



# Linear time algorithms for finding and representing all tandem repeats in a string

Dan Gusfield, Jens Stoye (1998)



**Aktuelle Themen der Bioinformatik**

Seminarleitung : Dirk Metzler  
Linn Himmelmann

Vortrag : Frank Abromeit

# Gliederung

- Motivation
- Begriffe, Notation
- Algorithmus
- Ausblick



# Was sind Tandem-Repeats ?

- Tandem Repeat ( $\alpha\alpha$ )

**aabaab =  $\alpha\alpha$**

( $\alpha = \text{aab}$ )

- Tandem Array ( $\alpha\alpha$ )<sup>n</sup>

# Motivation

- Menschliches Genom besteht zu etwa 10-30 % aus *Tandem Repeats*
- Vergleich Pflanzen:  
Genom der *P. coccineus* 65% TRs  
(Nagl et al. 1983).

# Wo findet man Tandem Repeats ?

- Nichtkodierende *TRs*

(*TRs* zwischen oder auch innerhalb (Introns) von Genen)

Beispiel: Telomere enthalten *TRs* (Funktion Chromosom-Replikation)

- Kodierende *TRs*

sind z.B. Gene für ribosomale RNA, Transfer-RNA und Histone (Kahl 1995)

# Funktion

- Ursprung und Funktion der *TRs* ist nicht vollständig bekannt
- Als Funktion nimmt z.B. an :
  - Gen-Regulation
  - Evolution
  - Protein-Bindung

# Eigenschaften von Tandem Repeats

- Anzahl der Repeats ist oft *polymorph*, (*heterozygot*)  
d.h. von Organismus zu Organismus verschieden.
  - Deshalb charakteristisch
- Gut als genetische Marker einsetzbar
  - DNA – Fingerprinting (Jeffreys et al. 1985)
  - Genetische Karten
    - „A comprehensive map of the human genome  
based on 5264 Mikrosatelliten“ Dib,C. et. Al. (1996)

## *Variation entsteht durch*

- Reproduktionsfehler  
(Bei langen Ketten hohe Wahrscheinlichkeit für Fehler)
- Evolutionsbedingt (Mutation)
- Durch äußere Einflüsse  
(z.B. Strahlenbelastung nach  
Atombombentests) Science 295, 946/1037 (2002).

- **Krankheiten**

- Bestimmte TR Häufigkeiten erscheinen im Zusammenhang mit Krankheiten

### **Trinucleotide Expansion Disorders**

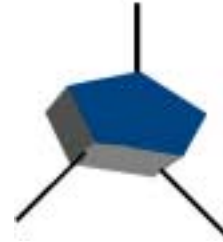
Name of disorder	Repeat pattern	Normal Copies	Mutant copies
DRPLA	CAG	6-35	49-88
Huntington's	CAG	11-34	40-120
Kennedy's	CAG	9-36	38-62
Type1 SCA1	CAG	25-36	41-48
Type2 SCA2	CAG	15-24	35-59
Friedreichs Ataxia	CGG	6-54	>100
Myotonic Dystr.	CTG	5-27	50-1000

(aus „Techniques for Detecting Approximate Tandem Repeats in DNA“,Thao, Tran T. et Al)



# Klassifikation

- *Satelliten* (100-1000 bp) (**short-tandem-R.**)
  - Länge bis 100k bp
- *Minisatelliten* (15-100 bp)
  - stark variabel am selben locus (heterozygot) **VNTR**
  - DNA Fingerprinting
- *Mikrosatelliten* (bis 4 bp) **short-tandem-Repeats**
  - 0,5 % des menschlichen Genoms
  - keine bekannte Funktion



# **Beispiel: Minisatelliten DNA**

*Human 33.1 minisatellite (62 bp)*

AAGGGTGGGCAGGAAGTGGAGTGTGTGCCTG  
CTTCCCTTCCCTGTCTTGTCCTGGAAACTCA

*Human 33.5 minisatellite (17 bp)*

YGGGCAGGAGGGGGGAGG



# Algorithmen für TRs

- Probabilistische (häufig)

„Finding Approximate TRs in Genomic Sequences“, Wexler Y. et Al.

