

# Bioinformatik

Zeichenketten und Stringalgorithmen



Silke Trißl / Ulf Leser

Wissensmanagement in der  
Bioinformatik



# Inhalt dieser Vorlesung

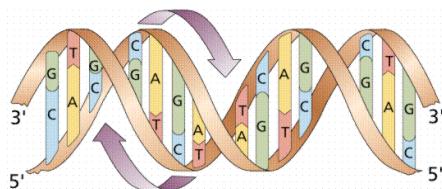
---

- Warum Stringmatching?
- Strings und Matching
- Naiver Algorithmus

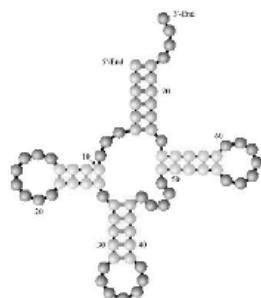
# Biomoleküle

---

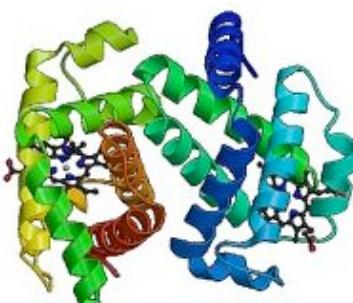
- DNA, RNA und Proteine lassen sich als Zeichenkette über festem Alphabet darstellen



**DNA**  
A C G T



**RNA**  
A C G U

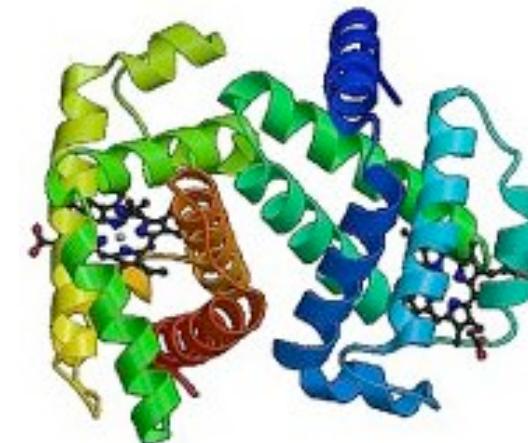
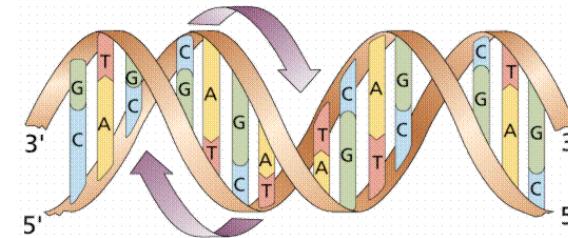


**Protein**  
A C D E F G H I K L M N P Q R S T V W Y

# Sequenz - Funktion

---

- DNA
  - Vererbung
  - Genotyp
  - Produktion von Proteinen
- Proteine
  - Phänotyp
  - Struktur
  - Bindungsverhalten



# Zwei Anwendungen

---

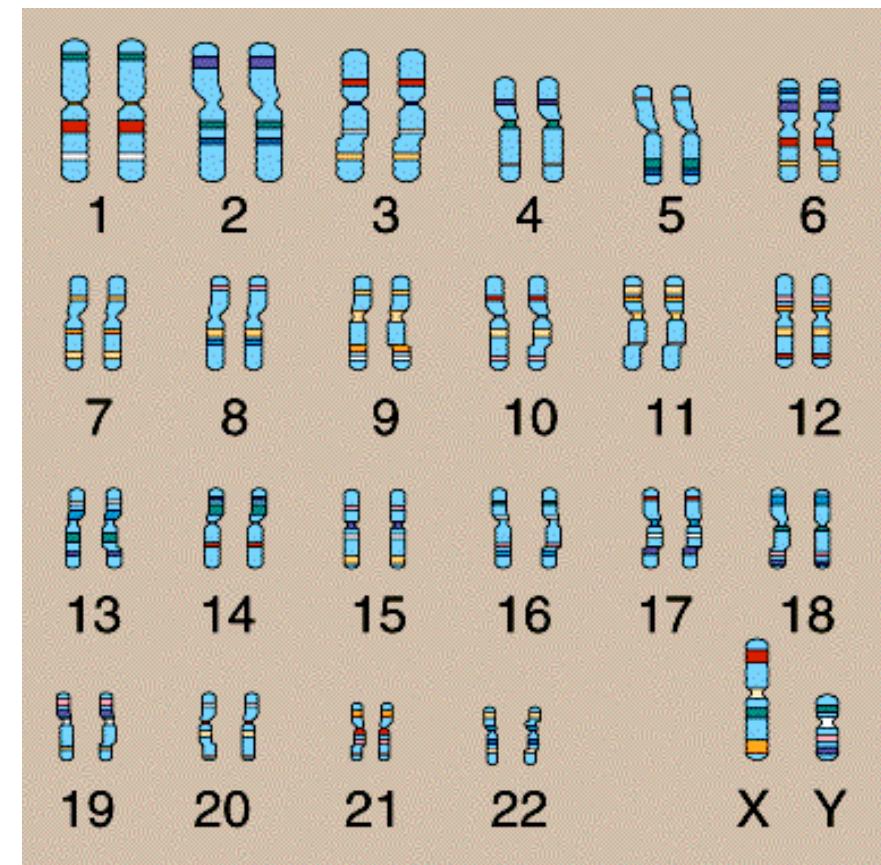
- Sequenzierung
  - Assembly von Teilsequenzen
- Funktionale Annotation
  - Schnelle Suche in Sequenzdatenbanken

# 1. Das menschliche Genom

---

- ... AGGCTGTGGATTAGAGACC ...

