# Bioinformatik

BLAT: Datenbanksuche für sehr ähnliche Sequenzen

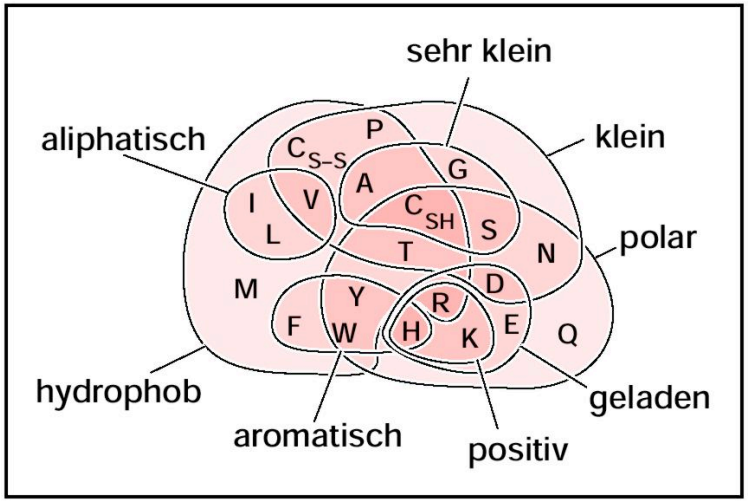
Multiples Sequenzalignment

Silke Trißl / Ulf Leser

Wissensmanagement in der Bioinformatik

# Hintergrund

- Schon öfters angesprochen ...
  - Ähnlichkeitsmatrizen, Substitutionsmatrizen, Scorefunktionen, ...
- Ersetzung einer Base/Aminosäure durch eine andere hat unterschiedliche Bedeutung
  - Basen: Auswirkungen auf kodiertes Protein nicht gleichverteilt über die drei Codon-Positionen
  - Aminosäuren
    - Ersetzung mit „sehr ähnlichen" Aminosäuren ändert Proteinstruktur kaum
    - Ersetzung mit „wenig ähnlichen" Aminosäuren kann Struktur vollkommen ändern



# Substitutionsmatrizen

- Bewertung individueller Substitutionen
- Algorithmen sind davon unberührt, nur Parameter ändern sich
- Das Ergebnis kann sich aber vollkommen ändern
- Beispiele: Blosum62, Identitätsmatrix

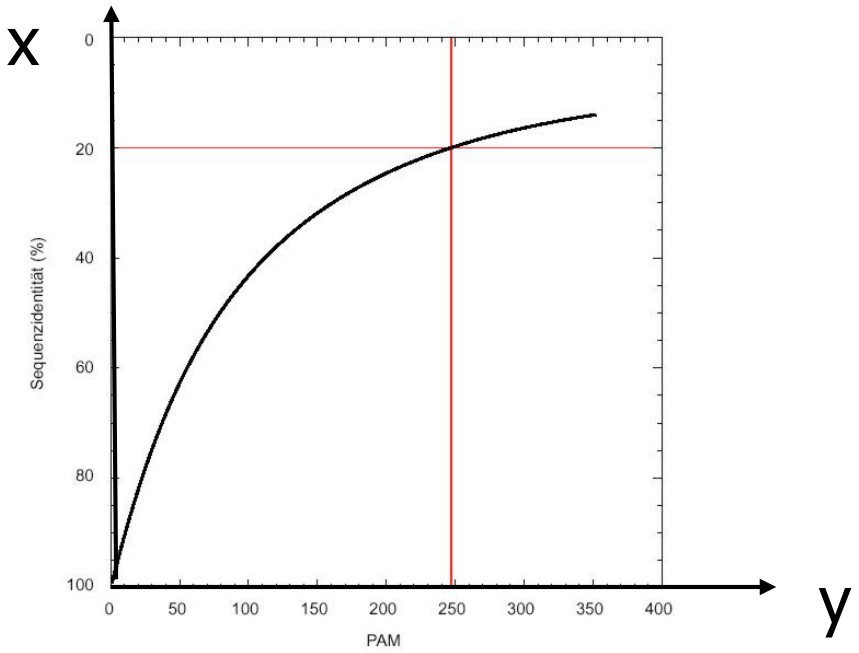
| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 4 | -1 | -2 | -2 | 0 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -3 | -2 | 0 | -2 | -1 |
| R | -1 | 5 | 0 | -2 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 2 | -1 | -3 | -2 | -1 | -1 | -3 | -2 | -3 | -1 | 0 |
| N | -2 | 0 | 6 | 1 | -3 | 0 | 0 | 0 | 1 | -3 | -3 | 0 | -2 | -3 | -2 | 1 | 0 | -4 | -2 | -3 | 3 | 0 |
| D | -2 | -2 | 1 | 6 | -3 | 0 | 2 | -1 | -1 | -3 | -4 | -1 | -3 | -3 | -1 | 0 | -1 | -4 | -3 | -3 | 4 | 1 |
| C | 0 | -3 | -3 | -3 | 9 | -3 | -4 | -3 | -3 | -1 | -1 | -3 | -1 | -2 | -3 | -1 | -1 | -2 | -2 | -1 | -3 | -3 |
| Q | -1 | 1 | 0 | 0 | -3 | 5 | 2 | -2 | 0 | -3 | -2 | 1 | 0 | -3 | -1 | 0 | -1 | -2 | -1 | -2 | 0 | 3 |
| E | -1 | 0 | 0 | 2 | -4 | 2 | 5 | -2 | 0 | -3 | -3 | 1 | -2 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | 4 |
| G | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 6 | -2 | -4 | -4 | -2 | -3 | -3 | -2 | 0 | -2 | -2 | -3 | -3 | -1 | -2 |
| H | -2 | 0 | 1 | -1 | -3 | 0 | 0 | -2 | 8 | -3 | -3 | -1 | -2 | -1 | -2 | -1 | -2 | -2 | 2 | -3 | 0 | 0 |
| I | -1 | -3 | -3 | -3 | -1 | -3 | -3 | -4 | -3 | 4 | 2 | -3 | 1 | 0 | -3 | -2 | -1 | -3 | -1 | 3 | -3 | -3 |
| L | -1 | -2 | -3 | -4 | -1 | -2 | -3 | -4 | -3 | 2 | 4 | -2 | 2 | 0 | -3 | -2 | -1 | -2 | -1 | 1 | -4 | -3 |
| K | -1 | 2 | 0 | -1 | -3 | 1 | 1 | -2 | -1 | -3 | -2 | 5 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 0 | 1 |
| M | -1 | -1 | -2 | -3 | -1 | 0 | -2 | -3 | -2 | 1 | 2 | -1 | 5 | 0 | -2 | -1 | -1 | -1 | -1 | 1 | -3 | -1 |
| F | -2 | -3 | -3 | -3 | -2 | -3 | -3 | -3 | -1 | 0 | 0 | -3 | 0 | 6 | -4 | -2 | -2 | 1 | 3 | -1 | -3 | -3 |
| P | -1 | -2 | -2 | -1 | -3 | -1 | -1 | -2 | -2 | -3 | -3 | -1 | -2 | -4 | 7 | -1 | -1 | -4 | -3 | -2 | -2 | -1 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -2 | 0 | -1 | -2 | -1 | 4 | 1 | -3 | -2 | -2 | 0 | 0 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 5 | -2 | -2 | 0 | -1 | -1 |
| W | -3 | -3 | -4 | -4 | -2 | -2 | -3 | -2 | -2 | -3 | -2 | -3 | -1 | 1 | -4 | -3 | -2 | 11 | 2 | -3 | -4 | -3 |
| Y | -2 | -2 | -2 | -3 | -2 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | -1 | 3 | -3 | -2 | -2 | 2 | 7 | -1 | -3 | -2 |
| V | 0 | -3 | -3 | -3 | -1 | -2 | -2 | -3 | -3 | 3 | 1 | -2 | 1 | -1 | -2 | -2 | 0 | -3 | -1 | 4 | -3 | -2 |
| B | -2 | -1 | 3 | 4 | -3 | 0 | 1 | -1 | 0 | -3 | -4 | 0 | -3 | -3 | -2 | 0 | -1 | -4 | -3 | -3 | 4 | 1 |
| Z | -1 | 0 | 0 | 1 | -3 | 3 | 4 | -2 | 0 | -3 | -3 | 1 | -1 | -3 | -1 | 0 | -1 | -3 | -2 | -2 | 1 | 4 |