„An algorithm for finding TRs of unspecified pattern size“, Benson G.

- String-Algorithmen (seltener)

„Finding maximal repetitions in a word in linear time“

Kolpakov, K. , Kucherov, G. (1999)

- dito

# Vorgestellter Algorithmus

- Schnellster bekannter Alg. für *perfekte* TRs, Laufzeit:  $O(n)$ 
  - sensibel für mutierte Tandem Arrays und Fehler in der Eingabe
- Stochastische Algorithmen erkennen mutierte Arrays

# Komplexer Algorithmus

Suffix-tree

Lempel-Ziv  
Dekomposition

Longest-common  
Extension

Algorithmus

Suffix-Walk

**Ergebnis:** Markierter Suffixbaum

# Algorithmus 3 Phasen

- *Phase I*

Finden der *left-most-covering-set*

- *Phase II*

Markieren einer Teilmenge aller TRs im Suffix-Baum von S

- *Phase III*

Markieren aller TRs des Vokabular von S



# Longest-Common-Extension

- Def. die Längste gemeinsame Erweiterung von Indizes  $i$  und  $j$  in  $S$  als
  - Länge des längsten gemeinsamen Teilstrings in  $S$  der an der Position  $i$  beginnt, und mit einem Teilstring, der an der Position  $j$  beginnt übereinstimmt (vorwärts)
  - Länge des längsten gemeinsamen Teilstrings in  $S$  der an der Position  $i$  endet, und mit einem Teilstring, der an der Position  $j$  endet übereinstimmt (rückwärts)

Aufwand : Berechnung für alle Positionen:  $O(n)$ , Recall in  $O(1)$

# Notation 1

- $S$  = String über einem Alphabet  $\Sigma$
- $S[i..j]$  = Infix von  $S$
- Menge aller Tandem Repeats in  $S$   
 $M = \{(i, l) \mid (i, l) \text{ ist TR in } S\}$

...aabaab...  $\leftrightarrow$  (i,6)

i

# Notation 2

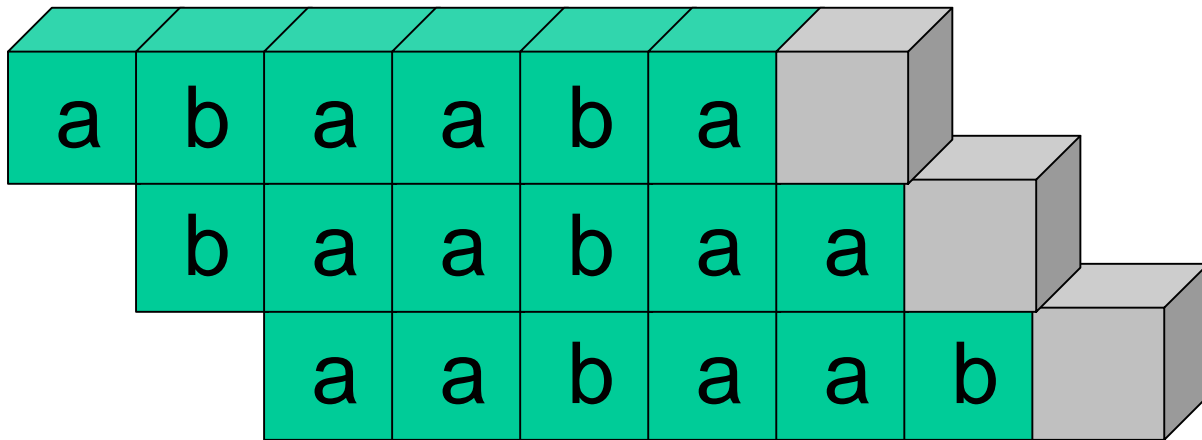
- Zwei TRs  $(i,l)$  und  $(j,l)$  heißen gleich, wenn sie denselben String bezeichnen
- Vokabular der TRs  
 *$\{(i,l) \mid (i,l) \text{ ist TR in } S, \text{ und } (i,l) \text{ ist ungleich jedem anderen TR in } S\}$*