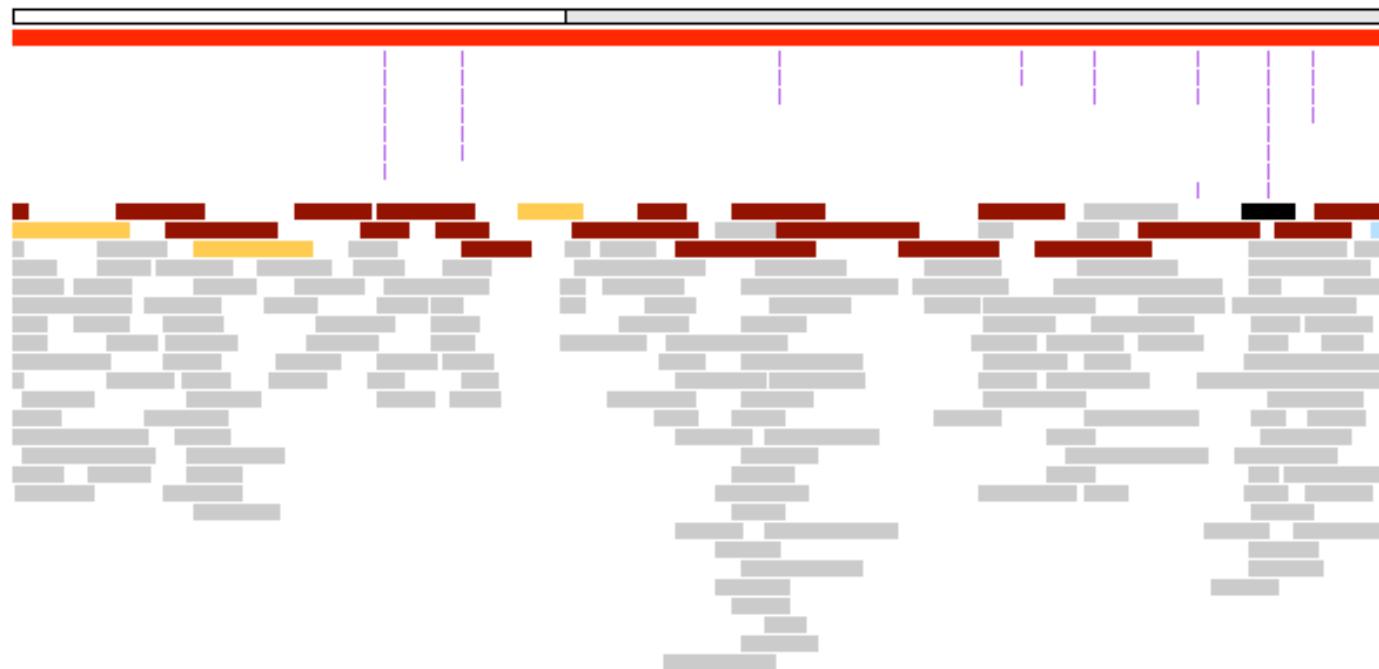
- 23 Chromosomenpaare
- ~ 3.000.000.000 Buchstaben
- Chromosomen kann man nicht direkt sequenzieren
- Sequenzierbar: 500-1000 Basen



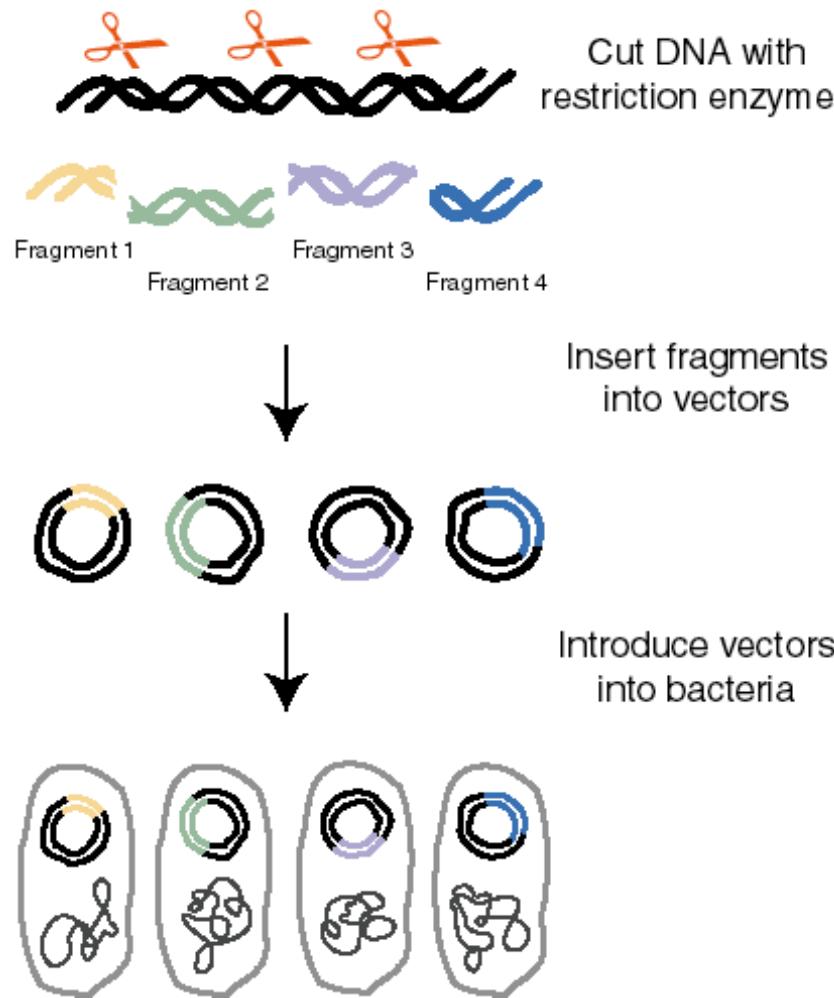
# Mapping und Sequenzierung

---

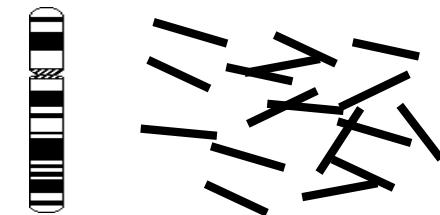
- Zerlegung in Bruchstücke (Clonierung)
- Berechnung aller Überlappungen
- Bestimmung der wahrscheinlichsten Gesamtsequenz
- Variante: Bestimmung des **Minimum Tiling Paths**