| | A | C | G | T |
|---|---|---|---|---|
| A | 5 | -4 | -4 | -4 |
| C | -4 | 5 | -4 | -4 |
| G | -4 | -4 | 5 | -4 |
| T | -4 | -4 | -4 | 5 |

# PAM als Sequenzabstand

- Definition
  *Seien $S_1$ und $S_2$ zwei Proteinsequenzen mit $|S_1|=|S_2|$. $S_1$ und $S_2$ heißen x PAM entfernt, wenn $S_1$ in $S_2$ überführt wurde mit x Punktmutationen pro 100 Aminosäuren*
- Eigenschaften
  - PAM beachtet keine Inserts und Deletions
  - x schätzt man aus den Unterschieden von $S_1$ und $S_2$
    - Kann man analytisch berechnen oder durch Simulation bestimmen
  - 50 PAM Abstand heißt nicht 50 Veränderungen pro 100 Aminosäuren

# Mutationshäufigkeit und Sequenzidentität




- Kein linearer Zhg: Rückmutationen, Doppelmutationen, etc.
- Mit steigendem y wächst der Abstand zur Ausgangsse-quenz ab einem bestimmten Punkt kaum noch
- Ab diesem Punkt kann man über Homologie kaum noch eine Aussage machen

# PAM Matrizen

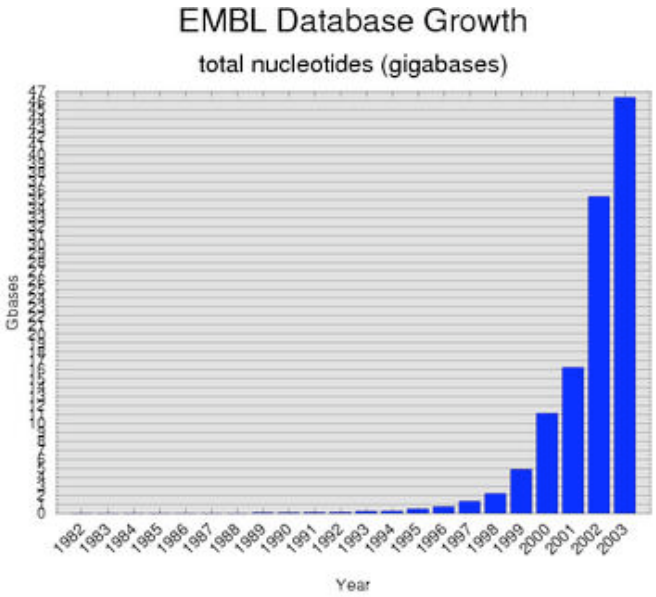
- Definition
  *Seien ($S_{1,1}$, $S_{2,1}$), ..., ($S_{1,n}$, $S_{2,n}$) Paare von Sequenzen, die jeweils x PAM entfernt sind. Dann ist die PAM-x Matrix $M_x$ wie folgt definiert:*
  - *Sei f(i) die relative Häufigkeit der Aminosäure $A_i$*
  - *Seien alle Paare optimal aligniert*
    - *Sei $S_{k,l}$' das $S_{k,l}$ mit den durch das Alignment eingefügten Leerzeichen*
  - *Sei f(i,j) die relative Übergangshäufigkeit von $A_i$ zu $A_j$ in allen alignierten Paaren*
    - *Anzahl von Positionen k mit $S_{1,z}$'[k]=$A_i$ und $S_{2,z}$'[k]=$A_j$ oder umgekehrt*
      - *Übergang ist „richtungslos": f(i,j) = f(j,i)*
      - *Paare ($A_x$, _) werden ignoriert*
    - *f(i,j) wird auf die Gesamtzahl aller Paare ohne INSDEL normiert*
  - *Damit:*

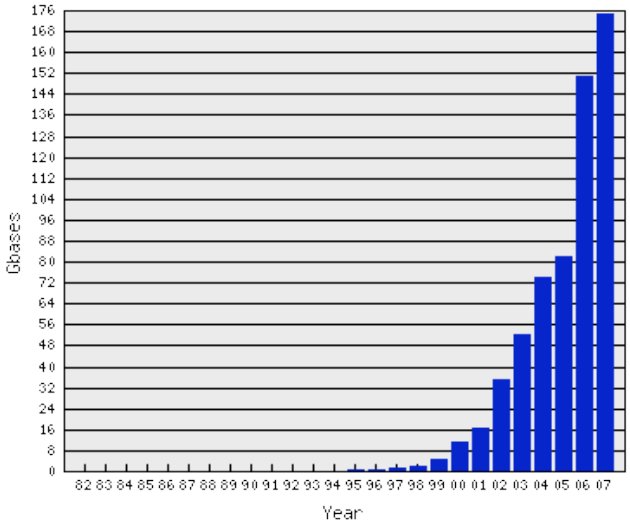
$$M_x(i,j) = \log\left(\frac{f(i,j)}{f(i) * f(j)}\right)$$

# Reale PAM Matrizen

- Vorgehen von Dayhoff et al.
  - Paare eng verwandter Sequenzen auswählen
    - >85% Identität, 34 Proteinfamilien
  - Manuell alignieren
  - PAM-1 Matrix $M_1$ aus Häufigkeiten berechnen
  - PAM-x Matrizen wie folgt berechnen: $M_n=(M_1)^n$
- Dem liegen viele Annahmen zugrunde
  - Evolutionary Clock Theory: Evolution verläuft gleichmäßig
    - in der Zeit und in den Sequenzpositionen
    - Proportionalität von Veränderungen
    - Hochrechnung langer Distanzen aus kurzen
  - Keine Insertions oder Deletions
  - Unabhängigkeit der Mutationswahrscheinlichkeit von der Position in der Sequenz (und der Nachbarschaft)