# Notation 3

- $(i,1)$  überdeckt (*covers*)  $(k, 1) \iff [i..j]^1 \wedge i < j \leq k$

# Def.: Run von Tandem-Repeats

- Sei  $S = \text{abaabaabbaaabaaba}$



Notation  $[1..3]^6$

# Phase I

## (Leftmost-covering-set)

Beispiel:

Sei  $S = \text{abaabaabb}$

Vokabular =  $\{aa,bb,aab,baabaa,abaaba\}$

Menge aller TRs =  $\{(1,2),(2,6),(3,6),(4,2),(4,6),(7,2),(9,2)\}$

Leftmost-covering-set =  $\{(1,6),(9,2),(7,2)\}$

# Lempel-Ziv-Dekomposition I

$L_i$  = Länge des maximalen Präfix von  $S[i..N]$

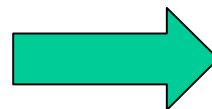
Für den es ein  $j < i$  gibt mit  $S[j..n] = S[i..m]$

$S$	a	b	a	a	b	a	a	b	b	a	a	a	b	a	a	b	a	\$
$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$L_i$	0	0	1	5	4	3	2	1	3	2	6	6	5	4	3	2	1	0

**Beispiel :** Für  $i = 8$  ist  $b$  maximal, denn

der Teilstring  $bb$  links der Position 8 nicht mehr vor.

Da die *Länge* von  $b = 1$



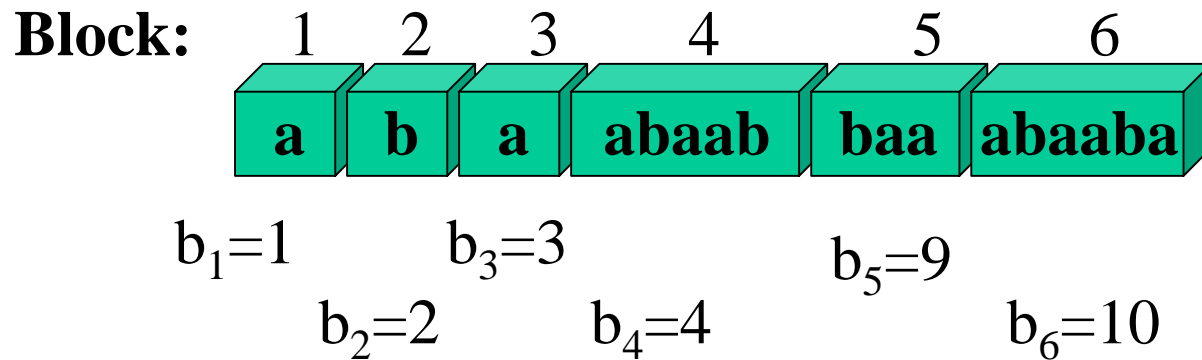
$$L_8 = 1$$

# Lempel-Ziv-Dekomposition II

$S$	a	b	a	a	b	a	a	b	b	a	a	a	b	a	a	b	a	\$
$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$L_i$	0	0	1	5	4	3	2	1	3	2	6	6	5	4	3	2	1	0

$S$  wird zerlegt:  $b_1 := 1, b_{i+1} = b_i + \max(1, L_{b_i})$

( $b_i$  ist die Anfangsposition eines Blocks in  $S$ )