# Clonieren



- Schneiden des Chromosoms mit Restriktionsenzymen
  - Länge hängt ab von
    - Spezifität des Restriktionsenzyms
    - Länge der Behandlung (partieller Verdau)
- Bruchstücke unterschiedlicher Länge
- Auftrennen nach Länge
  - Gelelektrophorese
- Clonierung in Bakterien
  - Vervielfältigung
- Ergebnis:



# Sequenzierung

---

- Gegeben: Clone unbekannter Sequenz
  - Gesucht: Sequenz
- 
- Unmöglich: Ansehen, Messen, Mikroskop, etc.
  - Verfahren von Sanger, 1972:  
**Radioactive Dideoxy Sequencing**

# Heute

---

- Fluoreszente Markierung
- Hochdurchsatz
- Sehr billig



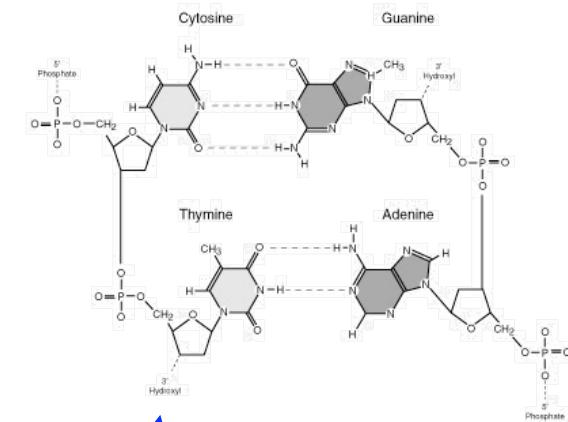
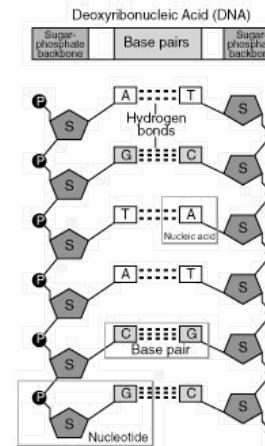
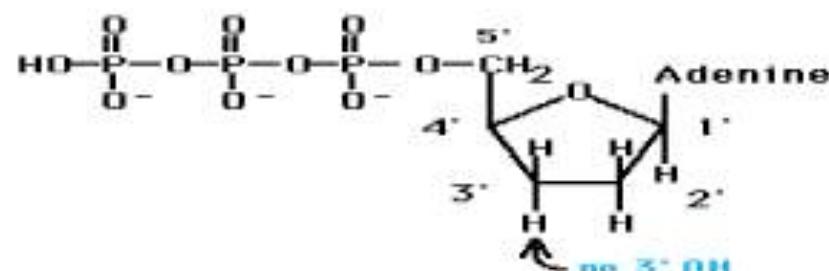
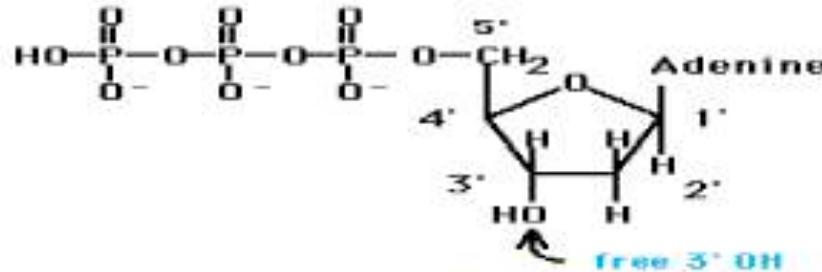
Quelle: <http://www.geneticsplace.com>

# Sequenzierung nach Sanger

---

- Voraussetzungen
  - Sequenz hat einen definierten Anfang
    - Teil des Clonierungsvektors
    - Dient als Bindungsstelle für Primer
    - Sequenzierungsanfang
  - Polymerase
    - Bindet an doppelsträngigem Abschnitt
    - Verlängert einsträngige DNA entlang des Templates
- Deoxy versus Dideoxy Nucleotide
  - DNA besteht aus Deoxy Nucleotiden (dNTP)
  - Einbau von Dideoxy Nucleotiden (ddNTP) möglich
  - ddNTP stoppt Polymerase

