Algorithmische Bioinformatik

Biologische Daten als Strings

Ulf Leser

Wissensmanagement in der Bioinformatik

Ziele für heute

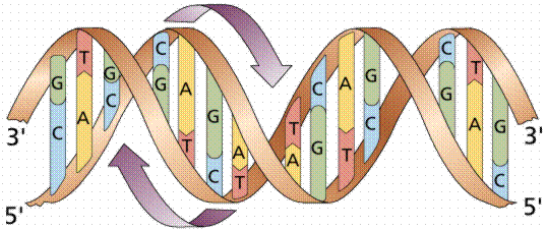
- Wert von Reduktionismus: Genome als Strings
- Anwendungen von Stringmatching in der Bioinformatik
- Varianten des Stringmatching

Inhalt dieser Vorlesung

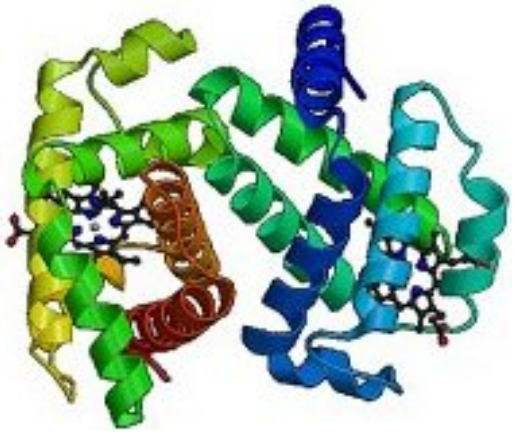
- Warum Stringmatching?
  - Von DNA zu Strings
  - Genomsequenzierung
  - Funktionale Annotation von Sequenzen
- Strings und Matching

Sequenz - Funktion

- DNA
  - Genotyp
  - Vererbung
  - Regulation
  - Produktion von Proteinen

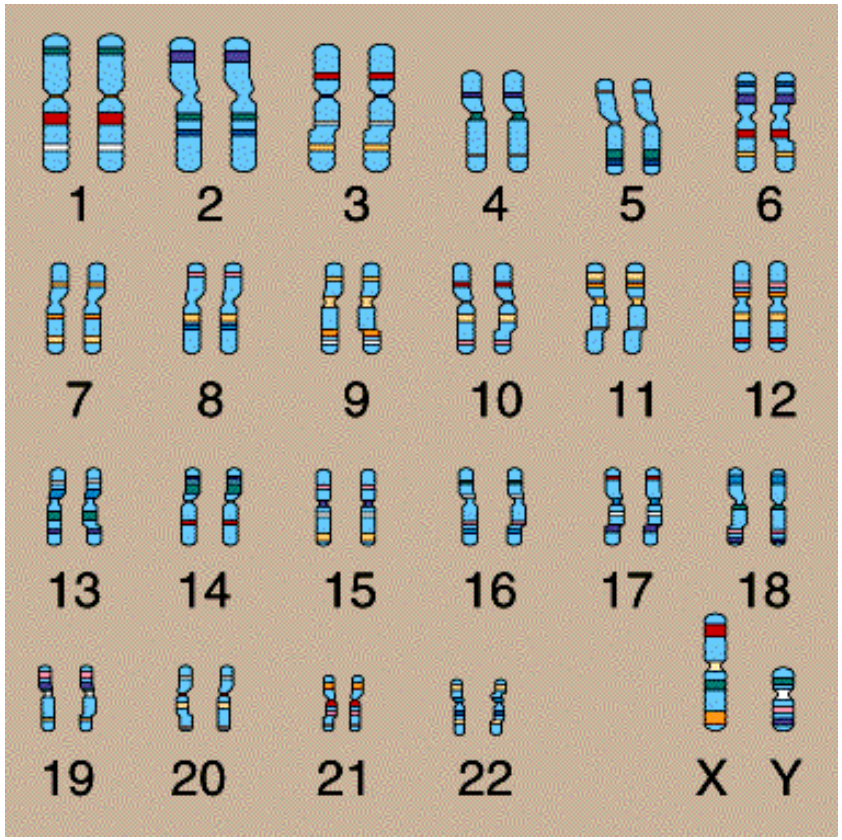


- Proteine
  - Phänotyp
  - Struktur
  - Bindungsverhalten
  - Vielfältigste Funktionen



Das menschliche Genom

- 23 Chromosomenpaare
  - ~ 3.000.000.000 Basen
- Chromosomen kann man (noch) nicht direkt und als Ganzes sequenzieren
- Erste Aufgabe: Kurze, stabile und kopierbare Sequenzabschnitte produzieren

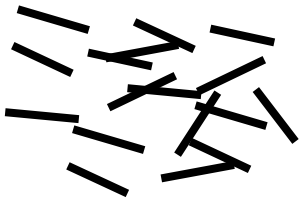


Inhalt dieser Vorlesung

- Warum Stringmatching?
  - Von DNA zu Strings
  - Genomsequenzierung
  - Funktionale Annotation von Sequenzen
- Strings und Matching

Frederick Sanger, 1918 - 2013

Sequenzierung

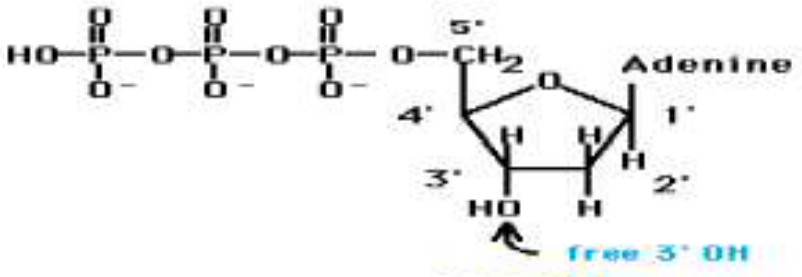


- Gegeben: Bruchstücke unbekannter Sequenz
- Gesucht: Deren Sequenz
- Unmöglich: Ansehen, Messen, Mikroskop, etc.
- (Radioactive) Dideoxy Sequencing
  - Auch „Sanger sequencing“
  - Verfahren von Sanger et al., 1972

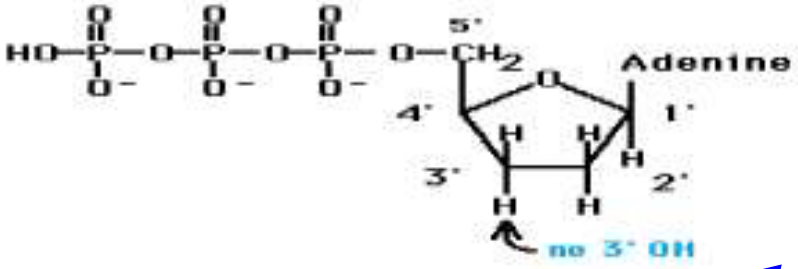
Sanger Sequencing

- Voraussetzungen
  - Sequenz hat einen definierten Anfang
    - Teil des Clonierungsvektors
    - Dient als Bindungsstelle für Primer
  - Polymerase
    - Bindet an doppelsträngigen 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

