Kap. 4 Positionsbäume

Kap. 4.1 Datenstruktur und Eigenschaften

Zweck:z.B. Textverarbeitung., genetische Information (Genom 10^9B)

$x = x_1x_2 \ldots x_n \in X^*;\ x_i \in X$ (X ist Alphabet)

␣ Leerzeichen; ε leere Zeichenreihe

n sehr groß, z.B. 10^7, wie groß für großen Brockhaus?

$y = y_1, \ldots, y_m$

Def.: ***y kommt in x an Stelle i vor*,**

wenn $x_i\ x_{i+1} \ldots x_{i+m-1} = y$

1

(P1) Finde alle Vorkommen von y in x,
z.B. für Buchindex
ersetze überall INHALT durch VOLUMEN

1. Verwende y, um Textstellen zu identifizieren
2. Wird Stelle durch u eindeutig identifiziert? „Positionsidentifikator“

Def.: u heißt ***Positionsidentifikator*** für Pos. i in x, wenn u kürzeste, eindeutige Teilzeichenreihe von x beginnend in Pos. i ist, d.h.

$x = yuz \ \wedge\ x = y`uz`$
$\Rightarrow y = y`$ und u ist kürzeste Teilzeichenreihe mit dieser Eigenschaft.

Hinw.: $|y| = i-1,\quad z = z`$

Problem: u existiert nicht immer,
z.B. bbb, Ident. für Pos.2?

2

Lemma: Sei $ Sonderzeichen, d.h. $ $\notin$ X,
dann hat jede Position in x$ einen Positions-Identifikator
(Umkehrung?)

Bew.: In $x_1x_2 \ldots x_u\$$ kommt
$x_i \ldots x_u\$$ genau einmal vor,
jetzt kürze $x_i \ldots x_u\$$ sukzessive von hinten

Beispiel: abab $
p(1) = aba; p(2) = ba; p(3) = ab$
p(4) = b $; p(5) = $

Def.: X-Baum: orientierter oder geordneter Baum mit Kantenmarken (Selektoren) aus **X**

3

Def.: ***Positionsbaum*** T für
$x\$ = x_1x_2 \ldots x_u\$$ mit $x_i \in X$
ist ein $\mathbf{X} \cup \{\$\}$-Baum mit:
1. T hat u + 1 Blätter markiert mit 1, 2, ..., u + 1 für Positionen von x$
2. Markenfolge längs Pfad von Wurzel zu Blatt i ist Pos.Id. p(i) für i

Problem: Invarianz von Pos.Ids., z.B. (Kap., Unterkap., Absatz, Wort)

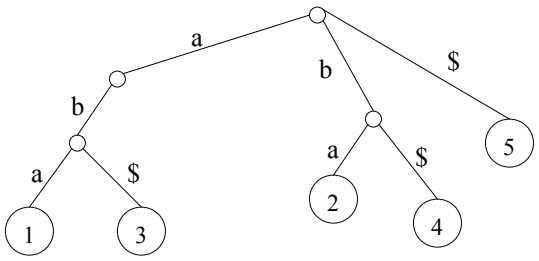
Lemma: x$ hat eindeutigen Positions-Baum

Bew.: Eindeutigkeit der Pos.Ids. in x$

4

Beispiel:

Für abab$

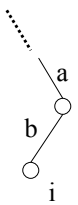


5

Kap. 4.2 Konstruktions-Algorithmen

Lemma: Falls $x \neq \varepsilon$, dann hat jedes Blatt von x $ mindestens einen Bruder

Bew.:



d.h. p(i) = ... ab
ist kürzeste identifizierende Teilzeichenreihe für Pos. i. Wenn (i) keinen Bruder hätte, könnte man p(i) kürzen,
siehe auch Beispiel für abab$ ↘

6

4.2.1 Naive Konstruktion

D A T E N S T R U K T U R E N $
1 2 3 4 5 6 7 8 . . . 15 16
1
2
3
4
.
.
.