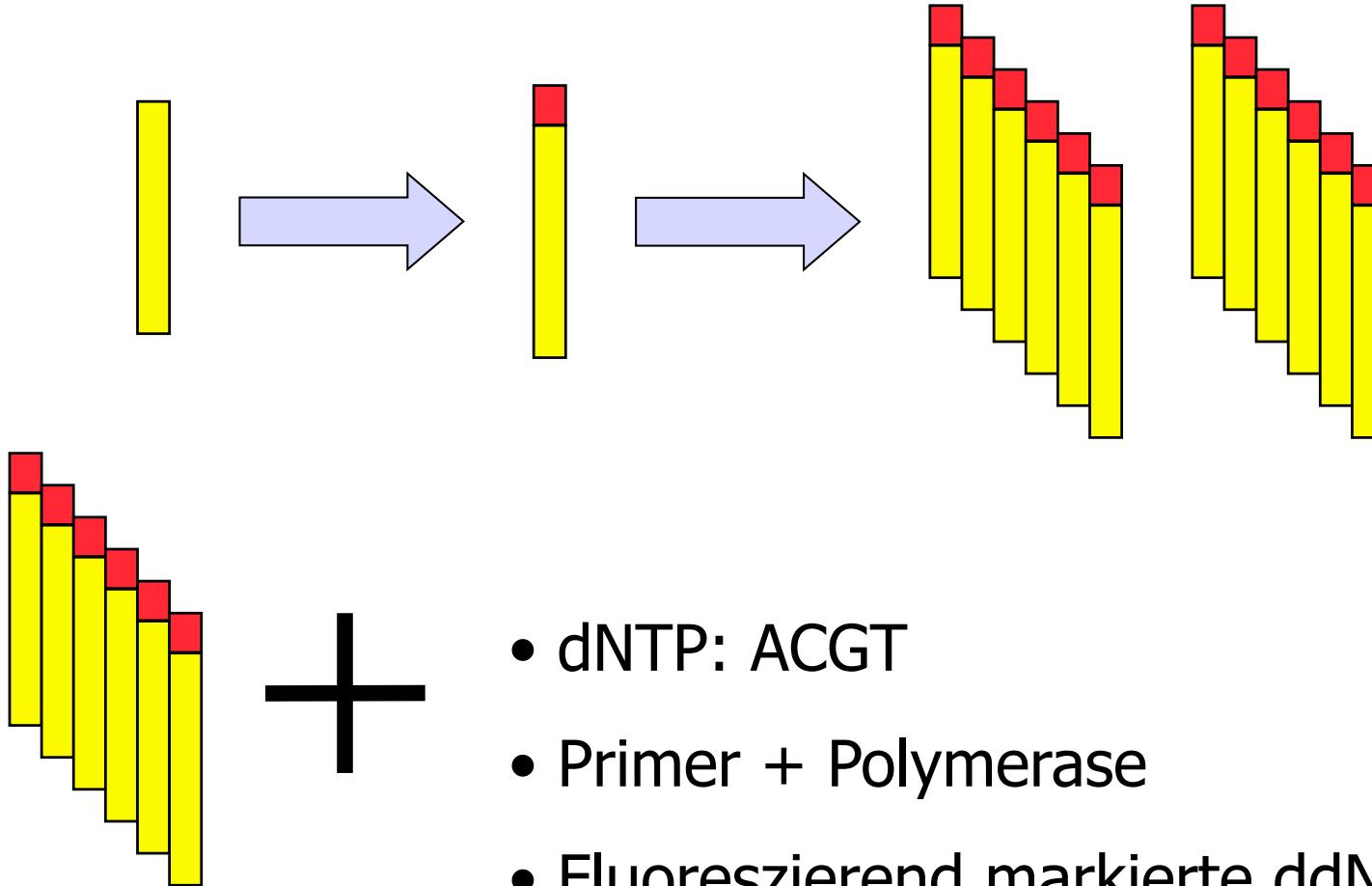
# dNTP versus ddNTP



- Dideoxy-Base: keine freie OH Gruppe
- Bei Einbau einer Dideoxy-Base durch Polymerase
  - Können keine weiteren Basen mehr angehängt werden
  - Polymerase fällt ab

# Schritt 1 und 2

---



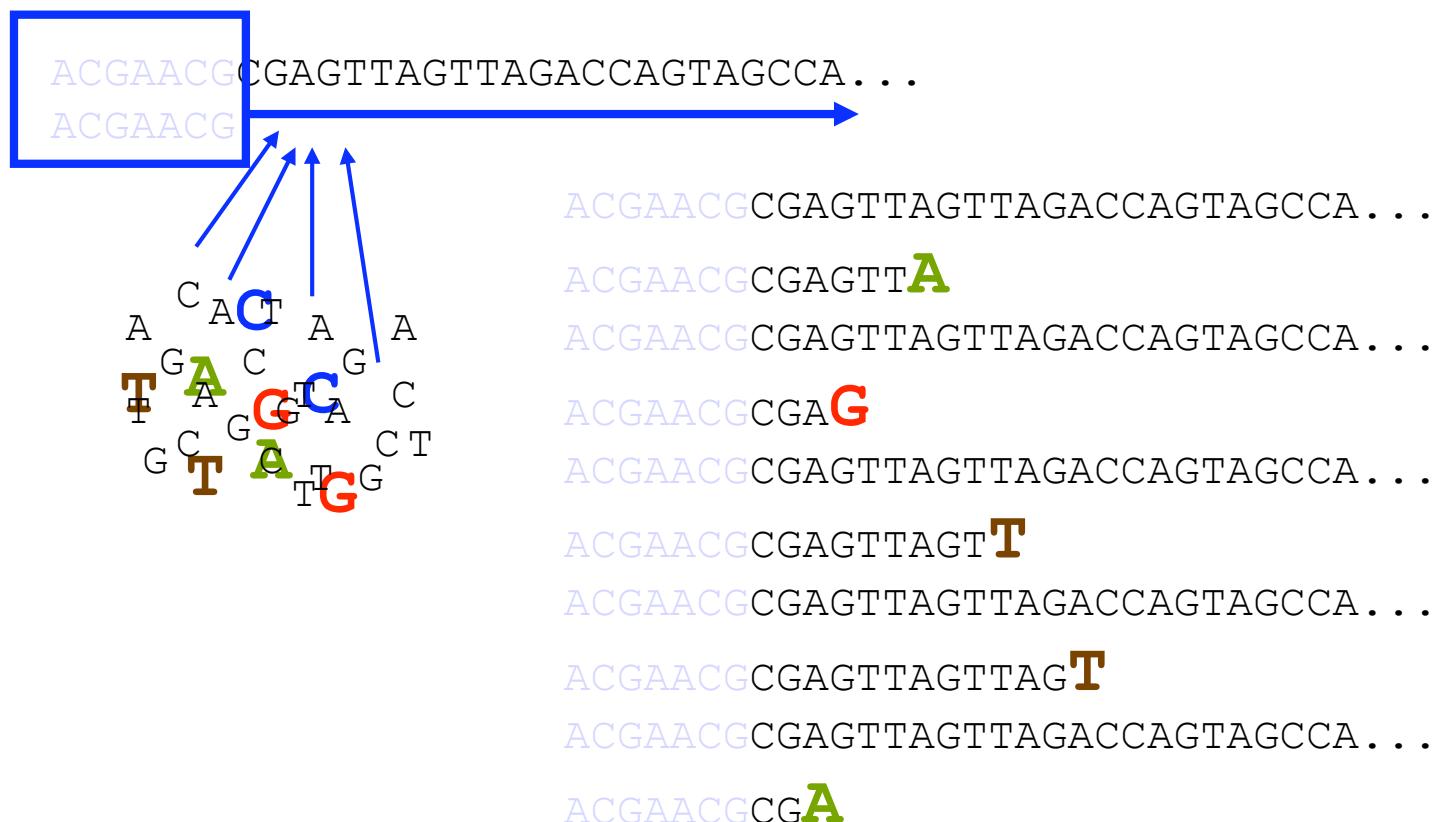
# Schritt 3

## Primer

ACGAACGCGAGTTAGTTAGACCAGTAGCCA...