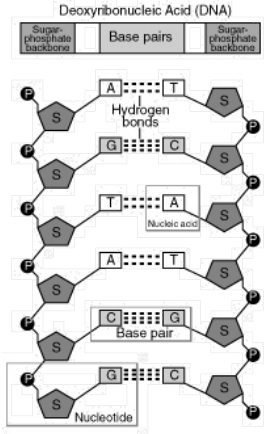
dNTP versus ddNTP

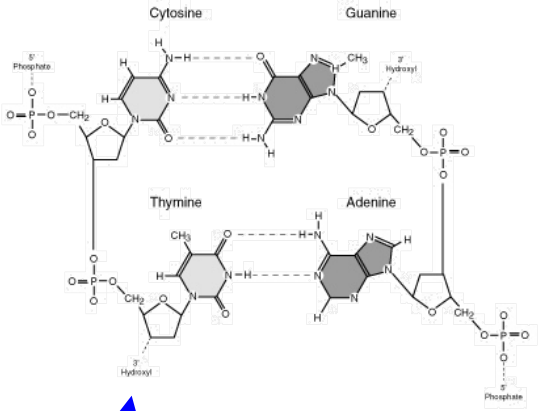


dNTP



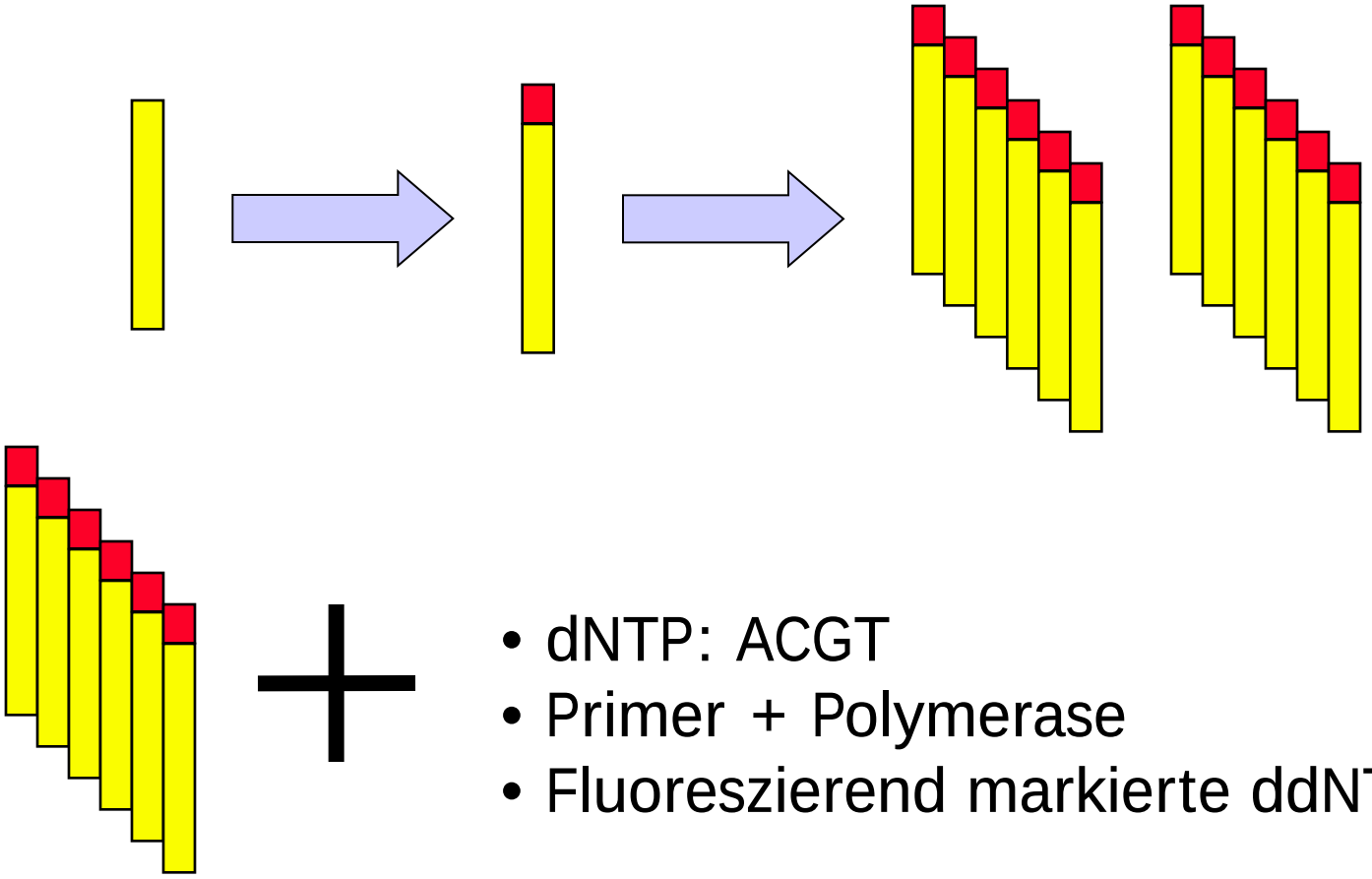
ddNTP





- Dideoxy-Base: keine freie OH Gruppe
  - Werden mit lösungs-abhängiger Wahrscheinlichkeit eingebaut
  - Danach können keine weiteren Basen mehr angehängt werden

