
W-KEIT: 10 5 6 8



$K = 3\cdot 5 + 2\cdot 10 + 2\cdot 8 + 6$
$= 57$

BESTE KOSTEN MIT A_1 AN

DER WURZEL:



$K = 10 + 2\cdot 8 + 3\cdot 6 + 4\cdot 5$

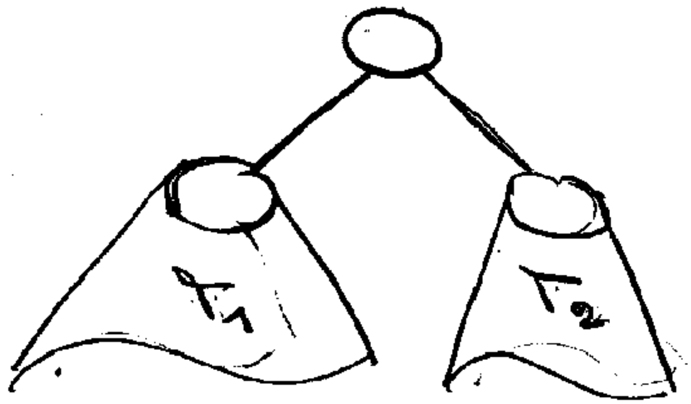
$= 64$

(12.5)

Ziel: Optimaler bin. Sb. in $O(n^3)$.

Haben wir einen beliebigen bin. Sb. für $a_1 \lneq a_2 \lneq \dots \lneq a_n$ vorliegen, so lassen sich seine Kosten bei Häufigkeiten p_i folgendermaßen ermitteln:

Wurzel betreten: $1 = \sum p_i$.



Mehrfach-zählung!

Linker Sohn der Wurzel: $\sum_{a_i \in T_1} p_i$

Rechter " : $\sum_{a_j \in T_2} p_j$

⋮ usf.

Allgemein sagen wir

$$k_T(a) = \sum_{a_j} p_j$$

Teilbaum mit Wurzel a von T.

Nach Definition von

<u>Folgerung</u>

$$K(S) = \sum_{i=1}^{n} k_S(a_i)$$

für binären Suchbaum S mit $a_1, \ldots, a_n$.

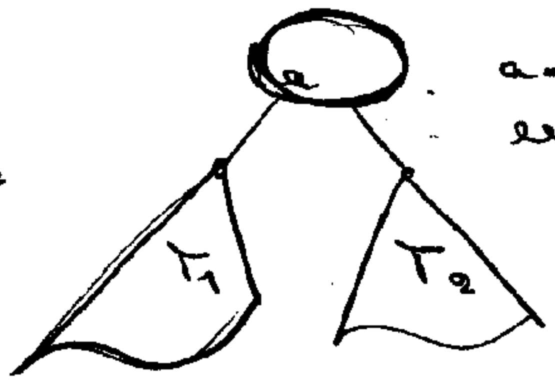
<u>Beweis</u>

Ind. über n. $n = 1$, dann ✓

Ind.-Schluß: Sei $n \geq 1$. Wir betrachten $a_1, \ldots, a_{n+1}$ mit beliebigen p_i. Es ist

127

$S =$



$a = a_j$ für ein a_j

Dann ist

$k(S)$

Definition

$$= \sum_{i=1}^{m+1} p_i \cdot \left(\text{Tiefe}_S(a_i) + 1\right)$$

$$= \sum_{i=1}^{m+1} p_i + \sum_{i=1}^{m+1} p_i \cdot \text{Tiefe}_S(a_i)$$

$\text{Tiefe}_S(a_j) = 0$

$$= 1 + \sum_{a_i \in T_1} p_i \cdot \text{Tiefe}_S(a_i) + \sum_{a_i \in T_2} p_i \cdot \text{Tiefe}_S(a_i)$$

$$= 1 + \sum_{a_i \in T_1} p_i\left(1 + \text{Tiefe}_{T_1}(a_i)\right) + \sum_{a_i \in T_2} p_i\left(1 + \text{Tiefe}_{T_2}(a_i)\right)$$

(12.8)