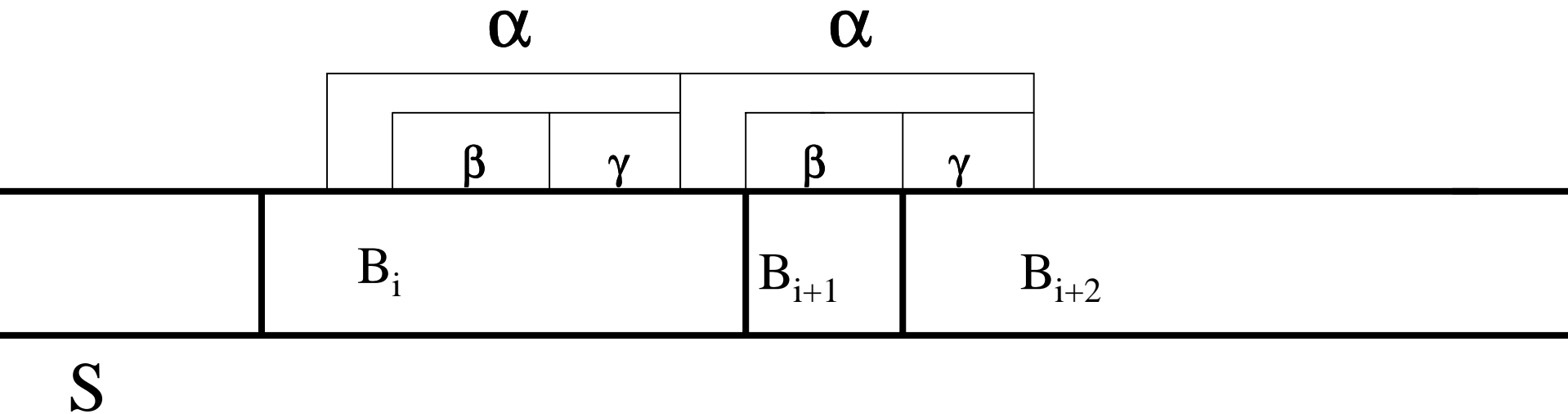


# Lempel-Ziv-Dekomposition

- *Blöcke* machen eine allgemeine Aussage über Strings, die mehrmals in  $S$  vorkommen.
- Das Aufstellen jetzt folgender *Theoreme* und *Lemmata* hat das Ziel gefundene Blöcke der LZ-Dekomposition mit *TRs* in Verbindung zu bringen !

# Lemma 1

Die rechte Hälfte von  $\alpha\alpha$  erstreckt sich über höchstens zwei Blöcke.



# Lemma 2

- Das linkeste *Vorkommen* eines TR erstreckt sich über mindestens 2 Blöcke.

# Theorem 2

Wenn das linkeste Vorkommen von  $\alpha\alpha$  seine Mitte in Block B hat,  
Dann gilt entweder :

$\alpha\alpha$  hat sein linkes Ende im Block B, und sein  
Rechtes Ende in Block B+1 (Bedingung 1)

oder

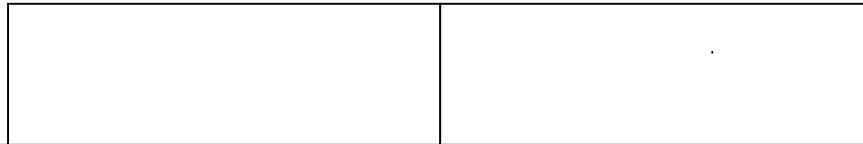
Das linke Ende von  $\alpha\alpha$  erstreckt sich in den Block  
B-1 und ev. weiter links. (Das rechte Ende kann  
In B oder B+1 liegen. (Bedingung 2)

# Theorem 2

Bedingung 1

$\alpha$

$\alpha$



$B_i$

$B_{i+1}$

# Theorem 2

Bedingung 2a

$\alpha$

$\alpha$



$B_i$

$B_{i+1}$

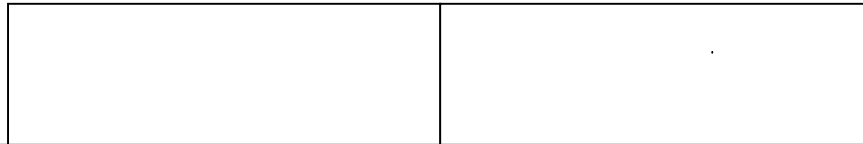


# Theorem 2

Bedingung 2b

$\alpha$

$\alpha$



$B_i$

$B_{i+1}$

# Algorithmus 1a

Für jeden Block B:

Für  $k = 1$  to  $|B|$  do

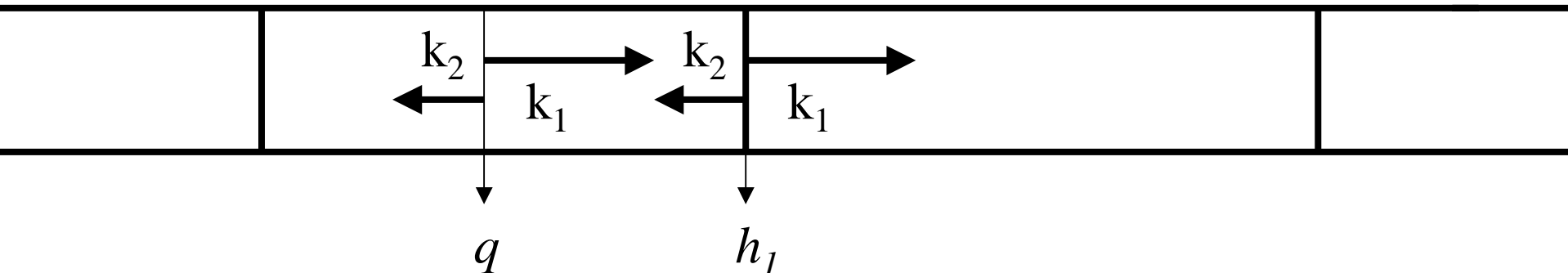
begin

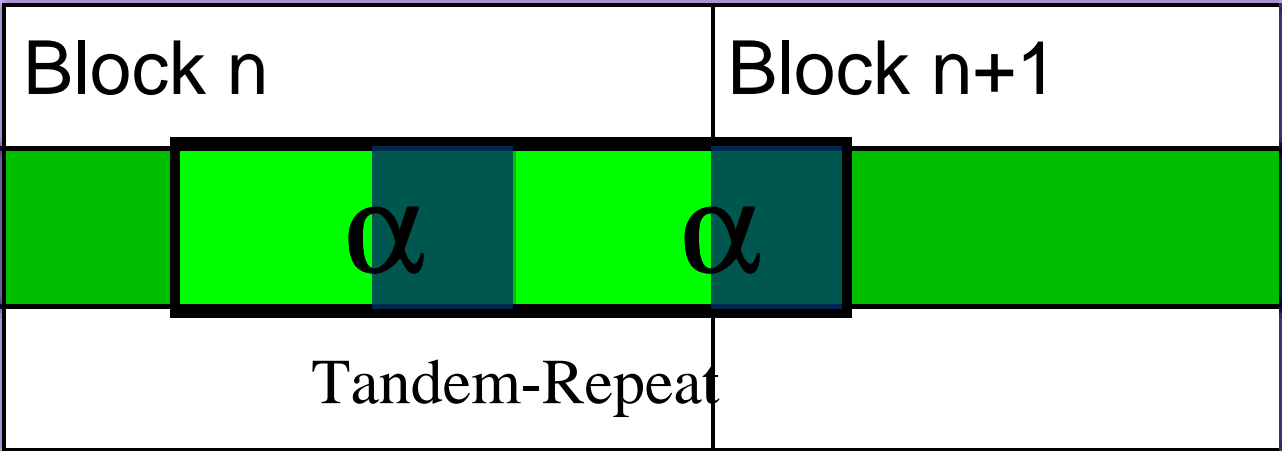
$q = h_1 - k$

1. Berechne die longest-common-extension (vorwärts) von  $h_1$  und  $q$ . Sei  $k_1$  = die Länge der Extension
2. Berechne die longest-common-extension (rückwärts) von  $h_1$  und  $q$ . Sei  $k_2$  = die Länge der Extension
3. Wenn  $k_1 + k_2 \geq k$  und  $k_1 > 0$  dann:  
Gebe den TR :  $(\max(q - k_2, q - k + 1), 2k)$  aus.