Schritt 1 und 2

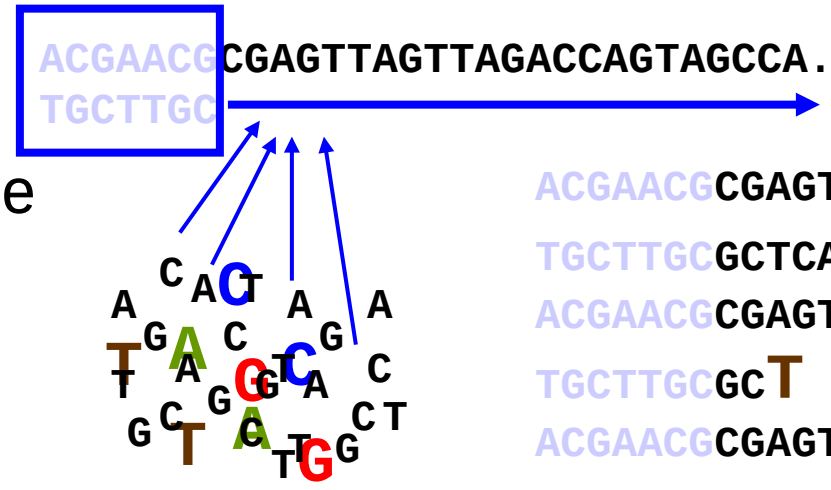


- dNTP: ACGT
- Primer + Polymerase
- Fluoreszierend markierte ddNTP: ACGT

Schritt 3

Primer Template

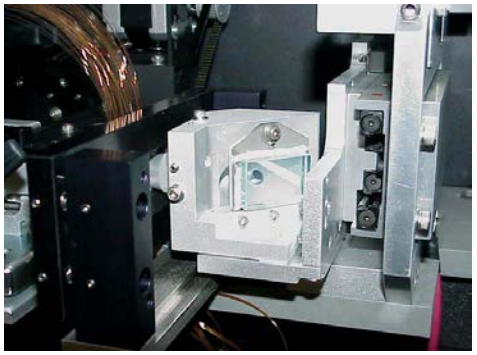
ACGAACGCGAGTTAGTTAGACCAGTAGCCA...



ACGAACGCGAGTTAGTTAGACCAGTAGCCA...
TGCTTGCGCTCAAT
ACGAACGCGAGTTAGTTAGACCAGTAGCCA...
TGCTTGCGCT
ACGAACGCGAGTTAGTTAGACCAGTAGCCA...
TGCTTGCTCTCAATC
ACGAACGCGAGTTAGTTAGACCAGTAGCCA...
TGCTTGCTCTCAATCAATCTG
ACGAACGCGAGTTAGTTAGACCAGTAGCCA...
TGCTTGCT

Schritt 4

Laser &
Detektoren



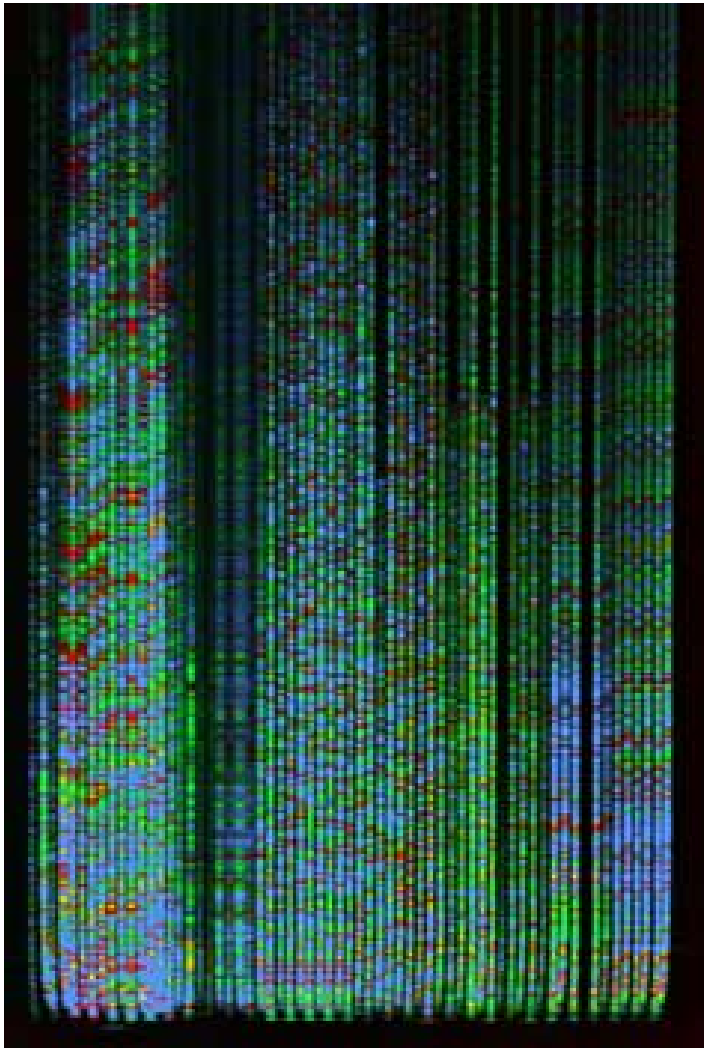
ACGAACGCGAGTTA
ACGAACGCGAG
ACGAACGCGAGTTAGTT
ACGAACGCGAGTTAGTTAGT
ACGAACGCGA

Gel / Kapillar
Elektrophorese

ACGAACGC
ACGAACGCG
ACGAACGCGA
ACGAACGCGAG
ACGAACGCGAGT
ACGAACGCGAGTTA
ACGAACGCGAGGTTAG

Primäres Ergebnis

- „Aktuelle“ Geräte
  - Kapillarelektrophorese
  - Bis zu 96 parallele Kapillare
- Früher (original)
  - Radioaktive Markierung
  - 4 Mischungen (A,G,T,P)
  - 4 Gele (Linien)

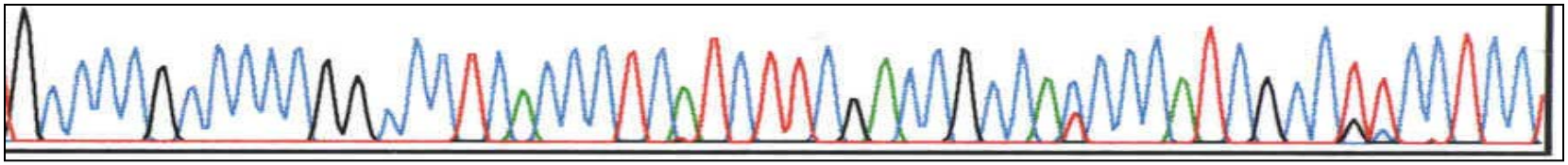


Vom Tracefile zur Sequenz

- Peak-Detection: Von Farbitensitäten zu Peaks
- Base Calling: Berechnung der wahrscheinlichsten Sequenz
- Assembly: Berechnung der wahrscheinlichsten ursprünglichen Gesamtsequenz
- Finishing: Füllen von Lücken durch gezieltes Nachclonieren / Resequenzieren

Traces

- Signalverarbeitung (Rauschen, ...)



- Übersetzung in Traces
  - 4 Arrays, jedes für eine Farbe
  - Intensitätswerte entlang der Zeitachse
- Base calling
  - Peaks entdecken (Abstände werden sukzessive kleiner)
  - Base zuordnen und Qualität bewerten
  - Wahrscheinlichkeit an jeder Position

Assembly

- Basis: Finden von Überlappungen von Sequenzen
- Redundanz ist
  - Notwendig: Verbindung von Teilstücken nur durch Überlappungen
  - Konfliktträchtig: Widersprüche, Mehrdeutigkeit
- Berücksichtigung von Sequenzierfehlern
- Approximation der wahrscheinlichsten Anordnungen

Fehler ?