$=$ Ind.-Vor. für T_1, T_2

$$1 + \sum_{a_i \in T_1} K_{T_1}(a_i) + \sum_{a_i \in T_2} K_{T_2}(a_i)$$

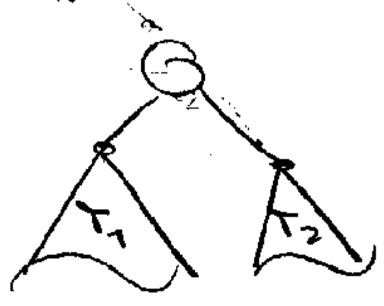
$=$

$$K_S(a_i) + \sum_{a_i \in T_1} K_S(a_i) + \sum_{a_i \in T_2} K_S(a_i)$$

$$= \sum_{i=1}^{n} K_S(a_i). \qquad \square$$

Folgerung

Ist für eine Menge $b_1, \ldots, b_l$ mit p_i, $\sum p_i \le 1$

$S =$ 

ein Baum, so daß $\sum K_S(b_i)$ minimal ist, so ist für T_1, T_2

$$\sum_{b_i \in T_1} k_{T_1}(b_i), \quad \sum_{b_i \in T_2} k_{T_2}(b_i)$$

minimal.

<u>Beweis:</u>

$$\sum_{i=1}^{\ell} k_T(b_i) = \sum p_i + \qquad + \qquad \square$$

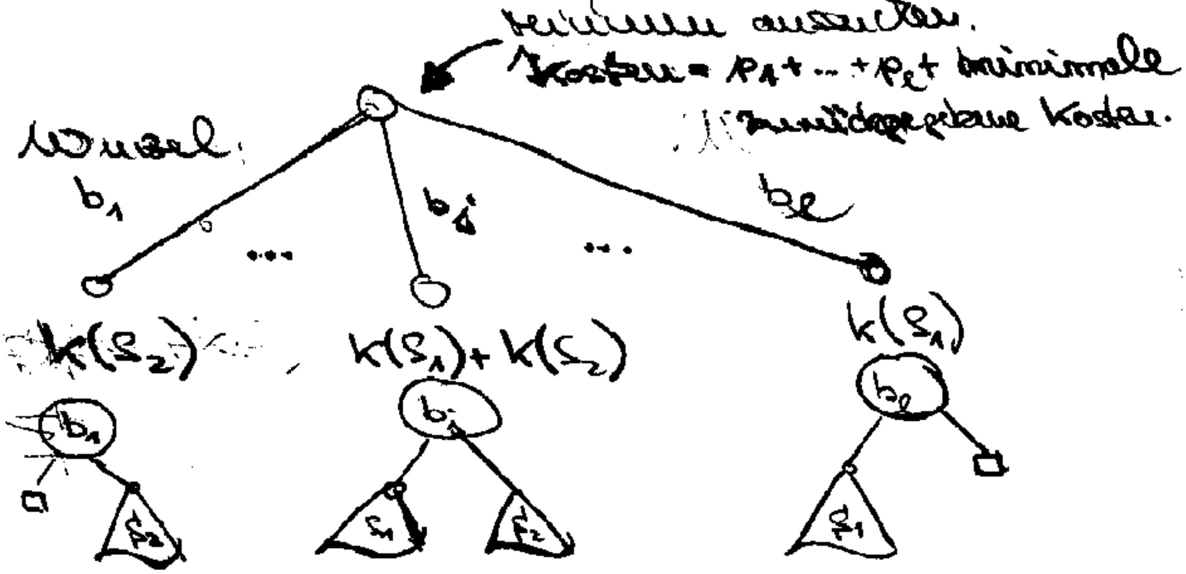
Prinzip: Teile optimaler Lösungen sind optimal. Dominiert beim dynamischen Programmieren.

Bei Eingabe von $b_1, \ldots, b_\ell$, $b_1 \le \ldots \le b_\ell$ mit W-keit p_i, $\sum p_i \le 1$

(12.10)

rekursive Ermittlung von einem Baum β mit

$$\sum_{i=1}^{\ell} k_\beta(b_i) \quad \text{minimal:}$$



β_2 rekursiv.

β_1 mit $b_1, \dots, b_{j-1}$
β_2 mit $b_{j+1} - b_\ell$
rekursiv

β_1 rekursiv

Laufzeit: $\Rightarrow \Omega(2^\ell)$, 2^ℓ, ...

nur ganz links und ganz rechts.