# Heuristische Alignierung




- Annotation neuer Sequenzen basiert auf Suche nach homologen Sequenzen in Sequenzdatenbanken
- Datenmenge wächst exponentiell
- Alignment ist zu langsam
- Gesucht sind Verfahren
  - Lineare oder sublinearer Laufzeit
  - Möglichst hohe Ergebnisqualität

# Heuristik

- Definition: *Heuristik bezeichnet die Kunst, mit begrenztem Wissen und wenig Zeit zu guten Lösungen zu kommen.*

- Beispiel: Exaktes Stringmatching; Finden von P in T
  - Naiver Algorithmus: verschiebe P um eine Position in T
    - zu langsam -> wir wollen P um mehr Zeichen verschieben
  - Annahme
    - in P gibt es keine Wiederholungen
    - wir können P immer um die Anzahl Zeichen verschieben, die wir verglichen haben
  - Problem: die Annahme stimmt nicht immer
    - wir finden vielleicht nicht alle Vorkommen von P in T

```
abyabyabyacyaby          abyabyabyacyaby
abyacy                   abyaby
     abyacy                   abyaby
      abyacy
```

# Sensitivität und Spezifität

| | | Reality | |
|---|---|---|---|
| | | + | - |
| Prediction | + | TruePositive (TP) | FalsePositive (FP) |
| | - | FalseNegative (FN) | TrueNegative (TN) |

- Spezifität = TP/(TP+FP) (Precision)
  - Wie viele der Treffen des Verfahrens sind wirklich welche?
- Sensitivität = TP/(TP+FN) (Recall)
  - Wie viele der echten Treffer findet das Verfahren?
- Eine Balance
  - Algorithmen berechnen einen Score pro Sequenz
  - Hohe Score – Positiv; Niedriger Score – Negativ
  - Wenn Score mit Wahrscheinlichkeit für korrekte Klassifikation korreliert, folgt daraus
    - Ergebnismenge klein: SP=hoch, SE=klein
    - Ergebnismenge groß: SP=niedrig, SE=hoch

# BLAST

- Altschul, Gish, Miller, Myers, Lipman: „Basic Local Alignment Search Tool", J Mol Bio, 1990.
- Heuristische Suche
  - Sehr schnell, findet aber nicht alle optimalen Alignments
  - Berechnet statistische Signifikanz der Treffer
- *Die* Erfolgsgeschichte der Bioinformatik
  - In manchen Darstellungen äquivalent zu „Bioinformatik"
  - Eingesetzt auf NCBI/EBI Server – von der ganzen Welt benutzt
  - Software frei erhältlich
  - „we blasted the sequence …"
- Diverse Weiterentwicklungen
  - Gapped-BLAST und PSI-BLAST (1997), MegaBlast, TextBlast, …

# BLAST Parameter

- BLAST sucht nach guten lokalen Alignments in Sequenzen
  - DNA, mRNA, Protein

- Gegeben
  - Suchsequenz P, Datenbank DB={$S_1$,...,$S_n$}
  - Substitutionsmatrix M
  - Parameter w: Länge der „Seeds“
  - Parameter t: Initialer Schwellwert
  - Parameter c: Gesamtschwellwert
    - Wird berechnet in Abhängigkeit von t, M, |DB|, |P|
  - Parameter v: Erwünschte Anzahl Treffer
    - Blast berechnet die v ähnlichsten Subsequenzen

# BLAST Schritt 1 und 2

- Schritt 1
  - Bestimme alle Teilwörter $P_1,...,P_m$ der Länge w in P
    - Mit Überlappung – keine Partitionierung
    - Wie viele gibt es?
- Schritt 2
  - Suche nach Hits von $P_1,...,P_m$ in DB mit Score über t
    - Hits müssen nicht exakt sein
    - Vergleiche alle $P_i$ mit allen Teilwörtern in DB der Länge w in T
      - Keine INSDELs, Verwendung von M
  - Durch den „unscharfen" Hit mit Schwellwert t
    - werden auch Hits gefunden, die keine perfekten Matches sind
    - werden wenig aussichtsreiche Positionen in DB ausgeschlossen
      - Annahme: Ein gutes lokales Alignment A in Sequenz S muss mindestens einen Hit enthalten
      - Es ist nicht notwendig, dass jedes Teilwort in A mit seinem Teilwortpartner in P einen Hit hat.

# BLAST Schritt 3

- Schritt 3
  - Für jeden Hit H zwischen DB-Sequenz S und $P_i$
  - Verlängere Bereich um H sowohl in P als auch in S
    - Gapfree Alignment nach links und rechts wachsen
    - Solange, bis
      - Sequenz P oder S zu Ende ist, oder
      - Alignmentscore fällt unter geschätzten Schwellwert c, oder
      - Alignmentscore fällt „signifikant" unter bisherige v beste Treffer
        - » „Signifikant" heuristisch bestimmt
    - Ergibt „Maximal Segment Pairs (MSP)"
    - Die besten v MSP sind das Ergebnis

# Bemerkungen

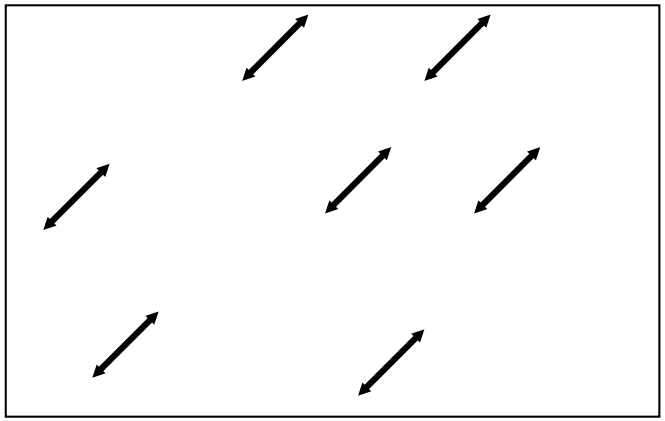
- BLAST ist eine Exklusionsmethode
  - Bestimme und suche nach Seeds = minimale Alignments, die (höchstwahrscheinlich) im Kern jedes sehr guten lokalen Alignments stecken
  - Erweitere nur diese zu Alignments für die gesamte Suchsequenz (bzw. deren „lokalen" Kern)
  - Praktisch alle Heuristiken zur Sequenzsuche arbeiten nach diesem Muster
    - Quasar, Biohunter, BLAT, FASTA, ...
- Heuristik
  - BLAST findet i.d.R. nicht alle hinreichend guten Alignments
  - Grund: Keine INSDEL, Schwellwerte w und t