Hinw.: i.e. Algorithmus $O(n^2)$
Bibel ≈ 100 MB = 10^8 Bytes

Beobachtung: kurze Pos. Identifikatoren

7

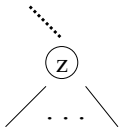
4.2.2 Alg. für Rechts-links Konstruktion

von Pos. Baum T_i ($x_i\, x_{i+1}$ ... x_n\$)
konstruiere T_{i-1} ($x_{i-1}\, x_i$ ... x_n\$)
d.h. in T_i folge Pfad mit Marken: $x_{i-1}\, x_i$ ... bis

Fall a: Blatt j erreicht wird, d.h.
p(j) kommt auch an Stelle i-1 vor
⇒ p(j), p(i-1) verlängern

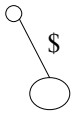
8

Fall b: Zwischenknoten z erreicht wird, von dem aus p(i-1) nicht weiterführt



Beispiel: T (abba$)

12345

T_5: 

T_4:

9

Ordnung nach Kantenmarken für schnellen Selektor-Zugriff!

Aufwand: $\sum_{i=1}^{n} |p(i)|$

mit Kostenamortisierung

Nachteil: Konstruktion von T(x $) nach voller Eingabe von x.

T für umgekehrten Text?

$x = x_1 x_2 \ldots x_n$ x$

$\overline{x} = x_n x_{n-1} \ldots x_1$ $\overline{x}$$

$x_i \ldots x_2 x_1$$

$\underbrace{\qquad}_{p`(i)}$

Textstellen zur Identifizierung rückwärts eingeben

10

4.2.3 Links-Rechts Konstr.

„on-line“ Verfahren, Konstruktion von T(x) in Realzeit mitlaufend.

Triviale Idee: Übergang von
$T(x_1 \ldots x_i\$) \Rightarrow T(x_1 \ldots x_i\, x_{i+1}\$)$
Änderungen in T an vielen Stellen

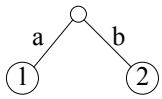
Def.: ***Partieller Pos. Baum*** $P_J(x_1 \ldots x_i)$
mit $J \subseteq [1{:}i]$ enthält für $k \in J$
genau kürzeste Zeichenreihe ab k:
$p_J(k)$, die Position k innerhalb von J identifiziert,
(d.h. $p_J(k)$ kommt an keiner anderen Stelle $m \in J$ vor.

Anm.: $P_J(x)$ existiert nicht immer

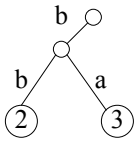
11

Beispiel: x = abbab

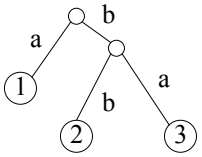
$P_{\{1\}}(x) = \varnothing$ $\quad P_{\{1,2\}}(x) =$



$P_{\{2,3\}}(x) =$

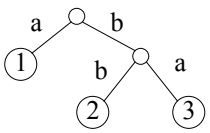


$P_{[1:3]}(x) =$



$\nexists\, P_{\{1,4\}}(x)$

Übergang: $P_{[1:3]}(x)$ zu $P_{[1:4]}(x)$

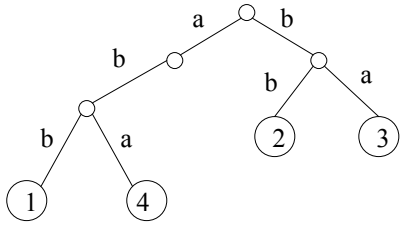


x = abbab
$P_{[1:4]}(x)$ existiert nicht
$P_{[1:4]}(xa)$

P[1 : 5](abbaba) $\quad \nexists$
P[1 : 5](abbaba\$) $\quad$ Jede Position hat Identifikator

12

Fortschreibung von $P_{[1:4]}$ (abbaba)
zu $P_{[1:5]}$ (abbaba$)
zu $P_{[1:7]}$ (...)