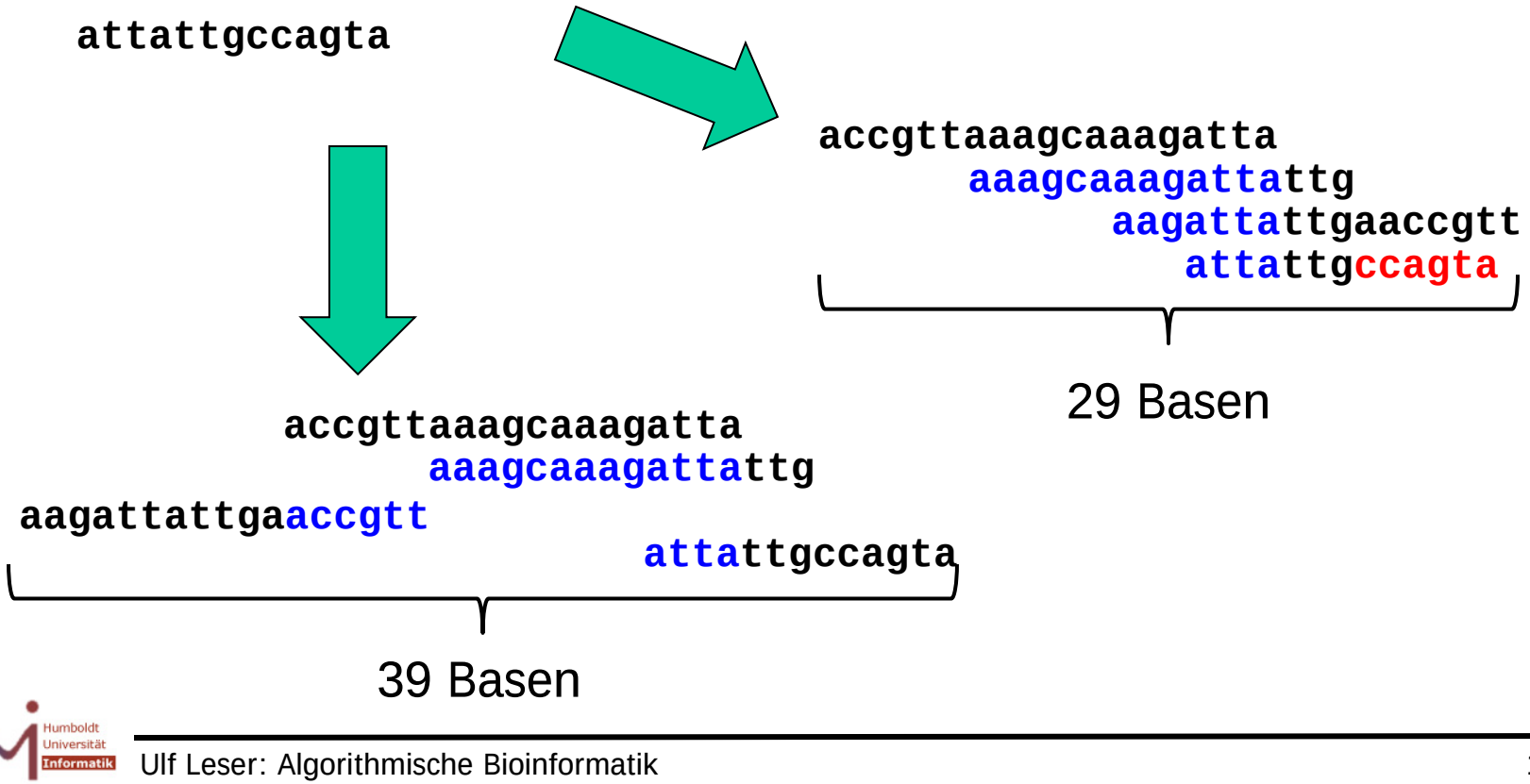
Greedy Algorithmus?

accgttaaagcaaagatta

aagattattgaaccgtt

aaagcaaagattattg

attattgccagta

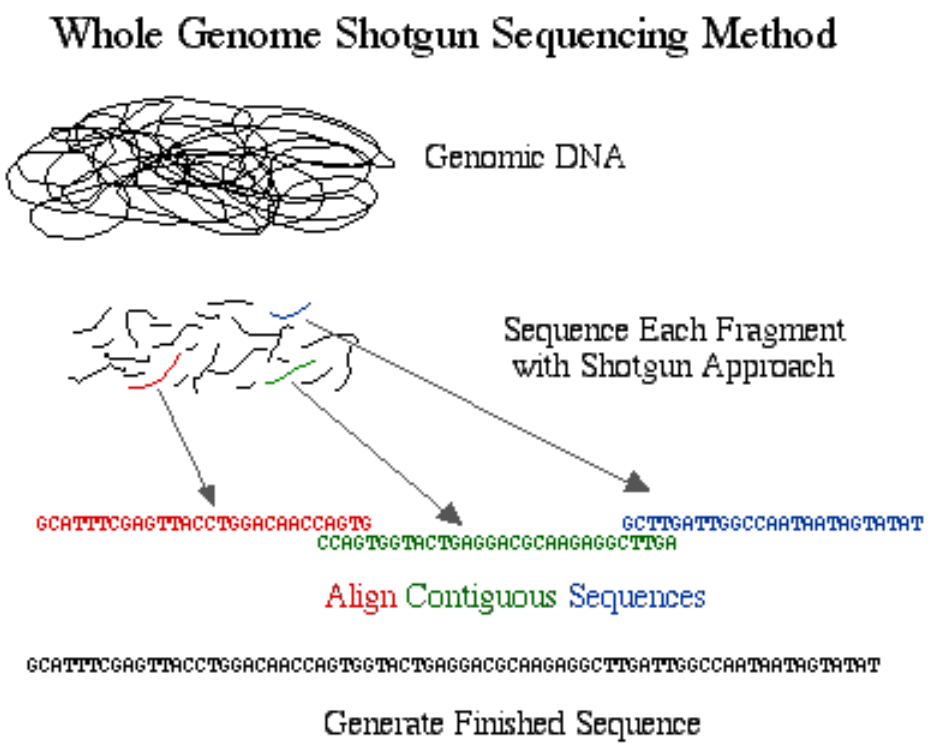


Abstrakte Formulierung

- 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)
- NP-vollständig (in der Zahl der Sequenzen)

- Assembly
  - Verschärfungen von SUPERSTRING wegen Fehler in Sequenzen (s „ungefähr Substring“ von T)
  - Dazu kommt das teure Berechnen der paarweisen Überlappungen