# BLAST-2

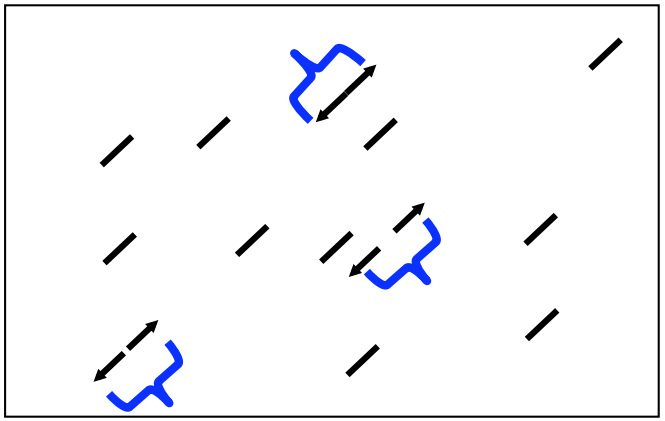
- 7 Jahre nach der Originalveröffentlichung
  - Altschul, Madden, Schaffer, Zhang, Zhang, Miller, Lipman: „Gapped BLAST and PSI-BLAST: a new generation of protein database search programs“, NAR, 1997
- Zwei Verbesserungen
  - Performance verbessern
    - Denn: Sequenzdatenbanken wachsen schneller als Geschwindigkeit der Computer
  - Gaps beachten
    - Denn: Mehrere kurze Alignments mit Gaps werden von BLAST-1 übersehen, wenn keines davon signifikant bzgl. t ist
    - Zusammen können diese Alignments aber hochsignifikant sein
    - BLAST-1 liefert bei mehrere, nahe beieinander liegenden MSP auch mehrere Ergebnisse (statt einem größeren)

# Illustration

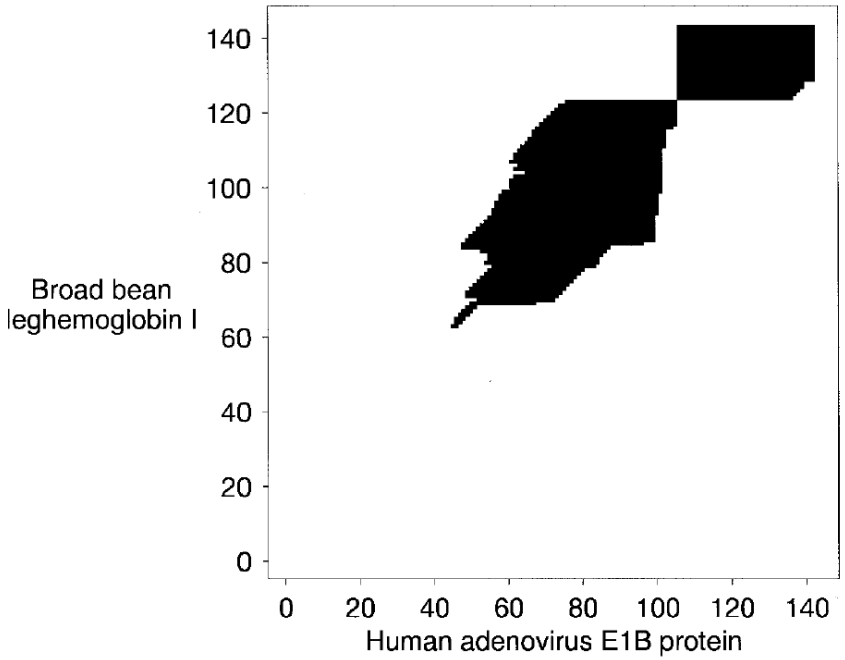
Blast-1



Blast-2



# Lokales Smith-Waterman




- SW ausgehend vom Seed-Point in beide Richtungen
- Abbruch bei Unterschreiten bestimmter Schranken
  - Abhängig von bisherigen besten Treffern etc.

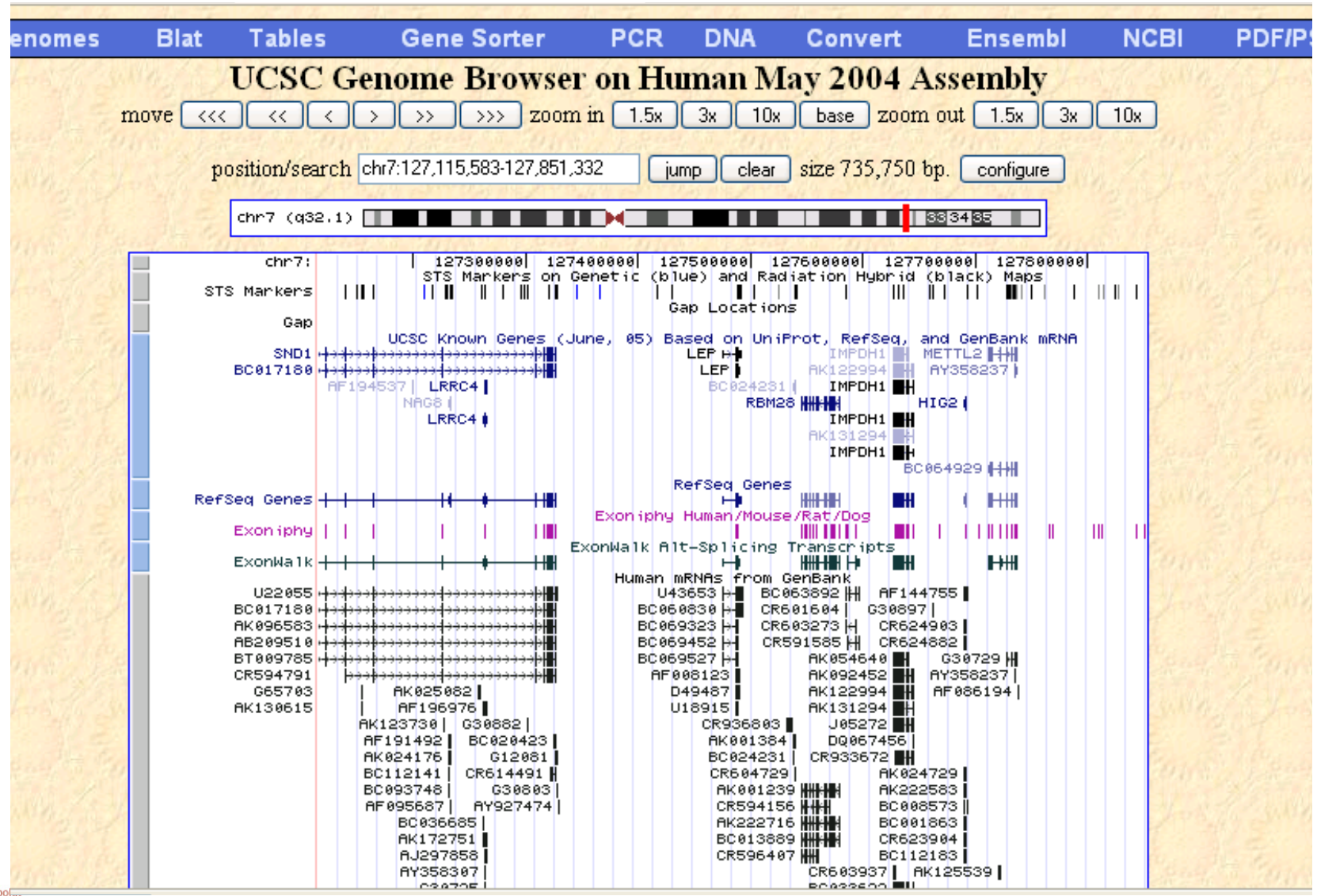
# Inhalt dieser Vorlesung

- BLAT
  - Datenbanksuche für sehr ähnliche Sequenzen
- Multiple Sequence Alignment (MSA)
  - Problem und eine Lösung
  - Heuristik: Clustal W