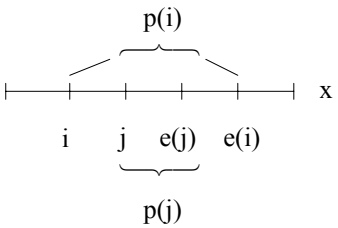
Lemma: Für n = |x| gilt:
$P_{[1: n+1]}(x\$) = T(x\$)$

Bew.: nach Konstruktion, es werden genau die Pos. Identifikatoren eingefügt

13

Monotonie Lemma für Pos. Identifikatioren:
Sei e(i) die Pos., bei der p(i) endet,
dann gilt: $i < j \Rightarrow e(i) \leq e(j)$

Bew.: Trivial, falls i = e(i), sonst Bew. durch Widerspruch:
Sei $i < j$ und $e(i) > e(j)$



$p(i) = x_i\ x_{i+1} \ldots x_{e(i)}$
d.h. $x_i \ldots x_{e(i)-1}$ kommt nochmals vor

d.h. p(j) kommt auch nochmals vor, ist nicht Pos.Id. ↯

14

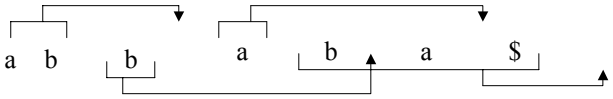
Graphische Deutung:

... x_i x_{i+1} ... x_k ...

Intervall von Positionen

deren Pos.Ids. hier enden

z.B.



15

d.h. bei links-rechts Aufbau von T(x)
kann für i < j p(j) frühestens mit p(i) entstehen.

Grundidee: für links – rechts Konstruktion des Positions-Baums:

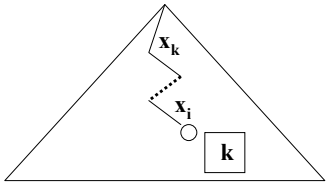
Für x_1x_2 ... x_i konstruiere partiellen Pos.B. für alle Positionen mit Pos.Ids. einschl der in i endenden

x_1x_2 ... x_{k-1} x_k ... x_i

d.h. $P_{[1:k-1]}$

16

x_k ... x_i ist Anfang von p(k),
entspricht Pfad durch $P_{[1:k-1]}$ (x_1 ... x_i)
Führe Marke [k] in $P_{[1:k-1]}$ (x_1 ... x_i)



Fortschreibung von $P_{[1:k-1]}$ (x_1 ... x_i)
bei Eingabe von x_{i+1}:

17

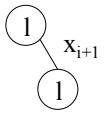
Fall 1: [k] neben internem Knoten
1.1 verschiebe [k] längs vorhandener Kante x_{i+1} weiter nach unten

1.2 [k] wird über neue Kante x_{i+1} zu Blatt (k), p(k) fertig.

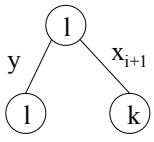
Führe [k+1] ein, iteriere

Fall 2: [k] steht neben Blatt l:
verlängere p(l) um 1 Zeichen y, neues Blatt (l)

2.1 $y = x_{i+1}$:



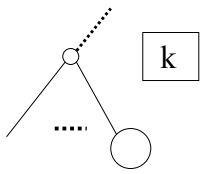
2.2 $y \neq x_{i+1}$:



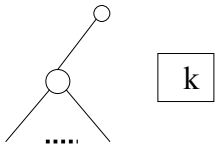
iteriere für
[k+1]

18

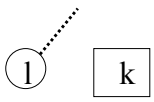
1.1



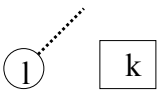
1.2



2.1



2.2



Kosten:
Eintrag von Blatt (k) , schrittweiser Aufbau von p(k), gesamt |p(k)|
für ganz T(x $):

$$c^{\sum_{i=1}^{|x|+1} |p(i)|}$$

19

Für sprachliche Texte typisch:

$$|p(i)| \sim \log |x|$$

⇒ **Gesamtkosten** O(|x|log|x|)
wie bei rechts - links Konstruktion!

Praktische Variante: partielle Pos.B. für Positionen entsprechend Wortanfängen, nach␣, CR, Seitenwechsel, etc. oder entsprechend Teilsatzanfängen (z.B. Lexikon für Zitate) nach. ; ! ? : etc.