# Template

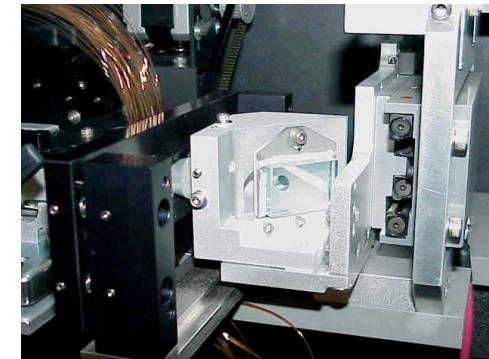
# Polymerase



# Schritt 4

---

## Laser & Detektoren



ACGAACGCGAGTT**A**  
ACGAACGCG**G**  
ACGAACGCGAGTTAGT**T**  
ACGAACGCGAGTTAGTTAG**T**  
ACGAACGCG**A**

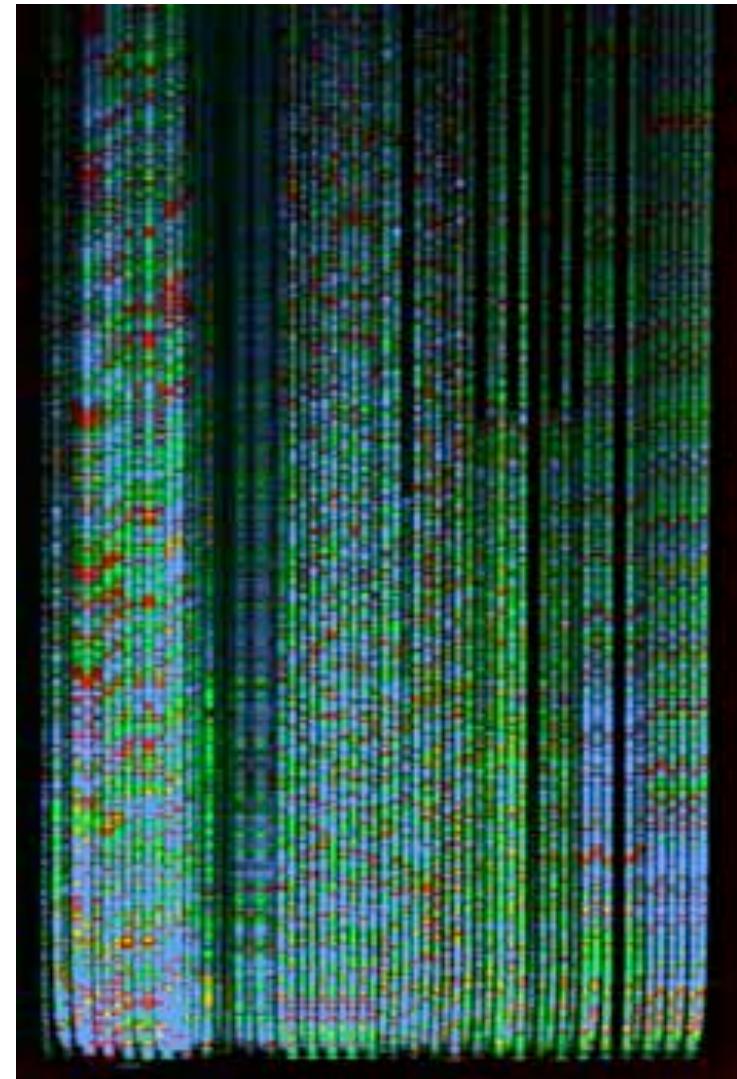
## Gel / Kapillar Elektrophorese

ACGAACG**C**  
ACGAACG**G**  
ACGAACGCG**A**  
ACGAACGCGA**G**  
ACGAACGCGAG**T**  
ACGAACGCGAGT**T**  
ACGAACGCGAGTT**A**  
ACGAACGCGAGGTTA**G**

# Ergebnis (roh)

---

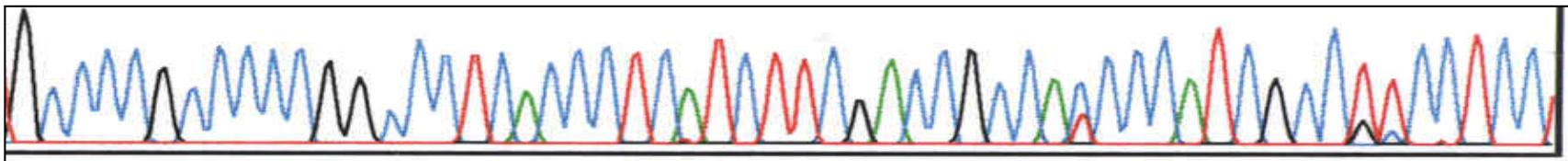
- Heutige Geräte
  - >36 Läufe parallel
  - Kapillarelektrophorese (statt Gelen)
  - Direktes Laden von 96 Quellen
- Sanger
  - Radioaktive Markierung
  - 4 Mischungen (A,G,T,P)
  - 4 Gele (Linien)



# Ergebnis (Zwischenprodukt)

---

- Signalverarbeitung (Rauschen, ...)