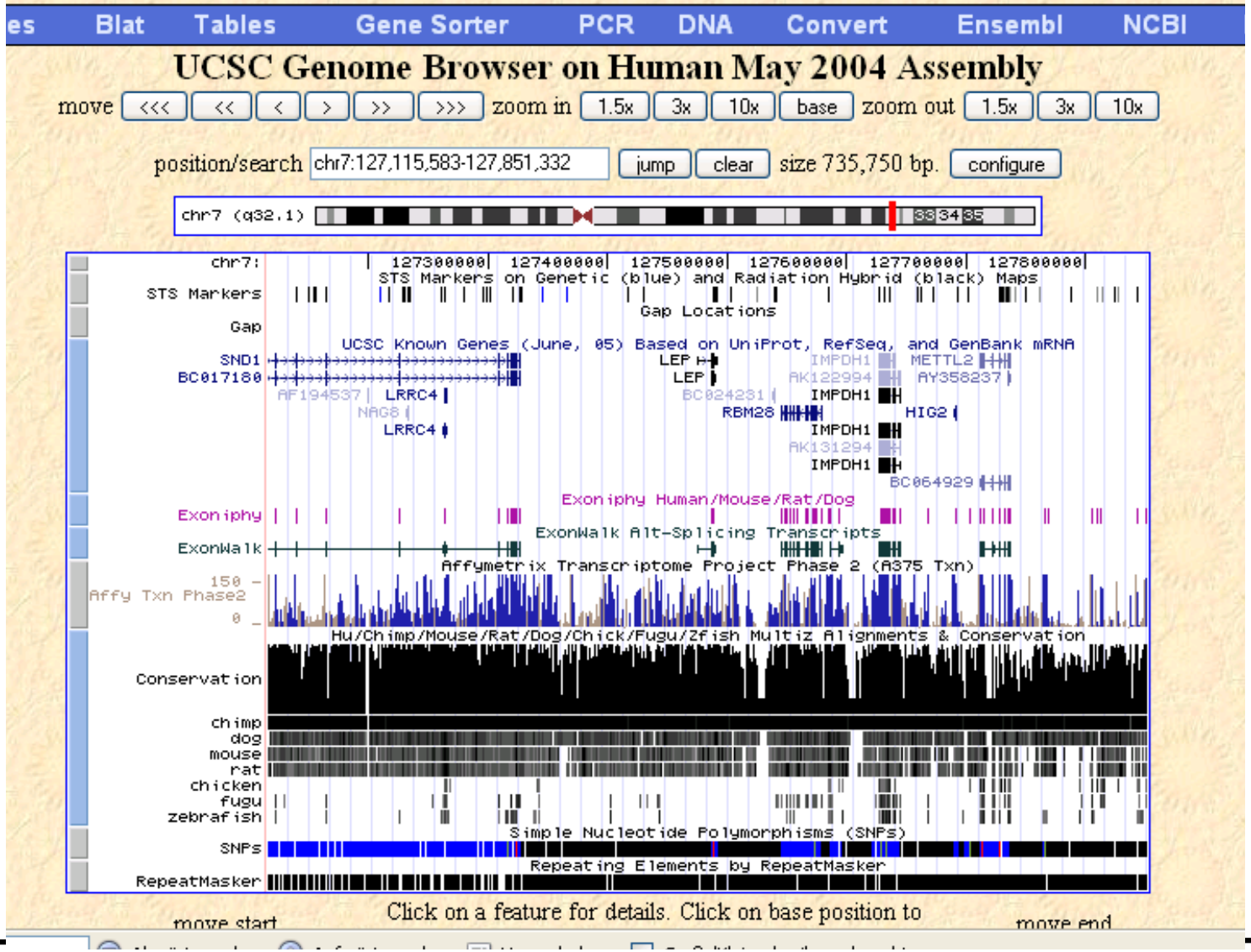
# BLAT

- Kent, W. J. (2002). "BLAT - the BLAST-like alignment tool." *Genome Res* **12**(4): 656-64.
  - Viel schneller als BLAST
  - Nur bei sehr ähnlichen erwarteten Alignments möglich
  - Verwendung im Bereich EST / cDNA
  - Verwendet im Genome Browser

# Genome Browser 1



# Screenshot 2

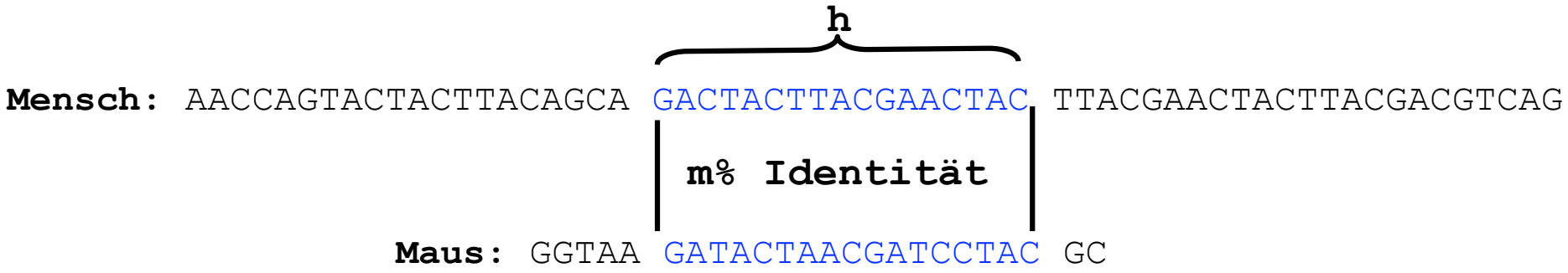


# Motivation

- Warum sehr schnelle Alignments?
  - Transcript Mapping
    - $2*10^9$ humane EST-Basen gegen das humane Genom ($2.9*10^9$ Basen)
  - Comparative Genomics
    - $1.7*10^6$ Maus-Reads gegen das humane Genom
  - Alle Maus-Human ESTs miteinander vergleichen
- BLAST ist nicht schnell genug
  - Neue Daten kommen schneller, als Alignments berechnet werden
- BLAT ist 500-mal schneller als BLAST und gleich sensitiv für sehr ähnliche Sequenzen
  - Wir wollen nur hoch konservierte Alignments sehen
  - Erlaubt deutlich höhere Anforderungen an Seeds und Gaps

# BLAT Szenario

- Vergleich einer Maus-cDNA Q mit humanen cDNAs
- Hintergrundwissen über Menschen und Mäuse
  - Wenn es eine homologe Teilsequenz gibt, wird diese im Schnitt zu m% identisch sein und eine durchschnittliche Länge von h haben
  - Frameshifts (Insertions/Deletions) sehr unwahrscheinlich
- Frage
  - Wie lang müssen Seeds aus Q sein, damit wir mit sehr hoher Wahrscheinlichkeit mindestens einen Treffer in einer homologen Region finden, wenn es diese gibt?