20

4.2.4 Korrektur von Texten und zugehörigen Positions-Bäumen

Hauptproblem: Invarianz der Positionen, verwende Pos. #, die gegenüber typischen Änderungen weitgehend invariant sind,
4.20 für Folie 20 in Kap. 4

Korrekturoperationen:

1. **streichen:** x = u v w z
   u, z : invariant;
   v : streichen; w : variant

Ansatz: lösche vw;
füge w mit neuen Pos. ein

21

2. **einfügen:** x = u w z
   v einfügen zu x` = u v w z
   u v w z wie vorher
   u w z wie vorher
   Ansatz: lösche w; füge v w mit neuer Position ein

3. **ersetzen:** v durch v`in x = u v w z
   zu x`= u v`w z
   Ansatz: lösche v w; füge v`w mit neuen Positionen ein

Grundidee für Alg.:
Schritt 1: bestimme y = v w in 1. und 3.
y = w in 2.
Schritt 2: delete (y, x) in x = u y z
durch Übergang von T(x) zu partiellem Pos.Baum
ohne diejenigen Positionen j, deren p(j) mit y überlappen.

22

Schritt 3: Sei y`, w, v w, v`w entsprechend 1., 2., 3.
Ergänze partiellen Pos.Baum aus Schritt 2 zu T(x`); x`= u y`z

---

zu Schritt 2: delete y in u y z, welche p(j) überlappen y?

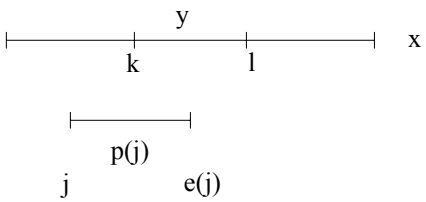
$u_1u_2 \ldots u_{k-1} \quad y_k \ldots y_lz$

Menge der Positionen j,
deren p(j) zu entfernen sind

23

Def.: $A(j) \underset{def}{=} \{i: e(i) = j\}$, Menge von Anfangspositionen

Anm.: nach Monotonie-Lemma ist A(j) ein Intervall



1. $e(j) < k$ keine Überlappung
2. $j > l$ keine Überlappung
3. $\neg (1. \vee 2.) = \neg 1. \wedge \neg 2.$
   $= e(j) \geq k \wedge j \leq l$ Überlappung

$\tilde{A}(k,l) \underset{def}{=} \{j : j \leq l \wedge e(j) \geq k\}$

24

Überlegung: $\tilde{A}(k,l)$ ist Intervall von Positionen:

$$\left[\begin{array}{l} \min(j) \ : \ l \\ e(j) \geq k \end{array}\right]$$

Def.: $T(x\$)\backslash J \underset{\text{def}}{=} P_{[1:|x|+1]\backslash J}(x\$)$

Lemma: Sei $x = u\,y\,z$; $u' = u\,y'z$

$T(x\$)\backslash\tilde{A}(k,l) = T(x'\$)\backslash\tilde{A}(k,m)$

25

Bew.: Nach Konstruktion

Korrektur Alg.: ersetze y durch y`
T(x$) sei vorhanden
1. Übergang von $T(x\$) \Rightarrow T(x\$)\backslash\tilde{A}(k,l)$
2. ersetze Text $x = u\,y\,z \Rightarrow x' = u\,y'z$
3. Ergänze $T(x'\$)\backslash\tilde{A}(k,m)$ zu $T(x''\$)$

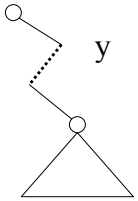
Korrektur-Kosten: pro Entfernung
von p(j) Aufwand $\leq |p(j)| \approx \log|x|$,
ebenso pro Einfügung
$\Rightarrow (|A(k,l)| + |A(k,m)|) \bullet \log|x|$

26

4.3 Anwendungen von Pos. Bäumen

für Texte und genetische Information:

(P_1) Finde alle Vorkommen von y in x



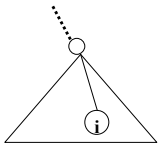
27

(P_2) Text Korrektur

(P_3) Finde längste mehrfach vorkommende Teilzeichenreihe:
Führe Pegel für tiefste Blätter bei Baumaufbau

Beispiel: Haben Programme X, Y größere gemeinsame Teile?