Problemdimension: Whole Genome Shotgun



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

Next Generation Sequencing

- New generation of sequencers since ~2005
  - Illumina, Solexa, 454, Solid, ...
- Much higher throughput
  - ~15 TB raw data in 3-5 days
  - ~600 GB processed data/week
  - Cost for sequencing a genome down to ~2.000 USD
- 3rd generation sequencers
  - Single molecule sequencing
  - A (human) genome in a day
  - Sequence every human
  - Sequence different cells in every human

Illumina HiSeq 2000. DNAVision

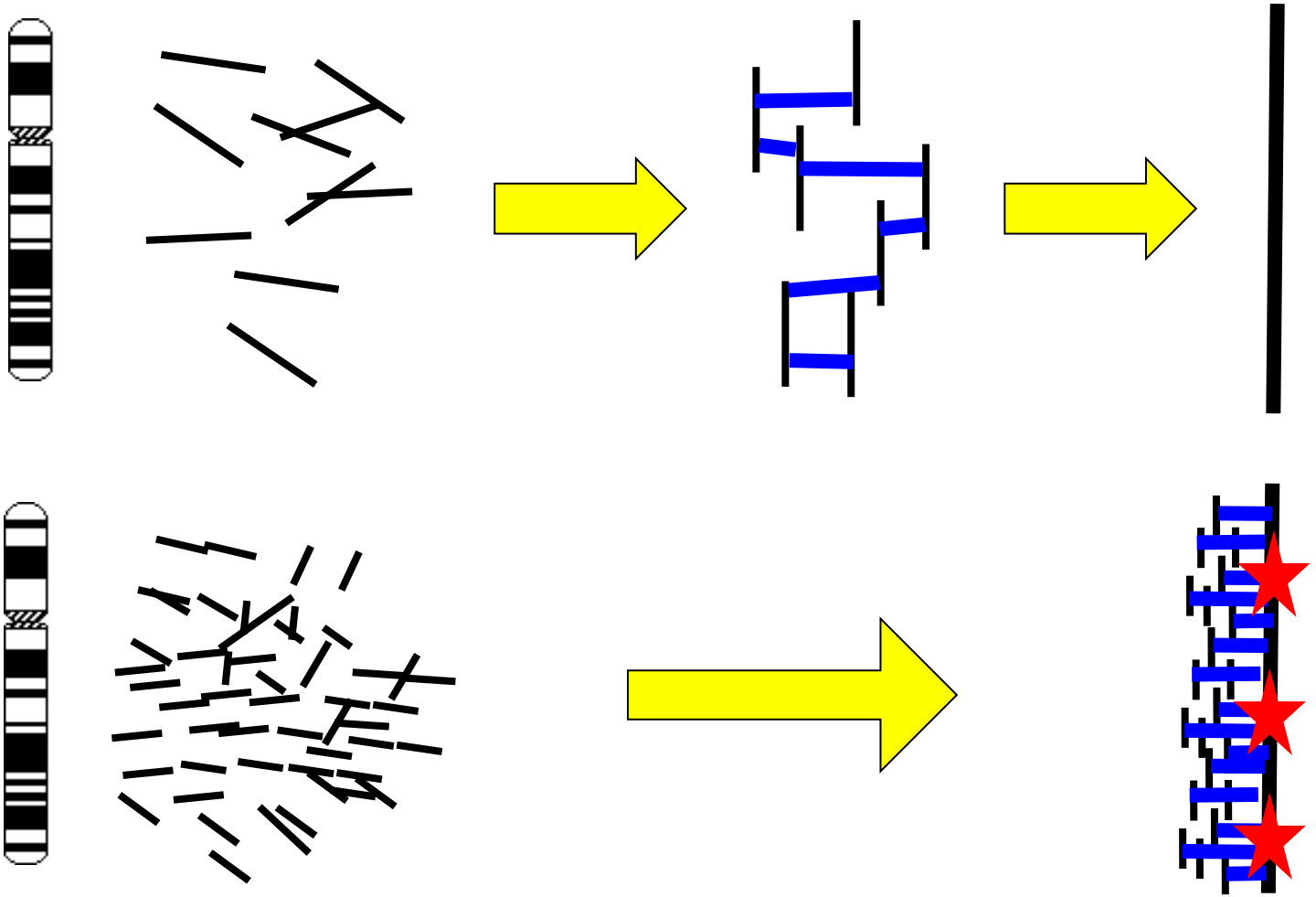
Latest

- 600GB / day, 18.000 genomes per year
- $1,000 genome at 30x coverage
  - Amortized over 18,000 genomes per year over four-year period
- (Not cheap)