# Keine Gaps

- BLAT sucht nach Seeds ohne Gaps
  - Ungapped Alignment ist gut geeignet für cDNA / Genom Mapping
    - Lange Gaps führen zu mehreren Treffern
    - Kurze Gaps sind unwahrscheinlich - Frameshifts
  - Ungapped-Alignment ist viel schneller als Gapped-Alignment
    - Linear versus quadratisch
- Nehmen wir eine feste Seedlänge k an
- BLAT vergleicht alle überlappenden k-Mere von Q mit allen nicht-überlappenden k-mere in DB Sequenz T
  - Die Anzahl nicht-überlappender k-mere in einer homologen Region ist $z \sim h/k$
- Gefundene Teilalignments werden in der BLAT Endphase heuristisch zu einem Gesamtalignment zusammengefasst

# BLAT – Statistik

- Ziel: Maximales k und Anzahl von Hits abschätzen
  - m: Erwartete Anzahl Matches in zwei Sequenzen
    - Z.B.: 99% für Maus-Mensch cDNAs, 90% für Proteine
  - h: Durchschnittliche Länge homologer Regionen
  - g: Größe der Datenbank (in Basen)
  - q: Länge der Querysequenz
  - a: Größe des Alphabets (DNA oder Protein)
- Berechnung
  - Wahrscheinlichkeit, dass ein beliebiges k-mer aus der Suchsequenz mit seinem Gegenstück in der homologen Datenbanksequenz perfekt matched
    - $p_1 = m^k$
  - Wahrscheinlichkeit, dass mindestens ein nicht-überlappendes k-mer aus T mit dem entsprechenden k-mer in Q perfekt matched (wenn Q,T homolog)
    - $p = 1-(1-p_1)^z = 1-(1-m^k)^z$

# Trefferwahrscheinlichkeiten

**Table 3.** Sensitivity and Specificity of Single Perfect Nucleotide K-mer Matches as a Search Criterion

| | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|
| **A.** 81% | 0.974 | 0.915 | 0.833 | 0.726 | 0.607 | 0.486 | 0.373 | 0.314 |
| 83% | 0.988 | 0.953 | 0.897 | 0.815 | 0.711 | 0.595 | 0.478 | 0.415 |
| 85% | 0.996 | 0.978 | 0.945 | 0.888 | 0.808 | 0.707 | 0.594 | 0.532 |
| 87% | 0.999 | 0.992 | 0.975 | 0.942 | 0.888 | 0.811 | 0.714 | 0.659 |
| 89% | 1.000 | 0.998 | 0.991 | 0.976 | 0.946 | 0.897 | 0.824 | 0.782 |
| 91% | 1.000 | 1.000 | 0.998 | 0.993 | 0.981 | 0.956 | 0.912 | 0.886 |
| 93% | 1.000 | 1.000 | 1.000 | 0.999 | 0.995 | 0.987 | 0.968 | 0.957 |
| 95% | 1.000 | 1.000 | 1.000 | 1.000 | 0.999 | 0.998 | 0.994 | 0.991 |
| 97% | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 0.999 |

- q=100, m und k variabel (a und g hier nicht notwendig)
- Werte sind Wahrscheinlichkeit, dass mindestens ein perfekter Match in der Region vorkommt
- Ein Match reicht – Extensionsphase wird die ganze Region finden
- Beispiel
  - Bei erwarteter Sequenzähnlichkeit von m=97% findet man in einer homologen Region T von 100 Basen praktisch immer einen perfekten Match der Länge 13 für einen Substring der Länge q

# Falsch-positive Treffer

- Aber: Wie viele k-mere matchen zufällig?
  - Abhängig von g, q und a
  - F= (q-k+1) * (g/k) * $(1/a)^k$
    - $(1/a)^k$ : Alle Zeichen eines k-mers matchen per Zufall
    - (g/k): Anzahl nicht-überlappender k-mere in DB
    - (q-k+1): Anzahl (überlappender) k-mere in Query

| B. K | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|
| F | 1.3e+07 | 2.9e+06 | 635783 | 143051 | 32512 | 7451 | 1719 | 399 |

a=4, g=$3*10^9$, q=500

- Falsch-positive werden in Extensionsphase ausgesiebt
- Trade-Off
  - Hohe k-Werte: Wenig falsch-positive, aber eventuell fehlende echte Hits
  - Niedrige k-Werte: Viele falsch-positive, aber weniger falsch-negative

# Variante 1 – Hits mit Mismatches

- BLAT kann auch Hits mit höchstens einem Mismatch erlauben
  - Wahrscheinlichkeit, dass mindestens ein k-mer in einer homologen Regionen perfekt oder mit einem Mismatch matched
    - $P_1 = k*m^{k-1}*(1-m) + m^k$
  - Restliche Formeln entsprechend
- Ergebnis
  - Wesentlich längere Seeds möglich
  - Dafür wird die Suche erschwert (Indizierung kompliziert)

**Table 5.** Sensitivity and Specificity of Single Near-Perfect (One Mismatch Allowed) Nucleotide K-mer Matches as a Search Criterion

| | | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A.** | 81% | 0.945 | 0.880 | 0.831 | 0.721 | 0.657 | 0.526 | 0.465 | 0.408 | 0.356 | 0.255 | 0.218 |
| | 83% | 0.975 | 0.936 | 0.904 | 0.820 | 0.770 | 0.649 | 0.591 | 0.535 | 0.480 | 0.361 | 0.318 |
| | 85% | 0.991 | 0.971 | 0.954 | 0.900 | 0.865 | 0.767 | 0.719 | 0.669 | 0.619 | 0.490 | 0.445 |
| | 87% | 0.997 | 0.990 | 0.983 | 0.954 | 0.935 | 0.867 | 0.833 | 0.796 | 0.757 | 0.634 | 0.591 |
| | 89% | 1.000 | 0.997 | 0.995 | 0.984 | 0.976 | 0.939 | 0.920 | 0.897 | 0.872 | 0.775 | 0.741 |
| | 91% | 1.000 | 1.000 | 0.999 | 0.996 | 0.994 | 0.979 | 0.971 | 0.962 | 0.950 | 0.890 | 0.869 |
| | 93% | 1.000 | 1.000 | 1.000 | 0.999 | 0.999 | 0.996 | 0.994 | 0.991 | 0.988 | 0.963 | 0.954 |
| | 95% | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 0.999 | 0.999 | 0.999 | 0.994 | 0.992 |
| | 97% | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| **B.** | K | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| | F | 275671 | 68775 | 17163 | 4284 | 1070 | 267 | 67 | 17 | 4.2 | 1.0 | 0.3 |