(P_4) Interne Mustergleichheit: längste Zeichenreihe ab Pos. i,
die mehrfach vorkommt? Länge = |p(i)|-1 an Positionen
in diesem Unterbaum



(P_5) Muster in DNA-Ketten

28

Anwendung Genom Datenbanken

U : Uracil
C : Cytosin
A : Adenin
G : Guanin

} Basen von Nukleotid-Bausteinen in Messenger-RNS

Triplett codiert den Einbau einer von 20 Aminosäuren bei Proteinsynthese,
z.B. UGG ~ Tryptophan (= Aminosäure) Einbau
4^3 Kombinationen = 64 Zeichen im Grundalphabet von Tripletts
RNS-Sequenzen von Trippletts ~
Aminosäuren-Sequenzen in Proteinmolekülen (Primärstruktur)

Problem: Finde gesuchte RNS-Sequenzen
innerhalb von bekannten RNS-Sequenzen
typisch 10^4 bis 10^5 Tripletts pro Protein

29

Mengengerüst menschliches Genom

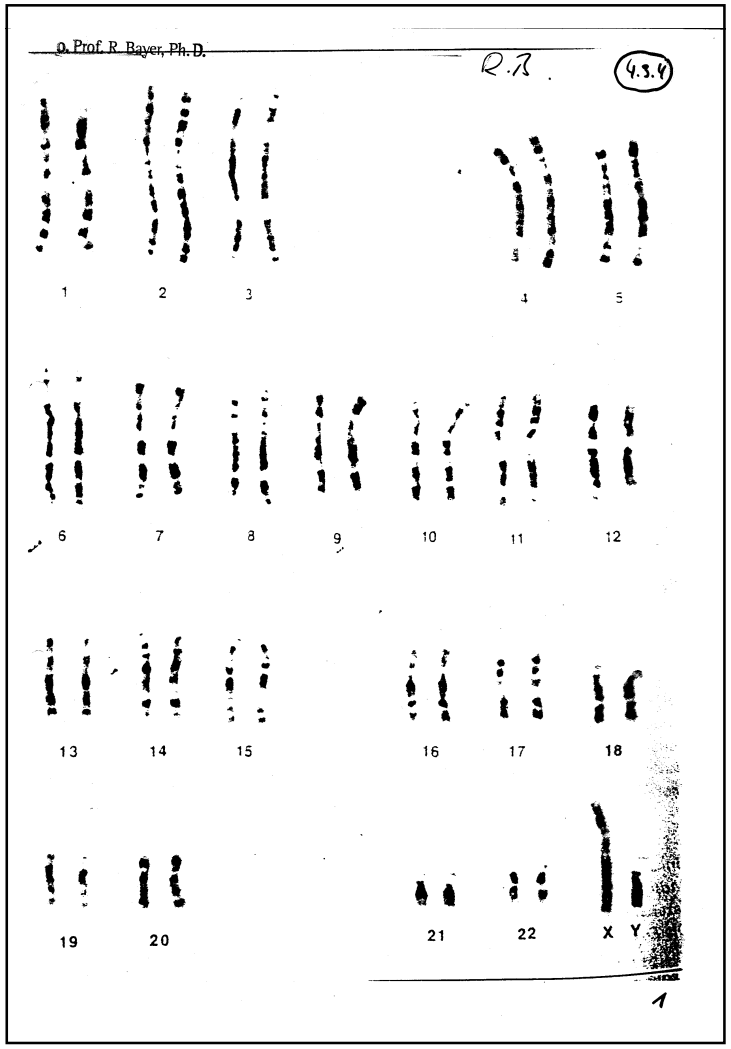
ca. 50.000 Gene
1 Gen ≈ Codierung eines Proteins