Struktur des Aufrufbaums statisch.

Verschiedene Aufrufe

(12.11)

$\leq$ # verschiedene Folgen der Art

$b_k \leq b_{k+1} \leq \dots \leq b_{h-1} \leq b_h$, wobei
$1 \leq k \leq h \leq l$

$= \left| \{ (k,h) \mid 1 \leq k \leq h \leq l \} \right|$

$= \underbrace{l}_{k=1} + \underbrace{(l-1)}_{k=2} + \underbrace{(l-2)}_{k=3} + \dots + \underbrace{1}_{k=l}$

$= \frac{l(l+1)}{2} = \Theta(l^2)$.

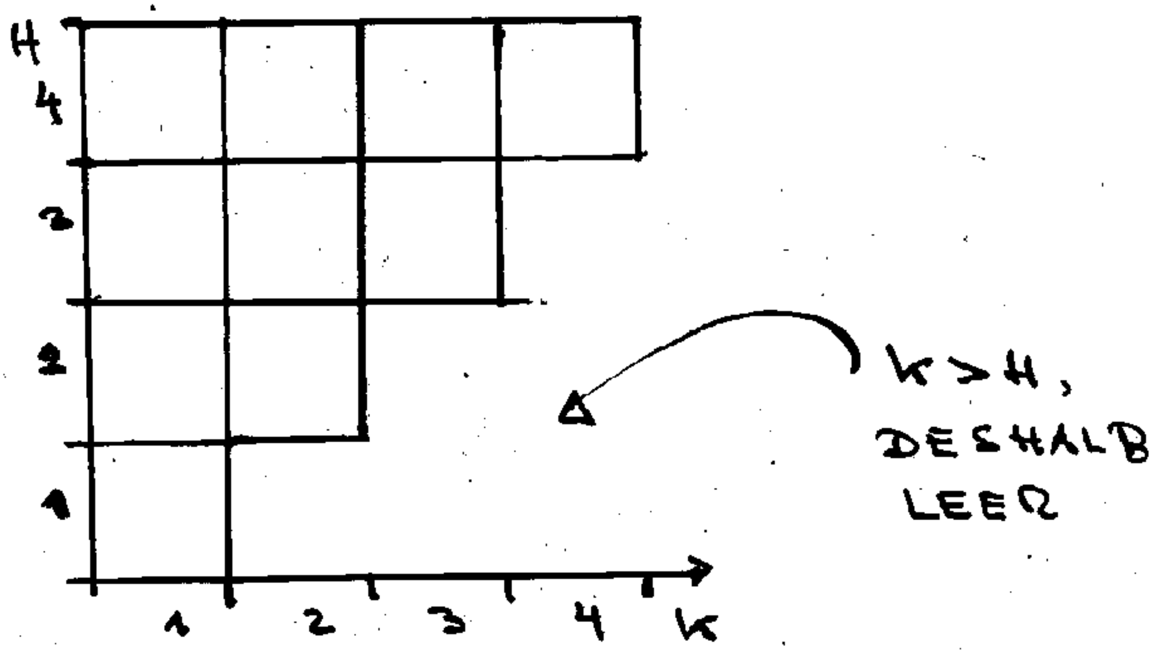
Bei $a_1, \dots, a_m$ mit p_i $\sum p_i = 1$

Tabelle $T(k,h)$ $1 \leq k \leq h \leq m$

12.12

TABELLE T(K,H)

$1 \leq K \leq H \leq N,\ N = 4$



$T[1,1] = P_1,\ T[2,2] = P_2,$

$T[3,3] = P_3 \quad T[4,4] = P_4$

$$T[1,2] = \mathrm{MIN}\left\{ \begin{array}{l} P_1 + P_2 + T[2,2] \\ P_2 + P_1 + T[1,1] \end{array} \right\}$$

$T[2,3],\ T[3,4]$

$$T[1,3] = P_1 + P_2 + P_3 + \min\{T[2,3],\ T[1,1] + T[3,3],\ T[1,2]\}$$

$T[2,4]$

$$T[1,4] = P_1 + P_2 + P_3 + P_4 + \min\{T[2,4],\ T[1,1] + T[3,4],\ T[1,2] + T[4,4],\ T[1,3]\}$$

IM ALLGEMEINEN:

$\Theta(n^2)$ EINTRÄGE IN T.