A: q=100; B: a=4, g=3*10^9, q=500

# Weitere Möglichkeiten

- Suche mit mehreren (perfekten) Hits
  - Idee von BLAST-2
  - Wir sparen uns die Formeln
  - Drastische Verringerung der erwarteten Anzahl falsch-positiver Hits

**Table 7.** Sensitivity and Specificity of Multiple (2 and 3) Perfect Nucleotide K-mer Matches as a Search Criterion

| | | 2,8 | 2,9 | 2,10 | 2,11 | 2,12 | 3,8 | 3,9 | 3,10 | 3,11 | 3,12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **A.** | 81% | 0.681 | 0.508 | 0.348 | 0.220 | 0.129 | 0.389 | 0.221 | 0.112 | 0.051 | 0.021 |
| | 83% | 0.790 | 0.638 | 0.475 | 0.326 | 0.208 | 0.529 | 0.339 | 0.193 | 0.099 | 0.045 |
| | 85% | 0.879 | 0.762 | 0.615 | 0.460 | 0.318 | 0.676 | 0.487 | 0.313 | 0.180 | 0.093 |
| | 87% | 0.942 | 0.866 | 0.752 | 0.611 | 0.461 | 0.809 | 0.649 | 0.470 | 0.305 | 0.177 |
| | 89% | 0.978 | 0.940 | 0.868 | 0.761 | 0.625 | 0.910 | 0.801 | 0.648 | 0.476 | 0.314 |
| | 91% | 0.994 | 0.980 | 0.947 | 0.884 | 0.787 | 0.969 | 0.914 | 0.815 | 0.673 | 0.505 |
| | 93% | 0.999 | 0.996 | 0.986 | 0.962 | 0.912 | 0.993 | 0.976 | 0.933 | 0.851 | 0.722 |
| | 95% | 1.000 | 1.000 | 0.998 | 0.993 | 0.979 | 0.999 | 0.997 | 0.987 | 0.961 | 0.902 |
| | 97% | 1.000 | 1.000 | 1.000 | 1.000 | 0.999 | 1.000 | 1.000 | 0.999 | 0.997 | 0.987 |
| **B.** | N,K | 2,8 | 2,9 | 2,10 | 2,11 | 2,12 | 3,8 | 3,9 | 3,10 | 3,11 | 3,12 |
| | F | 524 | 27 | 1.4 | 0.1 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 |

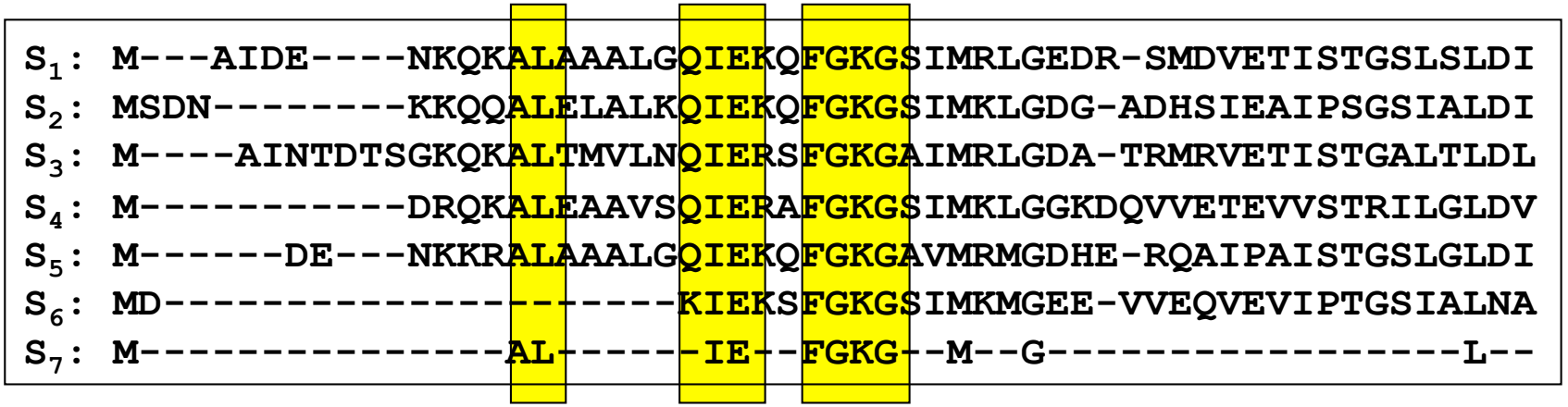
# Inhalt dieser Vorlesung

- BLAT
  - Datenbanksuche für sehr ähnliche Sequenzen
- Multiple Sequence Alignment (MSA)
  - Problem und eine Lösung
  - Heuristik: Clustal W

# Definition

- Bisher
  - Immer Vergleich zweier Strings
- Jetzt
  - Multipler Stringvergleich: Vergleich von k>2 Strings
- Definition
  - *Ein multiples Sequenzalignment (MSA) von k Strings $S_i$, $1 \leq i \leq k$, ist eine Tabelle mit k Zeilen und l Spalten, so dass*
    - *In Zeile i steht String $S_i$, mit beliebig eingefügten Leerzeichen*
    - *Jedes Zeichen jedes $S_i$ steht in exakt einer Spalte*
    - *In keiner Spalte stehen nur Leerzeichen*
- Bemerkungen
  - Direkte Generalisierung des Alignment zweier Strings
  - Es folgt, dass $l = |MSA| \leq \Sigma(|S_i|)$
    - Warum?