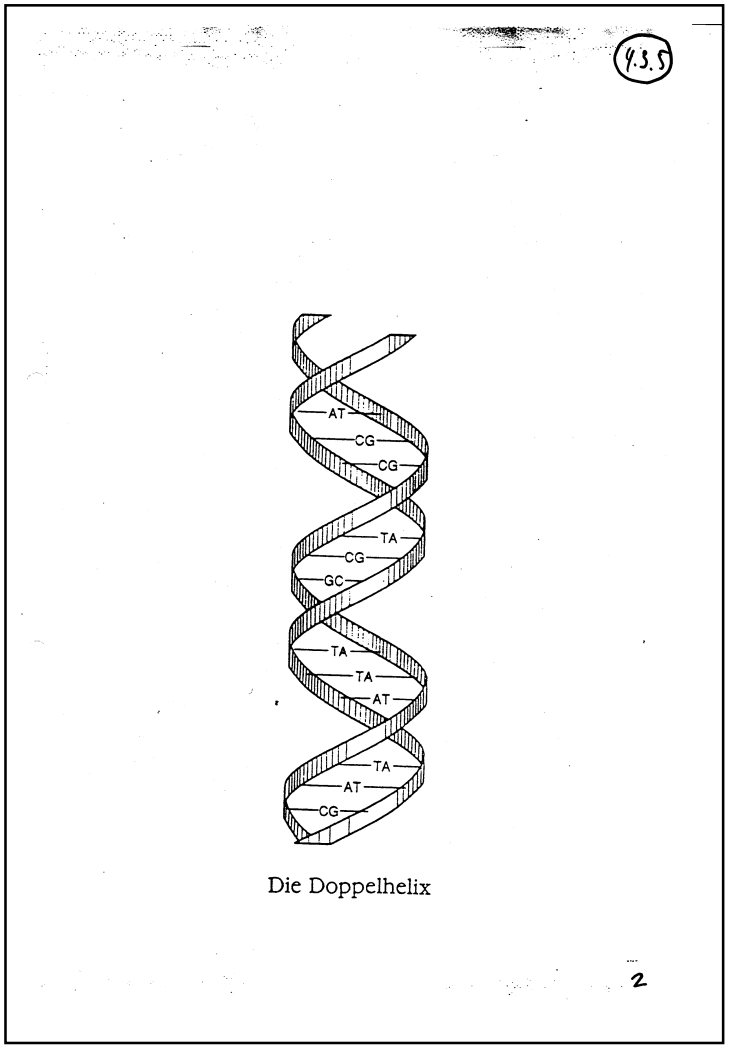
Größe eines Gens: ca. 40.000 Basenpaare,
einschließlich unverstandenes „Verpackungsmaterial“

⇒ menschl. Genom ca.
$2 \bullet 10^9$ Basen $= 4 \bullet 10^9$ Gbit = 500 MB = 1CD

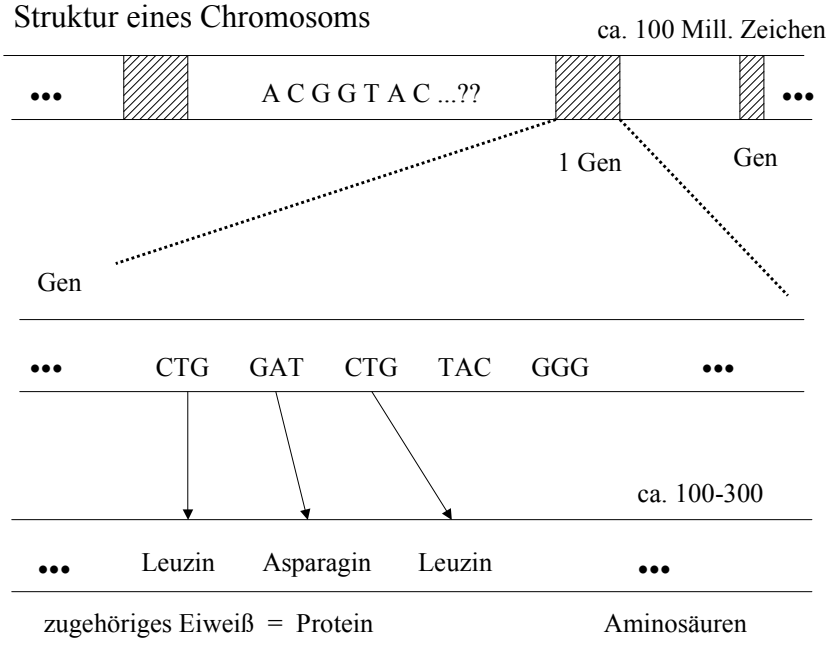
Quelle: Mannheimer Forum 90/91, Boehringer Mannheim, darin:
H.P. Vosberg: Konstanz und Variabilität im menschlichen Genom, S.81-142

30


o. Prof. R. Bayer, Ph. D.
R.B.
4.3.4
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
X
Y
1