$O(n)$ PRO EINTRAG

ALSO $O(n^3)$.

INDUKTIV ÜBER $n-k$!

$$T[k,n] = P_k + \dots + P_n + \min\left(\{T[k+1,n]\} \cup \{T[k,n-1]\} \cup \{T[k,j-1] + T[j+1,n] \mid k+1 \le j \le n-1\}\right)$$

BEISPIEL VON 12.4

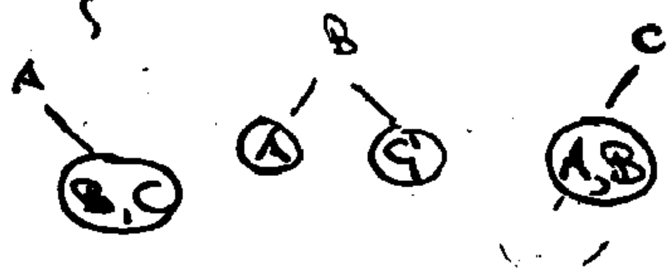
B 2 C 3 , A 4

$A \leq B \leq C$

H			
C	16	7	3
B	8	2	
A	4		
	A	B	C K

$T[AC] = 9 +$

$\min\{7, 4+3, 8\}$



Dynamisches Programmieren häufig bei Fragestellungen auf Worten (→ Bioinformatik, algorithmische Biologie). Worte der Art

$w = a_1 \dots a_m$, a_i ein character

als array $w[1, \dots, m]$ of char

mit $w[i] = a_i$.

Zunächst eine

Bezeichnung

Das Wort v ist eine Teilfolge von $w = a_1 \dots a_m$ gdw. es gibt $1 \leq i_1 \lneq i_2 \lneq \dots \lneq i_k \leq m$ so daß

$$v = a_{i_1} \cdot a_{i_2} \dots a_{i_k} .$$

□

Ist $w = abcd$, so sind

a, b, c, d

ab, ac, ad, bc, bd, cd

abc, acd

und $abcd$ und das leere Wort ε sind alle Teilfolgen von w. Ist

$$w = a_1 \dots a_m$$

so: Teilfolge von w entspricht Folge über $0,1$ der Länge m.

$$\# \text{ Teilfolgen von } w \leq 2^m.$$

(bei $w = abcd$ eben 16). Beachte aber $w = aaaa$, dann

$$\# \text{ Teilfolgen von } w = 5.$$

(12.17)

Beim Problem der längsten gemeinsamen Teilfolge haben wir 2 Worte v und w gegeben und suchen ein Wort u so dß:

- u Teilfolge von v und w.
- Es gibt kein s, $|s| > |u|$, und s ist Teilfolge von v und w.

Hier ist $|w|$ = # Ecken von w, die Länge von w. Wir setzen $|\varepsilon| = 0$.

Bezeichnung $\mathrm{lgT}(v, w)$.

Ist etwa:

$v = abab$, $w = baba$,

so aba und bab jeweils $\mathrm{lgT}(v, w)$.

(12.18)

Ist

$w = aba$, $v = acacac$,

so ist aa eine $lgT(v, w)$.

Also: Nicht direkt hintereinander!

Man kann $lgT(v, w)$ auf

$lgT(v', w')$ mit $|v'| + |w'| \lneq |v| + |w|$

zurückführen:

Satz

Ist $v = a_1 \dots a_m$ und $w = b_1 \dots b_n$, so gilt:

(a) Ist $a_m = b_n = a$, so ist jede $lgT(v, w)$ von der Form ua und u eine $lgT(a_1 - a_{m-1}, b_1 - b_{n-1})$.

(b) Ist $a_m \neq b_n$ so ist jede

$\mathrm{lgT}(v, w)$ eine $\mathrm{lgT}(a_1 - a_{m-1}, w)$

oder eine $\mathrm{lgT}(v, b_1 - b_{n-1})$.

Beweis

(a) Ist t eine Teilfolge von

v und w ohne a am Ende, so

ist ta eine längere gemeinsame

Teilfolge.

Bsp: $v = aa$,
$w = aaa$, $\mathrm{lgT}(v, w)$
$\Rightarrow \mathrm{lgT}(a, aa) \cdot a$.