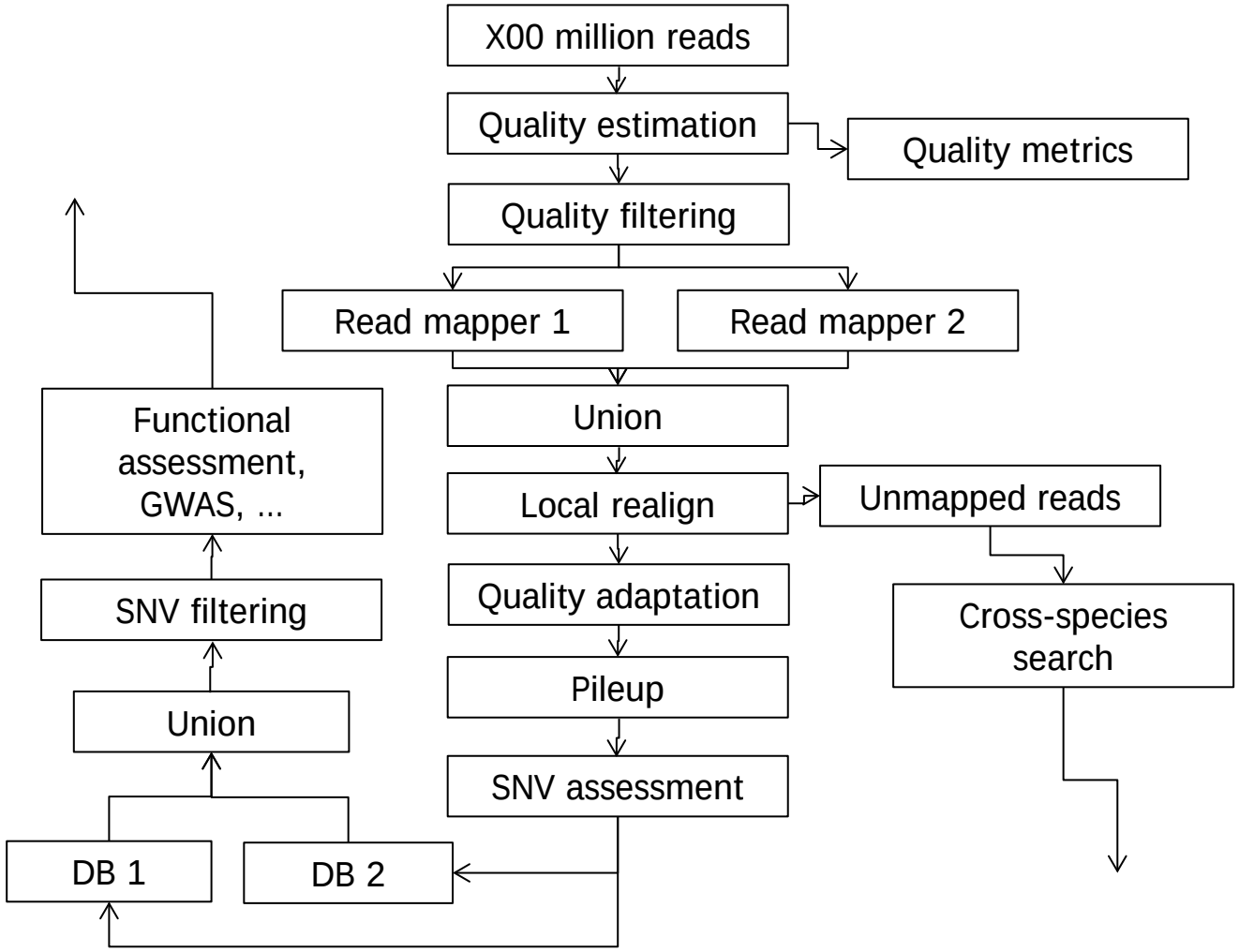
Read Mapping

- Sequenzieren mit NGS erzeugt kurze Reads
- De-Novo Assembly kaum möglich (repeats)
- Alternative: Read-Mapping gegen Referenzgenom
- Problem: Mismatches können verschiedene Quellen haben
  - Fehler in den Reads
  - Fehler im Referenzgenom
  - Natürliche Variationen
  - (Krankheitsassoziierte) Mutationen
  - ...
- Abdeckung und Entropy der Position durch Reads, Häufigkeit der Base in einer Population, Qualität des Reads, Quality-Score der Base im Read, ...

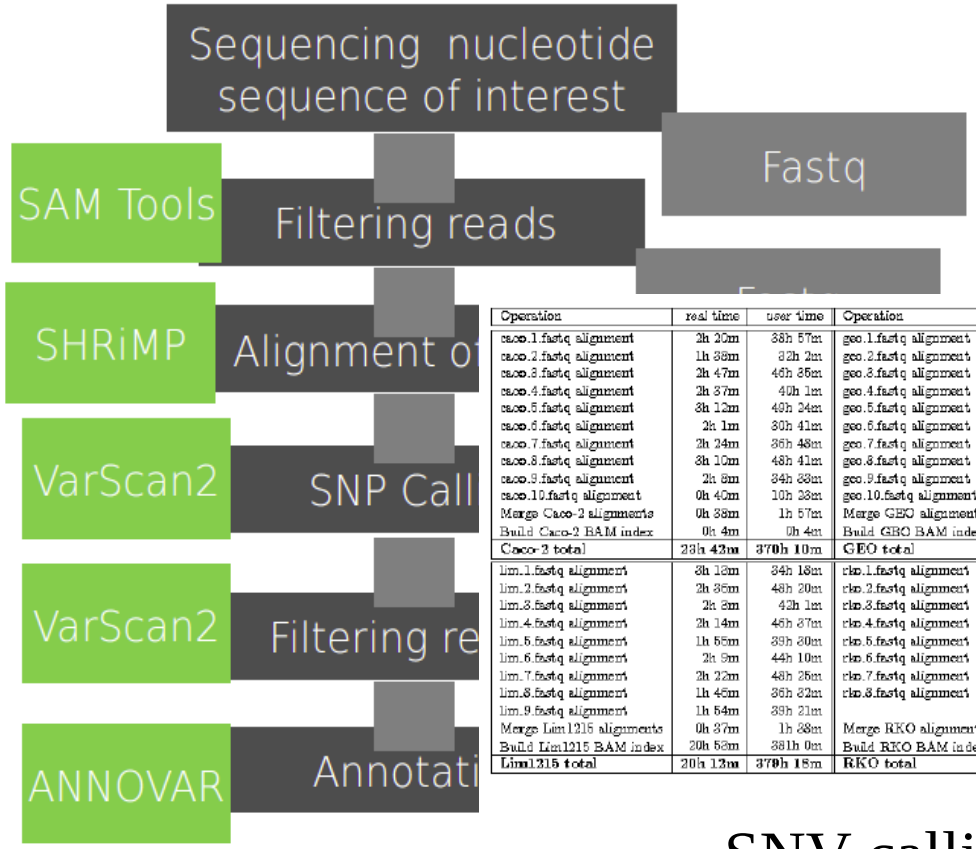
New Task: Read Mapping & SNV Detection



SNV Detection



Example (Single Threaded)



Operation	real time	user time	Operation	real time	user time
caco.1.fastq alignment	2h 20m	38h 57m	geo.1.fastq alignment	2h 14m	37h 46m
caco.2.fastq alignment	1h 38m	32h 2m	geo.2.fastq alignment	2h 3m	37h 32m
caco.3.fastq alignment	2h 47m	46h 35m	geo.3.fastq alignment	1h 58m	36h 10m
caco.4.fastq alignment	2h 37m	40h 1m	geo.4.fastq alignment	1h 40m	32h
caco.5.fastq alignment	3h 12m	49h 24m	geo.5.fastq alignment	1h 36m	34h 24m
caco.6.fastq alignment	2h 1m	30h 41m	geo.6.fastq alignment	3h 11m	69h 51m
caco.7.fastq alignment	2h 24m	36h 48m	geo.7.fastq alignment	3h 5m	68h 12m
caco.8.fastq alignment	3h 10m	48h 41m	geo.8.fastq alignment	2h 38m	55h 20m
caco.9.fastq alignment	2h 8m	34h 33m	geo.9.fastq alignment	3h 12m	73h 9m
caco.10.fastq alignment	0h 40m	10h 38m	geo.10.fastq alignment	2h 12m	37h 3m
Merge Caco-2 alignments	0h 38m	1h 57m	Merge GEO alignments	0h 39m	1h 41m
Build Caco-2 BAM index	0h 4m	0h 4m	Build GEO BAM index	0h 3m	0h 3m
Caco-2 total	**23h 42m**	**370h 10m**	GEO total	**24h 24m**	**483h 21m**
lim.1.fastq alignment	3h 12m	34h 18m	rko.1.fastq alignment	3h 9m	33h 53m
lim.2.fastq alignment	2h 36m	48h 20m	rko.2.fastq alignment	3h 9m	62h 18m
lim.3.fastq alignment	2h 3m	42h 1m	rko.3.fastq alignment	2h 11m	41h 14m
lim.4.fastq alignment	2h 14m	46h 37m	rko.4.fastq alignment	1h 58m	36h 32m
lim.5.fastq alignment	1h 55m	39h 30m	rko.5.fastq alignment	3h 20m	64h 32m
lim.6.fastq alignment	2h 9m	44h 10m	rko.6.fastq alignment	2h 11m	45h 44m
lim.7.fastq alignment	2h 22m	48h 26m	rko.7.fastq alignment	4h 16m	91h 26m
lim.8.fastq alignment	1h 45m	36h 32m	rko.8.fastq alignment	3h 12m	68h 56m
lim.9.fastq alignment	1h 54m	39h 21m			
Merge Lim1215 alignments	0h 37m	1h 38m	Merge RKO alignments	0h 33m	1h 45m
Build Lim1215 BAM index	20h 53m	381h 0m	Build RKO BAM index	0h 3m	0h 3m
Lim1215 total	**20h 13m**	**370h 18m**	RKO total	**23h 59m**	**446h 37m**