4.3.5
AT
CG
CG
TA
CG
GC
TA
TA
AT
TA
AT
CG
Die Doppelhelix
2

Struktur eines Chromosoms



33

Die Buchstaben: DNS-Basen

A : Adenin
C : Cytosin
T : Thymin
G : Guanin

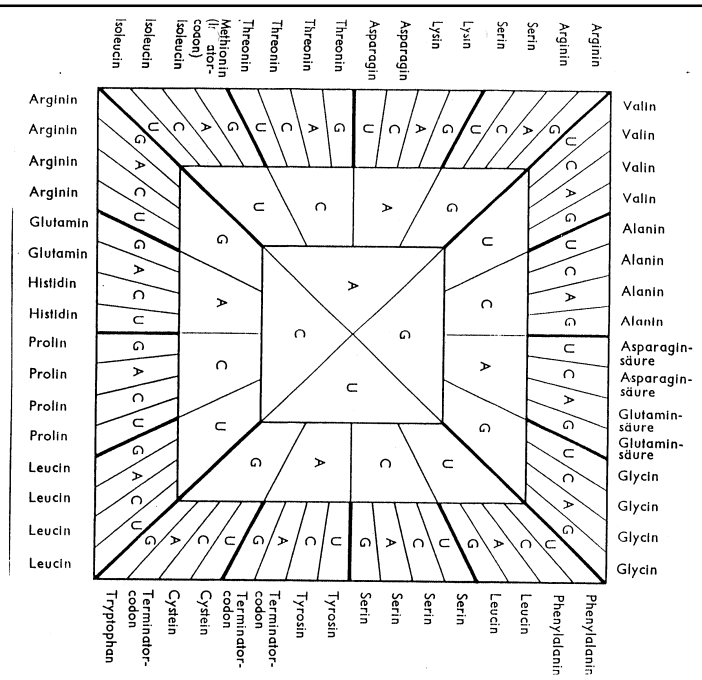
} 1 Buchstabe = 2 bit

Die Wörter:
CTG ~ 6 bit
GAT
•
•
•
64

Aminosäuren:
Leuzin
Asparagin
•
•
•
20

„Universeller genetischer Code“
genetische Sprache ≠ Genom (Wort der Sprache)
Deutsch ≠ Faust
Zucker & Insulin: Erbgut o.k. Fabrik defekt

34



Umfang des menschlichen Erbgutes

23 Chromosomen ≈ 100.000 Gene (Schätzungen)
1 Chromosom ≈ 5.000 Gene
1 Gen ≈ 1.000 Wörter
= 3.000 Buchstaben
= 1 Seite Text
= 1 Kochrezept

10^5 Gene • 3 • 10^3 Buchstaben = 3 • 10^8 Buchstaben

Umfang gesamtes Genmaterial:

1 Mill. Seiten Text = 3 Milliarden Buchstaben
= 23 Bände (Chromosomen)
je 50.000 Seiten mit 5.000 Rezepten

10 x größer ?? als informationstragendes Erbgut?

36

Variationen und Fehler

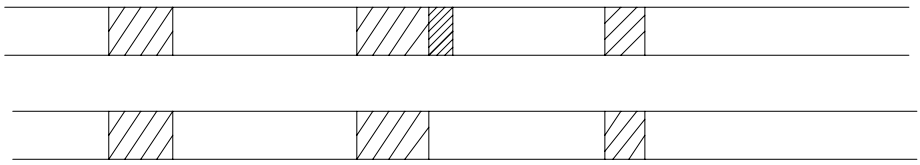
99.9 % gleich

– Kopierfehler
– Zeile fehlt, doppelt
– Seiten vertauscht
– Seite fehlt, doppelt
– Seite zerrissen

fehlerhafte Gene haben andere Länge als richtige
⇒ Längenvergleich

37

Vererbung:
Chromosomen von Vater



Chromosomen von Mutter

falsches Rezept für Protein,
z.B. für Insulin

Probleme
- wo?
- ausschneiden
- Längenvergleich

Sonde = identifizierender kleiner Text-Ausschnitt ~ Positions.Ident.
Enzym = genetische Schere
Southern Blot = Längenvergleich

38

Genom-Projekt

3 Milliarden Buchstaben
1 $ pro Buchstabe : A C T G

⇒ 3 Milliarden $ = 6% des UMTS Erlöses
2005 fertig !!! (war Schätzung 1996)
massiver Einsatz von Robotern, Sommer 2000

heute schon:
Hefesequenzierung
Bakterienstammbaum

2020: individuelles Genoprogramm auf 1 CD

39

Soziale Folgen
- Erkennung
- Vermeidung
- Auswahl gesunder Embryos
- Wunschkinder ?