B

B+1





# Algorithmus 1b

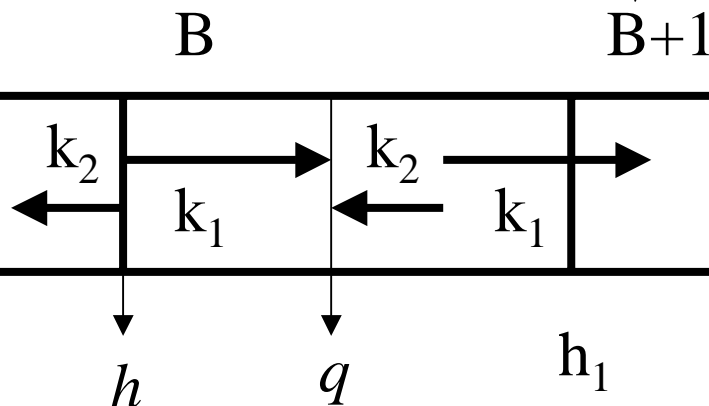
Für jeden Block B:

Für  $k = 1$  to  $|B| + |B+1|$  do

begin

$q = h + k$

1. Berechne die longest-common-extension (vorwärts) von  $h$  und  $q$ . Sei  $k_1$  = die Länge der Extension
2. Berechne die longest-common-extension (rückwärts) von  $h-1$  und  $q-1$ . Sei  $k_2$  = die Länge der Extension
3. Wenn  $k_1 + k_2 \geq k$  und  $k_1 > 0$ ,  $k_2 > 0$  und  $\max(h-k_2, h-k+1) + k < h_1$  dann  
Gebe den TR :  $(\max(h-k_2, h-k+1), 2k)$  aus.



# Ergebnis von 1a),1b)

- Ausgabe von Alg. 1a + 1b sind Listen  $P(i)$ , die alle *TRs*, die an Position  $i$  beginnen enthalten.

Lemma 3:

Die Listen können absteigend (nach Länge der *TRs*) sortiert gehalten werden, ohne die Laufzeit zu verschlechtern.

# Aufwand

- Phase I  
     $O(n)$

# Phase II

- Markieren der Endpunkte von einigen *TRs* im Suffix-Baum von *S* durch *bottom-up traversal* in  $O(n)$ .

1. Für jedes Blatt  $v$  mit Vaterknoten  $u$

2. Wiederhole solange  $P(v)$  kürzer wird:

Sei  $(i,l)$  das Paar am Kopf der Liste  $P(v)$

Falls  $l > \text{Tiefe}(u)$ , d.h.  $(i,l)$  endet am Knoten  $v$  oder auf der Kante  $(u,v)$

1. Schreibe  $l - \text{Tiefe}(u)$  an die Kante  $(u,v)$ , um aufzuzeichnen, daß  $(i,l)$   $l - \text{Tiefe}(u)$  Zeichen vom Knoten  $u$  entfernt endet.

2. Entferne  $(i,l)$  aus  $P(v)$

Gebe  $P(v)$  an den Vaterknoten  $u$  von  $v$  zurück. Setze  $v := u$  und Gehe zu 2.