- Übersetzung in **Traces**
  - 4 Arrays, jedes für eine Farbe
  - Intensitätswerte in regelmäßigen Zeitabschnitten
- Theoretisch
  - Peaks entdecken
  - Immer nur eine Farbe
  - Sequenz zuordnen

# Vom Tracefile zur Sequenz

---

- Tracefiles sind Rohdaten der Sequenzierung
- Verschiedene Verfahren / Tools, um aus Tracefiles Sequenzen zu berechnen
- Komplexe Probleme
  - Base Calling
  - Assembly
  - Finishing

# Assembly

---

- Gegeben: eine Menge von überlappenden Reads
- Aufgabe: Finde die **wahrscheinlichste Ursprungssequenz**
- Redundanz ist
  - Notwendig: Verbindung von Teilstücken nur durch Überlappungen
  - Konflikträchtig: Widersprüche, Mehrdeutigkeit

Fehler ?

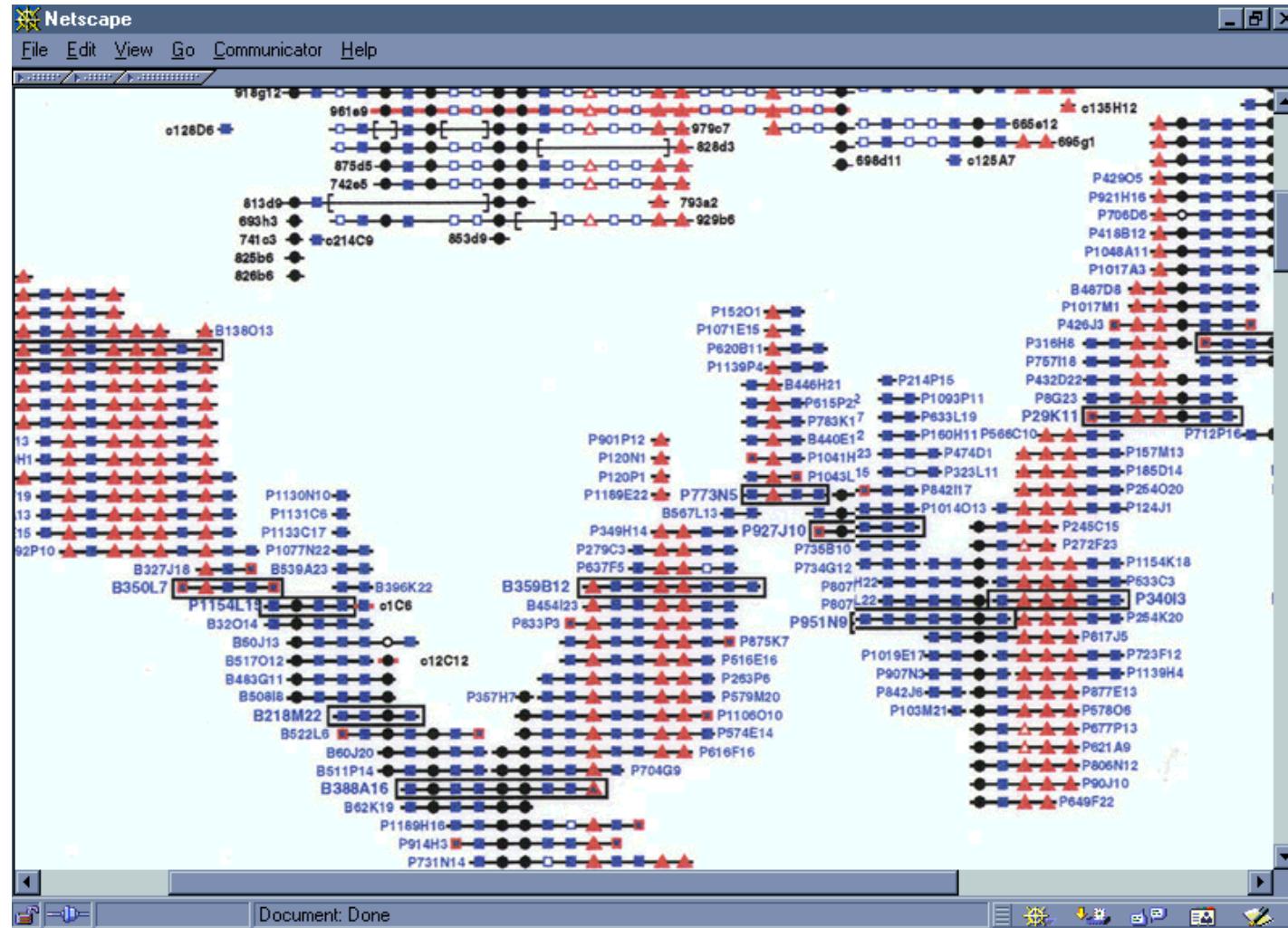
tggacaagcaa~~gatt~~  
acatttt~~t~~ttgaac  
gcaa~~a~~gatt~~g~~ttg  
  