Alignment: 4 days

SNV calling: 12h

Operation	Caco-2	GEO	Lim1215	RKO
Generate pileup	2h 23m	2h 4m	1h 52m	1h 55m
Call SNVs from VarScan	0h 22m	0h 21m	0h 17m	0h 17m
Call indels from VarScan	0h 22m	0h 23m	0h 21m	0h 20m
Call SNVs from SNVMix	0h 2m	0h 2m	0h 2m	0h 2m
Import SNVs into DB	< 1m	< 1m	< 1m	< 1m
Filter SNVs	0h 14m	0h 17m	0h 14m	0h 17m
Total	**3h 24m**	**3h 9m**	**2h 48m**	**2h 54m**

Annotation / filtering: 2h

Operation	Caco-2	GEO	Lim1215	RKO
Annotate genes	< 1m	< 1m	< 1m	< 1m
Annotate from dbSNP	0h 25m	0h 25m	0h 25m	0h 25m
Annotate SIFT predictions	0h 1m	0h 2m	0h 2m	0h 2m
Annotate PolyPhen-2 predictions	0h 1m	0h 2m	0h 2m	0h 2m
Total	**0h 27m**	**0h 29m**	**0h 29m**	**0h 29m**

Inhalt dieser Vorlesung

- Warum Stringmatching?
  - Von DNA zu Strings
  - Genomsequenzierung
  - Funktionale Annotation von Sequenzen
- Strings und Matching

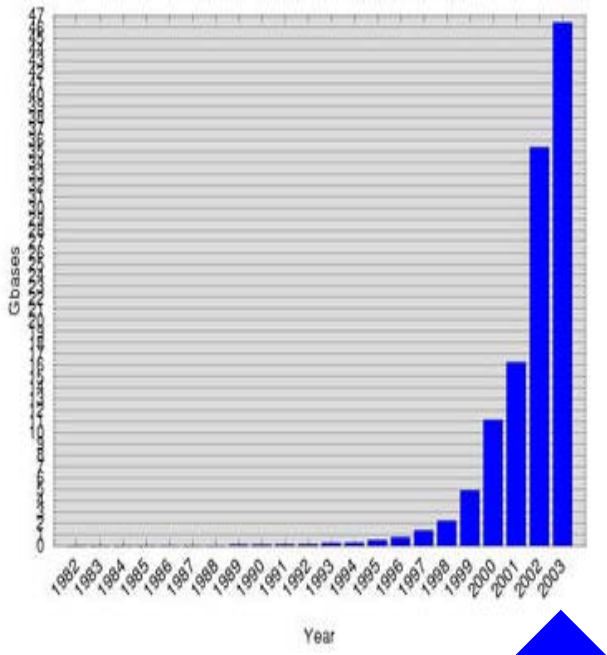
Funktionale Annotation

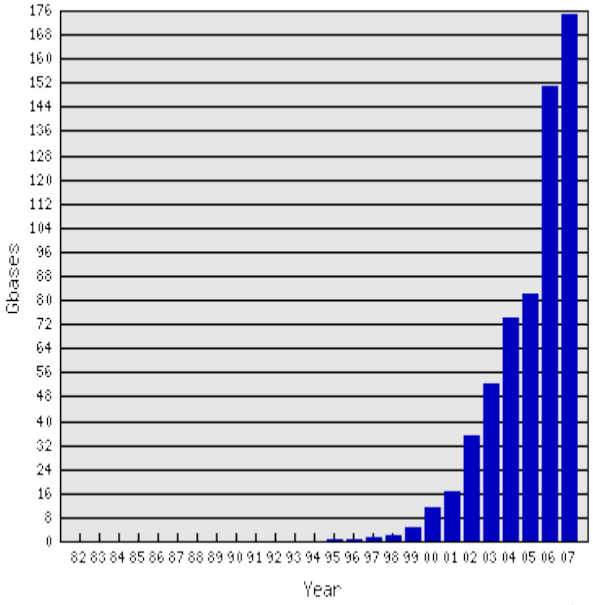
- DNA Sequenzen bestimmen Proteinfunktionen
  - Gensequenzen => Proteinsequenz
  - Proteinsequenzen => Struktur
  - Struktur => Funktion
- Beobachtung (Grundpfeiler der Bioinformatik)
  - Gleiche Sequenzen – gleiche Funktion
  - Sehr ähnliche Sequenzen – sehr ähnliche Funktion
  - Etwas ähnliche Sequenzen – verwandte Funktion?
- Insbesondere: Sequenzen aus verschiedenen Spezies

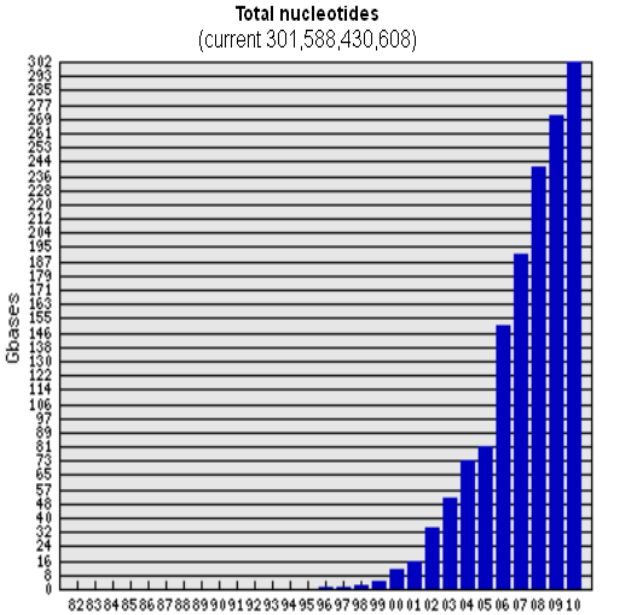
Standardvorgehen

- 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







Large Scale Human Sequencing

50.000 samples: To obtain a comprehensive description of genomic, transcriptomic and epigenomic changes in 50 different tumor types and/or subtypes which are of clinical and societal importance across the globe.