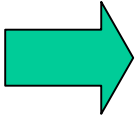
# Phase II


- Es werden erst alle Kinder  $v$  von  $u$  bearbeitet
- Der Vaterknoten  $u$  erhält die Liste  $P(i)$  des Kindes mit dem kleinsten Index. (Index = Startposition in  $S$ )  
Die anderen Listen enthalten nur TRs, die überdeckt sind.  
Sie können deshalb nicht *left-most* sein

# Phase III

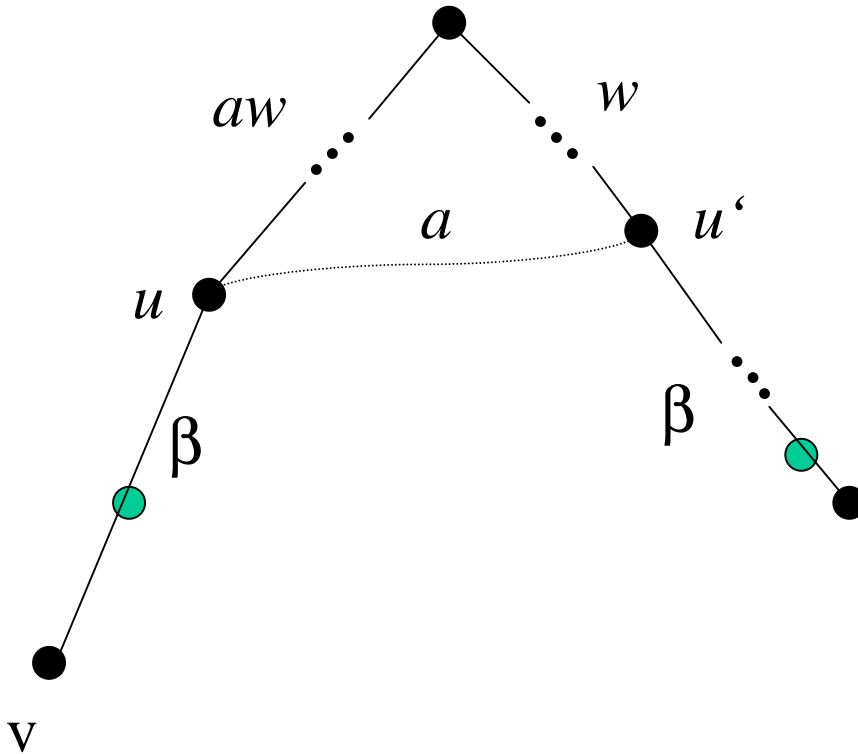
- Markieren des vollständigen Vokabulars im Suffix-Baum von  $S$  mit Suffix-Walk
- Trick:

$\alpha\alpha = a b b a a b b a$

$\alpha\alpha = a b b a a b b a$    $\alpha'\alpha' = b b a a b b a a$



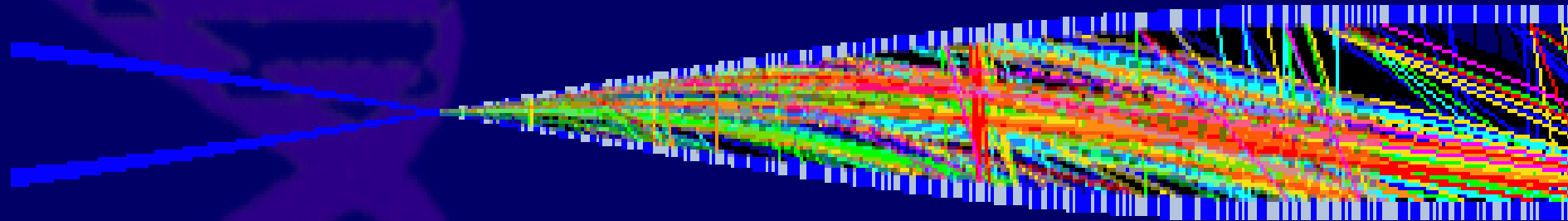
# Suffix-Walk



# Ausblick & Anwendungen

- Tools im Web (meist Alignment)
  - Einschränkungen
  - Sequenzlänge (max. 5 M bp)
  - Keine Aussagen über asymptotische Laufzeit
  - Laufzeit im Sekunden bis Minutenbereich
- Tandem-Repeat Datenbanken

# *REPuter* - The Repeats Computer



Are you looking for...

- Whole genome comparison?
- Sequence duplications?
- Palindromic structures?
- Degenerate repeats?
- ORFs in Transposons?
- EST Clustering?