Ist ta eine gemeinsame

Teilfolge von v und w aber

t keine $\mathrm{lgT}(a_1 - a_{m-1}, b_1 - b_{m-1})$,

so ist ta keine $\mathrm{lgT}(v, w)$.

(12.20)

(b) Für jede gemeinsame Teilfolge u von v und w gilt eine der Möglichkeiten:

v = abba, w = baab
ab ist lgT(abb, baab)
ba ist lgT(abba, baa)

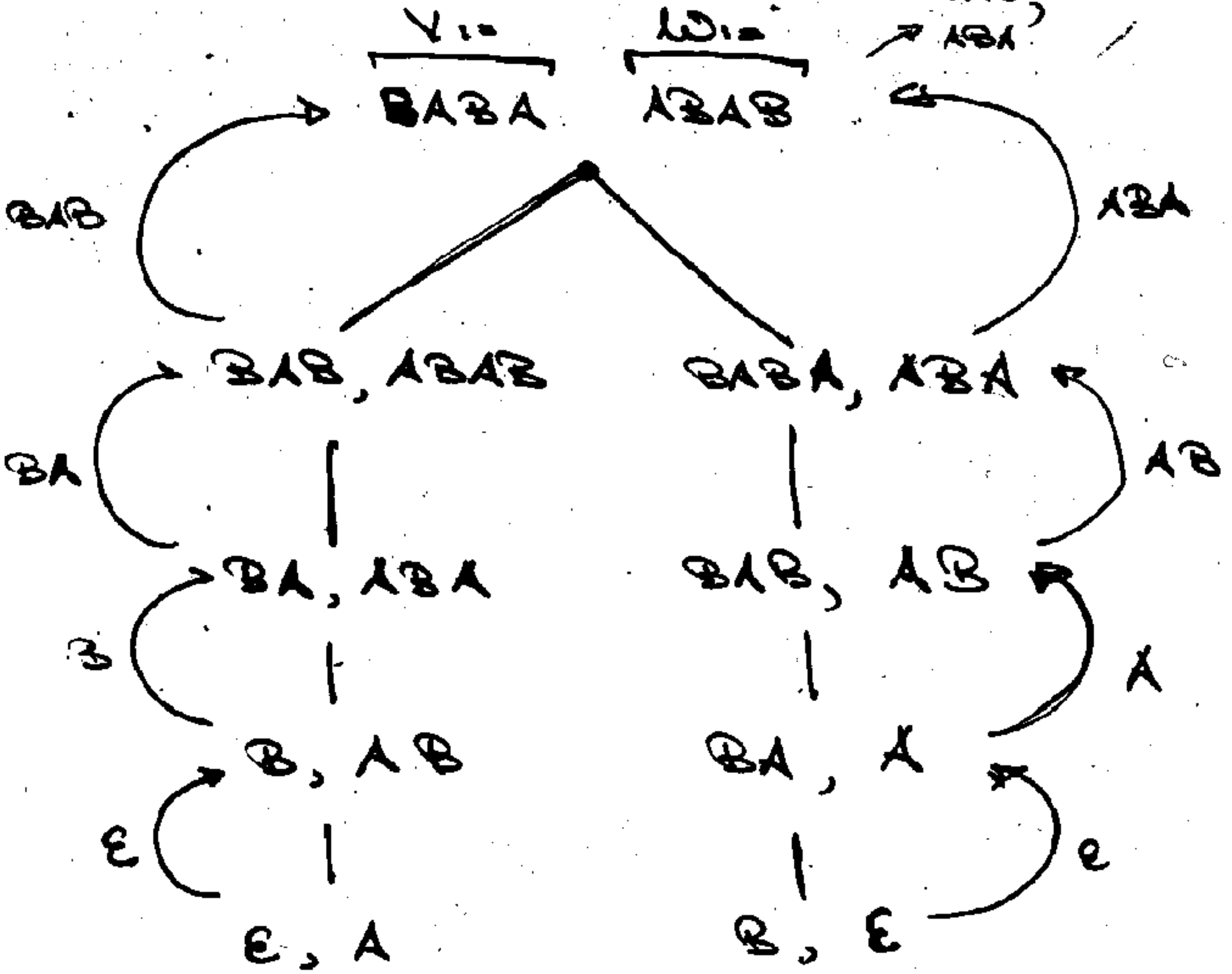
- u Teilfolge von $a_1 - a_{m-1}$, $b_1 - b_{n-1}$
- " $a_1 - a_{m-1}$, $b_1 - b_n$
- " $a_1 - a_{m-1}$, $b_1 - b_{n-1}$.