Genomics England … is creating a lasting legacy for patients, the NHS and the UK economy through the sequencing of 100,000 genomes: the 100,000 Genomes Project.

The Veterans Affairs (VA) Office of Research and Development is launching the Million Veteran Program (MVP) …. The goal of MVP is to better understand how genes affect health and illness in order to improve health care.

Inhalt dieser Vorlesung

- Warum Stringmatching?
  - Von DNA zu Strings
  - Genomsequenzierung
  - Funktionale Annotation von Sequenzen
- Strings und Matching

Zeichenketten

- Definition
  Ein String S ist eine von links nach rechts angeordnete Liste von Zeichen eines Alphabets Σ
  - *|S| ist die Länge des Strings*
  - *Positionen in S sind 1,…,|S|*
    - *Wir zählen ab 1*
  - *S[i] ist das Zeichen an der Position i im String S*
  - *S[i..j] ist der Substring, der an Pos. i beginnt und an Pos. 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*

Problemklassen

- Exaktes Substring-Matching (Patternmatching, Matching)
  - Gegeben: Strings P, T
  - Gesucht: Alle Auftreten von P in T
  - Variante: Gegeben $P_1,...,P_n$, T: Vorkommen aller P_i in T?
- Approximatives Matchen
  - Gegeben: Strings S, T
  - Gesucht: Wie ähnlich sind sich S und T?
  - Variante: Ist S in T mit höchstens k Fehlern enthalten?
- Approximatives lokales Matching
  - Gegeben: Strings S, T
  - Gesucht (Read-Mapping): Substring in T, der ähnlich zu S ist?
  - Variante (BLAST): Substring in T, der ähnlich zu Substring in S ist?

Datenbank-Varianten

- Gegeben: Datenbank D von Sequenzen, String P
- Exaktes Substring-Matching: Alle $d \in D$, die P enthalten
- Approximatives Matchen: Alle $d \in D$, die zu P ähnlich sind
- Approximatives lokales Matching: Alle $d \in D$, die einen zu P ähnlichen Substring enthalten
- Top-K: Die k zu P ähnlichsten Sequenzen $d \in D$
- Lokales Top-K: Die k Sequenzen $d \in D$, die die k zu P ähnlichsten Substrings enthalten

Exaktes Matching

- Gegeben: P (Pattern) und T (Text)
  - Trivialerweise verlangen wir |P| ≤ |T|
- Gesucht: Sämtliche Vorkommen von P in T
- Beispiel: Erkennungssequenzen von Restriktionsenzymen

Eco RV - GATATC

tcagcttactaattaaaaattctttctagtaagtgctaagatcaagaaaataaattaaaaataatggaacatggcacattttcctaaactcttcacagattgctaatga
ttattaattaaagaataaatgttataatttttatggtaacggaatttcctaaaatattaattcaagcaccatggaatgcaaataagaaggactctgttaattggtact
attcaactcaatgcaagtggaactaagttggtattaatactctttttacatatatatgtagttattttaggaagcgaaggacaatttcatctgctaataaagggattac
atatttatttttgtgaatataaaaaatagaaagtatgttatcagattaaacttttgagaaaggtaagtatgaagtaaagctgtatactccagcaataagttcaaataggc
gaaaaacttttaataacaaagttaaataatcattttgggaattgaaatgtcaaagataattacttcacgataagtagttgaagatagtttaaatttttctttttgtatt
acttcaatgaaggtaacgcaacaagattagagtatatatggccaataaggtttgctgtaggaaaattattctaaggagatacgcgagagggcttctcaaatttattcaga
gatggatgtttttagatggtggtttaagaaaagcagtattaaatccagcaaaactagaccttaggtttattaaagcgaggcaataagttaattggaattgtaaaagatat
ctaattcttcttcatttgttggaggaaaactagttaacttcttaccccatgcagggccatagggtcgaatacgatctgtcactaagcaaaggaaaatgtgagtgtagact
ttaaaccatttttattaatgactttagagaatcatgcatttgatgttactttcttaacaatgtgaacatatttatgcgattaagatgagttatgaaaaaggcgaatatat
tattcagttacatagagattatagctggtctattcttagttataggacttttgacaagatagcttagaaaataagattatagagcttaataaaagagaacttcttggaat
tagctgcctttggtgcagctgtaatggctattggtatggctccagcttactggttaggttttaatagaaaaattccccatgattgctaattatatctatcctattgagaa
caacgtgcgaagatgagtggcaaattggttcattattaactgctggtgctatagtagttatccttagaaagatatataaatctgataaagcaaaatcctggggaaaatat
tgctaactggtgctggtagggtttggggattggattatttcctctacaagaaatttggtgtttactgatatccttataaataatagagaaaaaattaataaagatgatat

Notation

- Annahmen
  - $|T| = m \neq 0$
  - $|P| = n \neq 0$
  - $m >> n$
  - Alphabet Σ endlich
  - P, T sind Strings über Σ
  - Kosten für Vergleich zweier Zeichen aus Σ : 1
- Komplexitätsanalyse: Anzahl an Zeichenvergleichen
  - Manchmal auch $|\Sigma|$

Übersicht Exaktes Matching

- Naiver Algorithmus: O(n*m)
- Z Algorithmus: O(m+n)
  - Wird gerne in anderen Verfahren zum Pre-Processing verwendet
- Boyer-Moore: Sublinear im Average Case
  - Worst Case O(n*m), aber Average Case sublinear
  - Erweiterung zu linearem Worst-Case möglich (aber irrelevant)
- Knuth-Morris-Pratt: O(m+n)
  - Voraussetzung für Aho-Corasick zur Suche nach mehreren Pattern
- Später: Indexstrukturen, z.B. Suffixbäume
  - O(n+k) (nach Preprocessing: O(m))

Fazit

- Stringalgorithmen für viele Fragestellungen der Bioinformatik essentiell, wie z.B.
  - Assemblierung von Genomen
  - Funktionale Annotation von Genen / Proteinen
- Wegen großer Datenmengen ist hohe Performance wichtig
- Viele Varianten existieren – exakt / approximativ, Einzelvergleich / Datenbanksuche, ähnlichste / top-K, etc.

Selbsttest

- Wie funktioniert Sanger-Sequenzierung? Welche Fehlerquellen gibt es?
- Welche Rolle kann Stringmatching bei der Aufklärung der Funktionen von Genen spielen?
- Komplexität des Assembly-Problems inkl. Begründung?
- Was ist paired-end Sequenzierung?