tg~~g~~acaagcaaagatt~~a~~  
acatttt~~g~~aac  
gcaaagat~~t~~ttg

# Abstrakte Formulierung

---

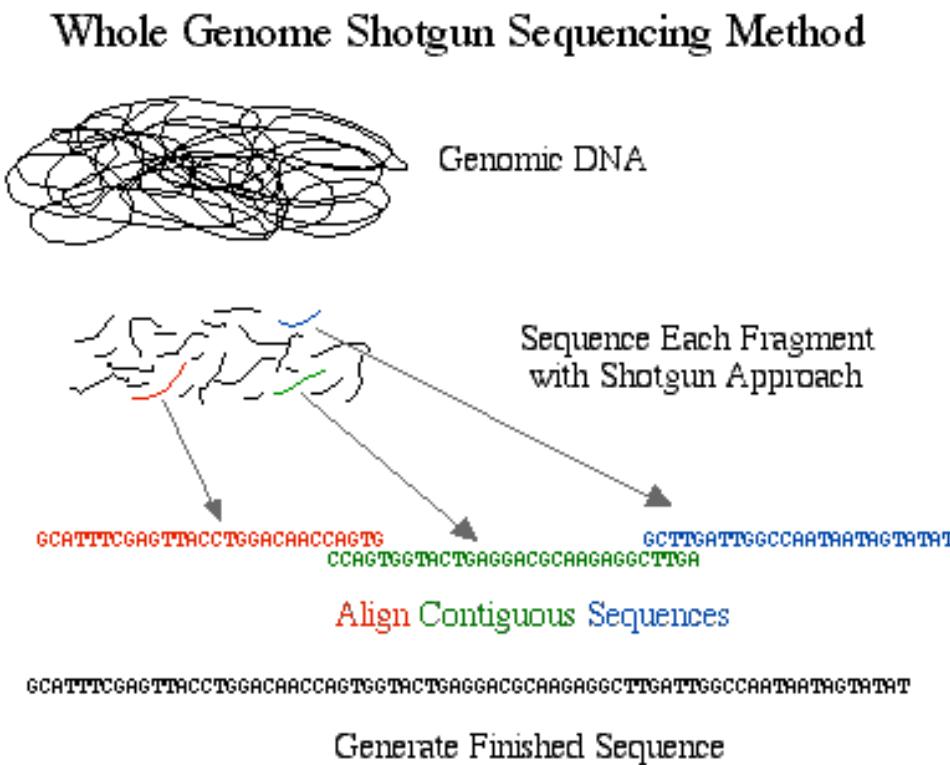
- **SUPERSTRING**
  - Geg.: Menge  $S$  von Strings
  - Ges.: String  $T$  so, dass
    - (a)  $\forall s \in S: s \in T$  (s Substring von  $T$ )
    - (b)  $\forall T'$ , für die (a) gilt, gilt:  $|T| \leq |T'|$  (  $T$  ist minimal)
  - Sehr schwieriges Problem
    - Exponentielle Komplexität
    - „NP-vollständig“
- **Assembly: Verschärfungen von SUPERSTRING**
  - Fehler in Sequenzen (s „ungefähr Substring“ von  $T$ )
  - Zwei Orientierungen von  $s$  möglich
- **Kernproblem**
  - Finde alle **Überlappung** zwischen Paaren von Reads

# Resultat



# Problemdimension: Whole Genome Shotgun

---



- Zerbrechen von **kompletten Genomen** in Stücke 1KB-100KB
  - Alle Stücke (an-) sequenzieren
  - Celera:
    - Homo sap.: Genom: 3 GB, **28.000.000 Reads**
    - Drosophila: Genom: 120 MB, 3.200.000 Reads
- Schnelle Algorithmen notwendig

## 2. Funktionale Annotation

---

- Sequenzen bestimmen Funktionen
  - Gensequenzen => Proteinsequenz
  - Proteinsequenzen => Struktur
  - Struktur => Funktion
- Grundannahme der Bioinformatik
  - Gleiche Sequenzen – gleiche Funktion
  - Sehr ähnliche Sequenzen – sehr ähnliche Funktion
  - Etwas ähnliche Sequenzen – verwandte Funktion?
  - (Stimmt nicht immer)
- Comparative Genomics