□

Also wieder: Reduktion auf optimale Lösungen von Teilen. Rekursive Lösung:

```
lgT(v, w) { // v = a_1 - a_m, w = b_1 - b_n
1. if m = 0 oder n = 0 { return ε // ε = leeres
                       }         // Wort
2. if a_m = b_n { d := lgT(a_1 - a_{m-1}, b_1 - b_{n-1});
                  return „d verlängert um a_m"
                }
3. else { d := lgT(a_1 - a_{m-1}, b_1 - b_n);
          e := lgT(a_1 - a_m, b_1 - b_{n-1});
        } return „Längeres von d, e" // |d|=|e|, beliebig.
}
```

AUFRUFBAUM

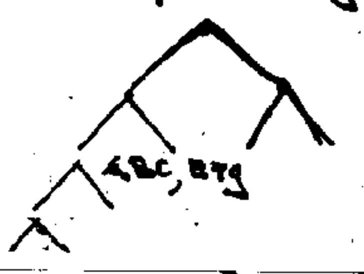


ALSO ABA ODER BAB.

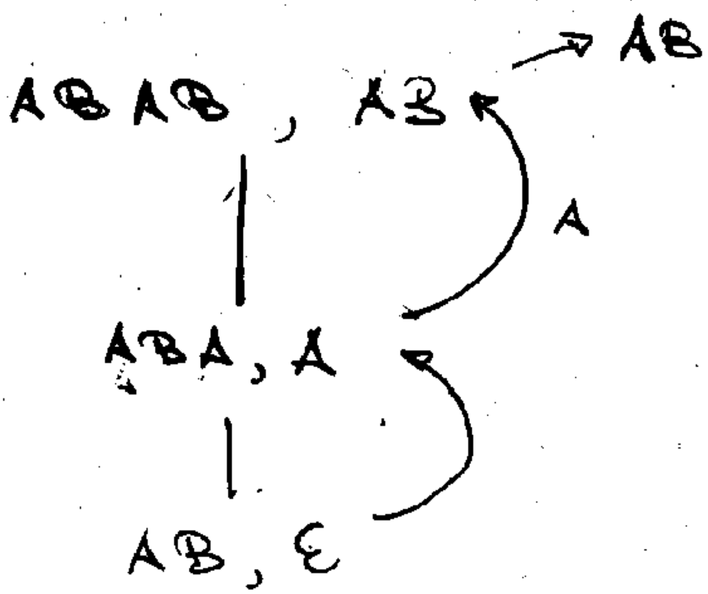
STRUKTUR DES BAUMS:

NICHT ! STATISCH.

GRÖSSE DES BAUMS:

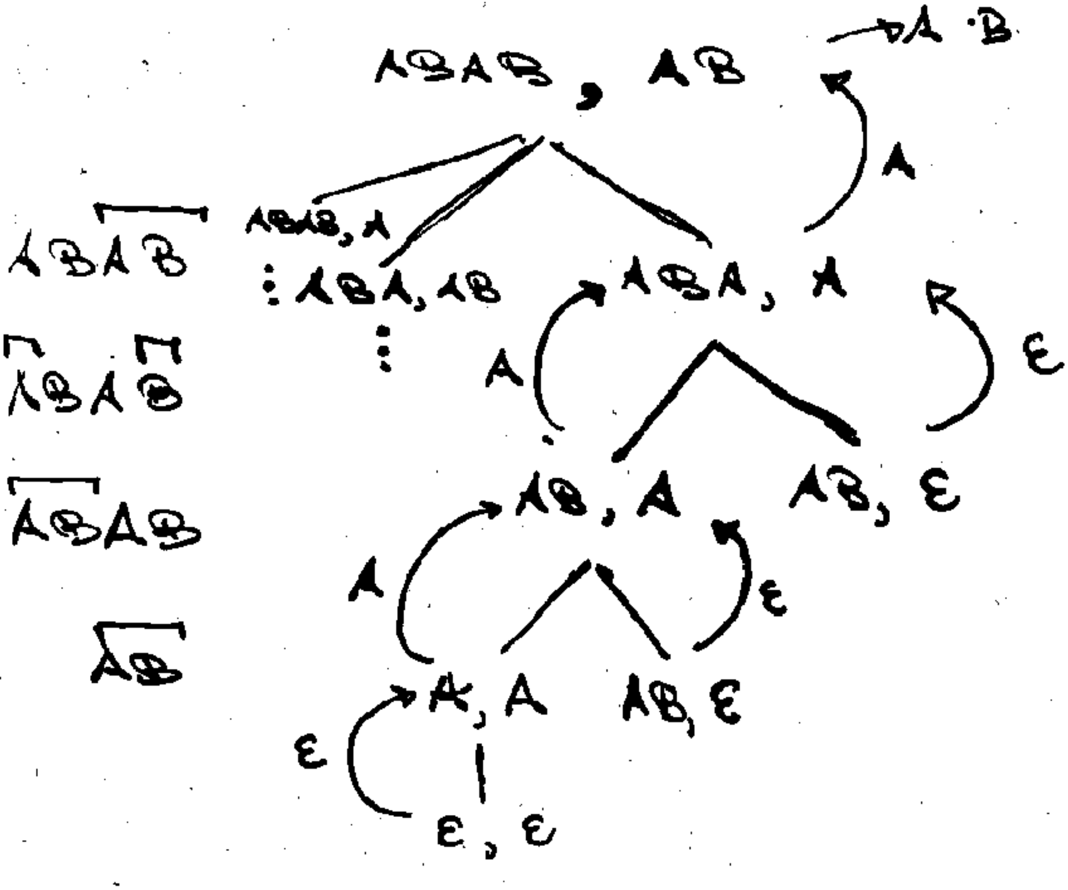


$\Rightarrow \geq 2^N$ BEI $|V| = |W| = N$.



FINDEN ABAB , AB

WIE ALLE MÖGLICHKEITEN?



(12.23)

Sei jetzt wieder $|v| = m$, $|w| = n$.

Wieviele verschiedene rekursive Aufrufe haben wir?

Aufruf $\Leftrightarrow$ 2 Anfangsstücke von v und w.

v hat $m+1$ Anfangsstücke (inklusive v und ε), w hat $n+1$ Anfangsstücke. Also

$$\#\text{Aufrufe} \leq (m+1)(n+1).$$

Tabelle $T[k,h]$, $0 \leq k \leq m$,
$0 \leq h \leq n$.

$T[k,h]$ = Länge eines $\mathrm{lgT}(a_1 - a_k, b_1 - b_h)$

$v = a_1 - a_m$, $w = b_1 - b_n$.

(12.24)

Dann T[

$$T[0,h]=0 \text{ für } 0 \le h \le m$$

$$T[k,0]=0 \text{ für } 0 \le k \le n.$$

und weiter für $h,k \geq 0$

$$T[k,h] = \begin{cases} T[k-1,h-1] & \text{falls } a_k = b_h \\ \max\{T[k,h-1], T[k-1,h]\} & \text{falls } a_k \neq a_h \end{cases}$$

Hier aus $\max\{T[k-1,h-1], T[k,h-1], T[k-1,h]\}$

und Mitführen von pointern zu den Maxima erlaubt die Ermittlung aller lgT(v,w) mit ihren Positionen.

Laufzeit: $\Theta(m \cdot n)$, da pro Eintrag $O(1)$.

BEISPIEL

$V = ABCD$, $W = ABC$

$T[H,K]$, $0 \le H \le 4$, $0 \le K \le 3$

		A	B	C	D
K \ H	0	1	2	3	4
0	0	0	0	0	0
A 1	0	↖ 1	← 1	← 1	← 1
B 2	0	↑ 1	↖ 2	← 2	← 2
C 3	0	↑ 1	↑ 2	↖ 3	← 3

$T[H,K]$

↖ $\Leftrightarrow +1$, ↑, ← $\Leftrightarrow +0$

SPALTENWEISE VON LINKS NACH RECHTS

ZUSÄTZLICH $B[I,J]$ $\forall$ $\{1,\dots,H\}\times\{1,\dots,K\}$ WEGEN ↖, ↑, ←.