Erbkrankheiten
1983 Huntington
1987 Muskeldystrophie
1989 zystische Fibrose
Hypercholestrolämie
Dickdarmkrebs
Brustkrebs
Alzheimer
Multiple Sklerose
Diabetes
Schizophrenie
Alkoholismus
kriminelles Verhalten ...

40

Zukunft: 2020 Genogramm

- Berufswahl
- Einstellung durch Arbeitgeber
- Versicherungen
- Porsche Fahrer?
- vorbeugende Medizin (Medikamente ?)
- Partnerwahl

•
•
•

41

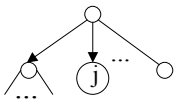
Kap. 4.4 Repräsentation von Positions-Bäumen

1. Geflecht

Marke 1 Byte
Zeiger 4 Bytes
} intern
+ Pos # 4 Bytes für Blatt

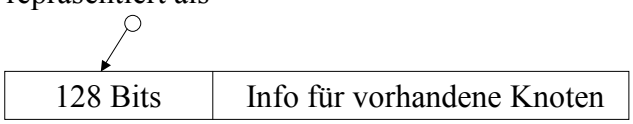
$\Rightarrow$ Expansionsfaktor 9 über Text

2. Bitvektoren + Geflecht



Bis zu 64 Söhne:
i interne
b Blätter

repräsentiert als



42

00 : nicht vorhanden	0
10 : Blatt	10
11 : interner Knoten	11

- Marke codiert durch Position in Bit Vektor
- für internen Knoten: Zeiger auf Bit Vektor für alle Söhne
- für Blatt: Positionsnummer

Speicherbedarf: für Knoten mit i internen Söhnen und b Blatt-Söhnen

Geflecht	Bitvektoren
$i \cdot 5 + b \cdot 9$	$4 + 16 + b \cdot 4$

$4 + 16 + b \cdot 4$: 16 └ Bitvektor, 4 └ Zeiger

Bitvektoren besser?

$$\geq$$

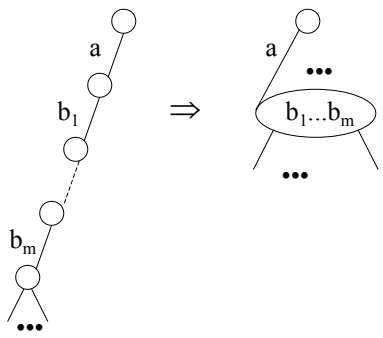
$$i \cdot 5 + b \cdot 5 \geq 20$$

$$i + b \geq 4$$

43

Hypride Repr.: lokal entscheiden,
1 Zusatzbit pro Knoten

Weitere Idee: Pfadkontraktion



d.h. jetzt haben alle Knoten eine Verzweigung

$$\geq 2$$

darüber Info in internen Knoten

44

Nochmal Bitvektoren und Geflecht

Vorschlag: Art des Knotens nicht in Bitvektor kodieren, sondern bei Knoten selbst, 1 Bit provorhendenem Knoten, z.B. Zeiger mit 31 Bits anstatt 32

00 11 10 11 ... $\Rightarrow$ | 0111 ... $z_1 1, z_2 0, z_3 1, ...$ |

Problem: Zugriff auf Info für vorhandene Knoten! Vorher war Direktzugriff auf Knoteninfo anhand des Bitvektors bestimmbar.

Lösung: alle vorhandenen Söhne sequentiell abarbeiten

Speicherbedarf:

Geflecht	Bitvektoren
$i \cdot 5 + b \cdot 9$	$4 + \underline{\mathbf{8}} + b \cdot 4$

Vektoren besser?

$$i \cdot 5 + b \cdot 9 \geq 12 + b \cdot 4$$

$$i + b \geq \frac{12}{5}$$

45