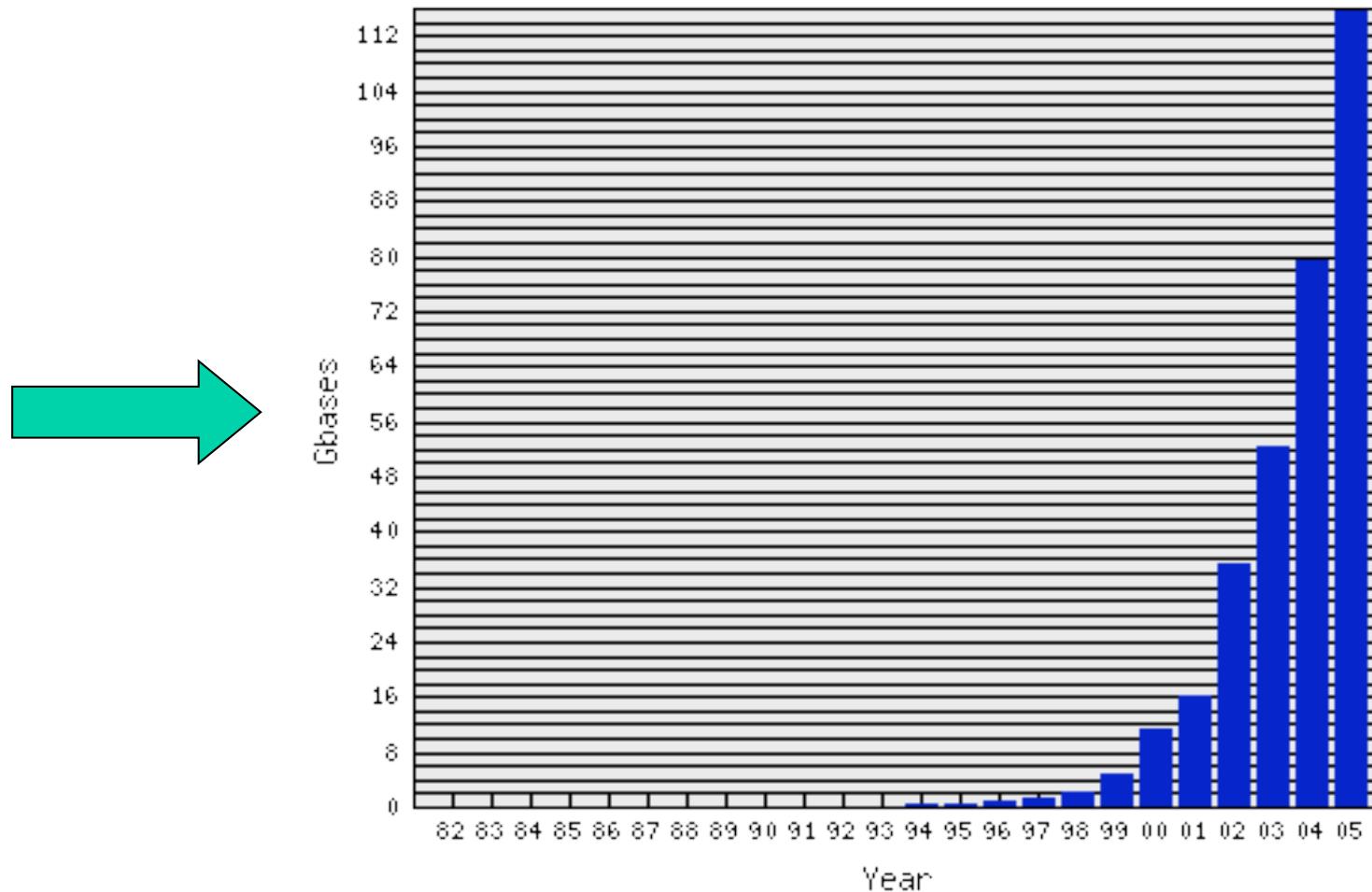
# Standardvorgehen

---

- Gegeben: Eine frisch sequenzierte DNA Sequenz
- Annotationspipeline
  - Suche nach ähnlichen Gensequenzen
  - Suche nach ähnlichen Promotersequenzen
  - Suche nach ähnlichen Proteinen (Übersetzung- Rückübersetzung)
  - Vorhersage neuer Genen durch Programme  
(trainiert auf bekannten Gensequenzen)
  - Suche nach ähnlichen Proteindomänen durch Programme  
(trainiert auf bekannten Proteindomänen)
  - ...
- Alternative: Experimentelle Überprüfung
  - Teuer, auch nicht fehlerfrei
  - Ethische / technische Machbarkeit

# Problemdimension

---



Quelle: EMBL, Genome Monitoring Tables, Stand 9.11.2005

- 
- Strings und Matching - Problemklassen

# Problemklassen

---

- **Exaktes Matching**
  - Gegeben: Strings P, T
  - Gesucht: Alle Auftreten von P in T
  - Variante: Gegeben  $P_1, \dots, P_n, T$ : Vorkommen aller  $P_i$  in T?
- **Approximatives Matchen**
  - Gegeben: Strings S, T
  - Gesucht: Wie ähnlich sind sich S und T?
- **Suche in Datenbanken**
  - Gegeben: Datenbank D von Sequenzen, String P
  - Gesucht: Die k zu P ähnlichen Sequenzen in D

# Exaktes Matching

- Gegeben: P (Pattern) und T (Text)
    - Trivialerweise verlangen wir  $|P| \leq |T|$
    - In der Regel nehmen wir an  $|P| \ll |T|$
  - Gesucht: Sämtliche Vorkommen von P in T
  - Beispiel
    - Auffinden der Erkennungssequenzen von Restriktionsenzymen

# Eco RV - GATATC

# Zeichenketten

---

- Definition

*Ein **String S** ist eine von links nach rechts angeordnete Liste von Zeichen eines Alphabets  $\Sigma$*

- $|S|$  ist die Länge des Strings
- Positionen in  $S$  sind  $1, \dots, |S|$
- $S[i]$  beschreibt das Zeichen an der Position  $i$  im String  $S$
- $S[i..j]$  ist der Substring, welcher an Position  $i$  beginnt und an Position  $j$  endet
- $S[i..j]$  ist ein leerer String, falls  $i > j$
- $S[1..i]$  heißt **Präfix** von  $S$  bis zur Position  $i$
- $S[i..]$  ist das **Suffix** von  $S$ , welches an Position  $i$  beginnt
- **Echte Präfixe und echte Suffixe** umfassen nicht den gesamten String  $S$  und sind nicht leer

# Naiver Ansatz

---

1. P und T an Position 1 ausrichten
2. Vergleiche P mit T von links nach rechts
  - Zwei ungleiche Zeichen  $\Rightarrow$  Gehe zu 3
  - Zwei gleiche Zeichen
    - P noch nicht durchlaufen  $\Rightarrow$  Verschiebe Pointer nach rechts, gehe zu 2
    - P vollständig durchlaufen  $\Rightarrow$  Merke Vorkommen von P in T
3. Verschiebe P um ein Zeichen nach rechts
4. Solange Startposition  $<= |T|-|P|$ , gehe zu 2

|          |               |
|----------|---------------|
| <b>T</b> | ctgagatcgcgta |
| <b>P</b> | <b>gagatc</b> |
|          | <b>gatatc</b> |
|          | <b>gatata</b> |
|          | <b>gatata</b> |

# Naiver Ansatz (cont.)

---

```
for i = 1 to |T| - |P| + 1
    match := true;
    j := 1;
    while ((match) and (j <= |P|))
        if (T(i + j - 1) <> P(j)) then
            match := false;
        else
            j := j + 1;
    end while;
    if (match) then
        -> OUTPUT i
end for;
```

Worst-case

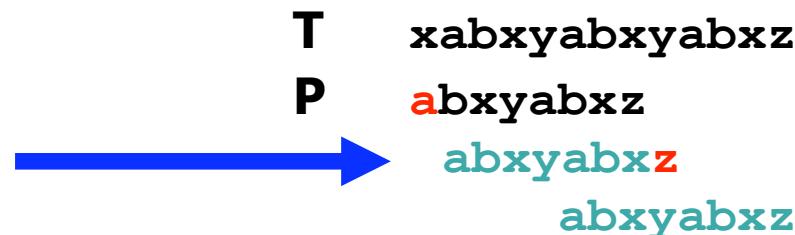
T   aaaaaaaaaaaaaa  
P   aaaaat  
     aaaaat  
     aaaaat  
     aaaaat  
     ...  
     ...

Vergleiche :  $n * (m-n+1) \Rightarrow O(m*n)$

# Optimierungsseite 1

---

- Anzahl der Vergleiche reduzieren
  - P um mehr als ein Zeichen verschieben
  - Aber nie soweit, dass ein Vorkommen von P in T nicht erkannt wird
- Beobachtung: Zeichen



- Substring in  $T$  muss mit a beginnen
- Nächstes a in  $T$  erst an Position 6 – springe 4 Positionen
- Vorkommen von Buchstaben in  $T$  kann während des Vergleichs ab Position 2 gelernt werden