# Beispiel



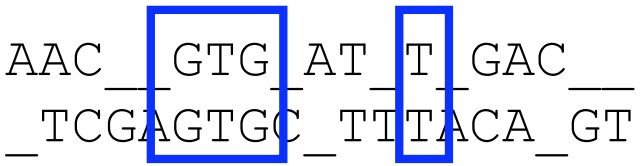

- Uns interessieren „möglichst gute“ MSAs
  - Intuition
    - Möglichst wenig Spalten – wenig Leerzeichen
    - Möglichst homogene Spalten – hohe Übereinstimmung
  - Exakte Definition später
- MSAs erfassen das Gemeinsame verschiedener Sequenzen

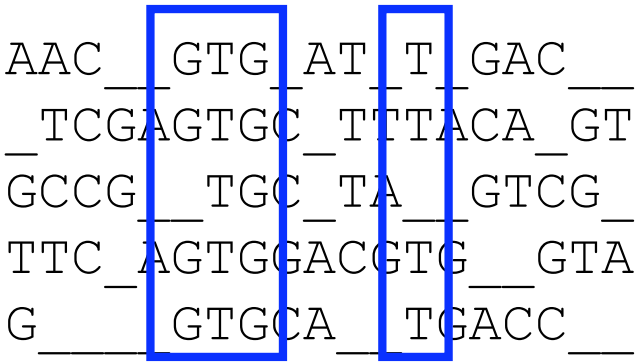
# Motivation

- Alignment sucht ähnliche Sequenzen
  - Weil: ähnliche Sequenz – ähnliche Struktur – ähnliche Funktion
- MSA sucht „das Ähnliche" in vielen Sequenzen
  - Argumentationsrichtung ist umgekehrt
  - Auch beim paarweisen Alignment findet man Regionen guter Übereinstimmung – aber nur für diese zwei Sequenzen
  - MSA startet mit vielen Sequenzen, bei denen man ähnliche Funktion / Struktur vermutet
  - MSA stellt fest, was das Gemeinsame dieser Sequenzen ist – Domänen, Motive, Signaturen, Profile, ...
  - These: dieses Gemeinsame ist biologisch relevant

# Motivation II

- Proteine (und damit auch DNA) setzen sich aus funktionalen Blöcken und „Zwischenraum“ zusammen
  - Die Blöcke findet man nicht, wenn man Sequenzen nur paarweise vergleicht
    - Bzw. man kann sie nicht vom Rauschen unterscheiden
- Trennung des eventuell zufällig Gemeinsamen (Alignment) vom bedeutungsvoll Gemeinsamen (MSA)



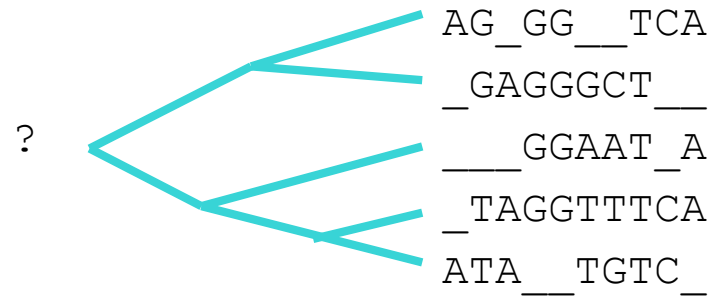



# Blöcke, Domänen, Sites

- Proteine
  - Bindungsstelle für andere Proteine / Liganden / Moleküle
  - Bindungsstelle an DNA
  - Phosphorylierung / Dephosphorylierungsstelle
  - Signal zum Transport des Proteins
  - Signal zum Abbau des Proteins
  - ...
- DNA
  - Bindungsstellen für Proteine
  - Promotoren und Inhibitoren
  - Strukturtragende Bereiche
  - Start- und Stoppcodons
  - Signal für differenzielles Splicen
  - ...

# MSA Zielfunktion

- Zielfunktion beim einfachen Alignment war klar
  - Möglichst wenig I,R,D
  - Eventuell mit Substitutionsmatrix
  - Eventuell mit spezieller Behandlung von Gaps
- Zielfunktion für MSA ist nicht so klar
  - Score einer Spalte mit 2 T, zwei G und einem Leerzeichen?
  - Angabe einer Substitutionsmatrix für k Sequenzen über Alphabet $\Sigma$ würde $O(|\Sigma|^{k+1})$ Werte erfordern
  - Nicht machbar und biologisch nicht begründbar



# MSA Überblick

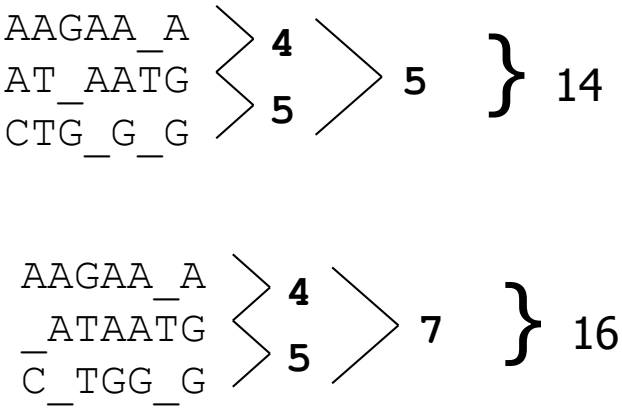
- Weg über Substitutionsmatrizen nicht gangbar
- Verschiedene alternative Vorschläge für Zielfunktionen existieren
  - Maximiere die Summe aller paarweisen Alignments
  - Maximiere die Summe der Alignments jeder Sequenz zu einer Consensussequenz (Center-Star)
  - Maximiere die Summe der Alignments folgend dem phylogenetischen Baum der Sequenzen

# Definitionen

- Definition
  - *Gegeben ein MSA M für Sequenzen $S_1,...,S_k$. Das durch M induziertes Alignment für zwei Sequenzen $S_i$ und $S_j$ ist das folgende:*
    - *Entferne aus M alle Zeilen außer i und j*
    - *Entferne alle Spalten, die in i und j ein Leerzeichen enthalten*
    - *Berechne den Alignmentscore für Sequenzen i und j*
  - *Gegeben ein MSA M für Sequenzen $S_1,...,S_k$. Der Sum-Of-Pairs Score für M (SP-Score) ist die Summe aller Alignmentscores der durch M induzierten paarweisen Alignments*
  - *Das SP-Alignment Problem für Sequenzen $S_1,...,S_k$ sucht das MSA M mit minimalem SP-Score*
- Bemerkung
  - Vergleich aller Sequenzen mit allen anderen Sequenzen – aber entsprechend dem vorgegebenen MSA

# Beispiel

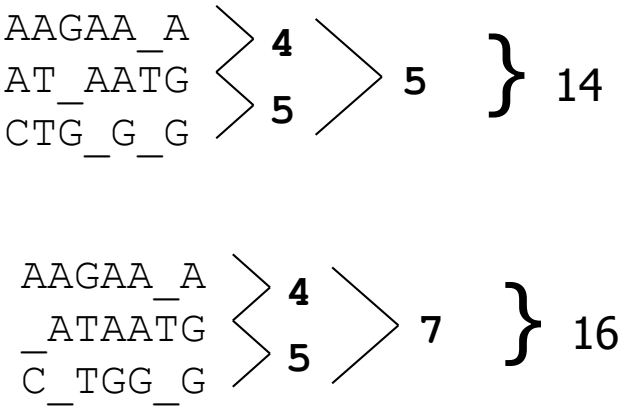
| | | |
|---|---|---|
| d/i | = | 1 |
| r | = | 1 |
| m | = | 0 |




- Die Berechnung des SP-Scores für ein gegebenes MSA der Länge l über k Sequenzen ist einfach
  - Nämlich?

# Beispiel

| | |
|---|---|
| d/i | = 1 |
| r | = 1 |
| m | = 0 |



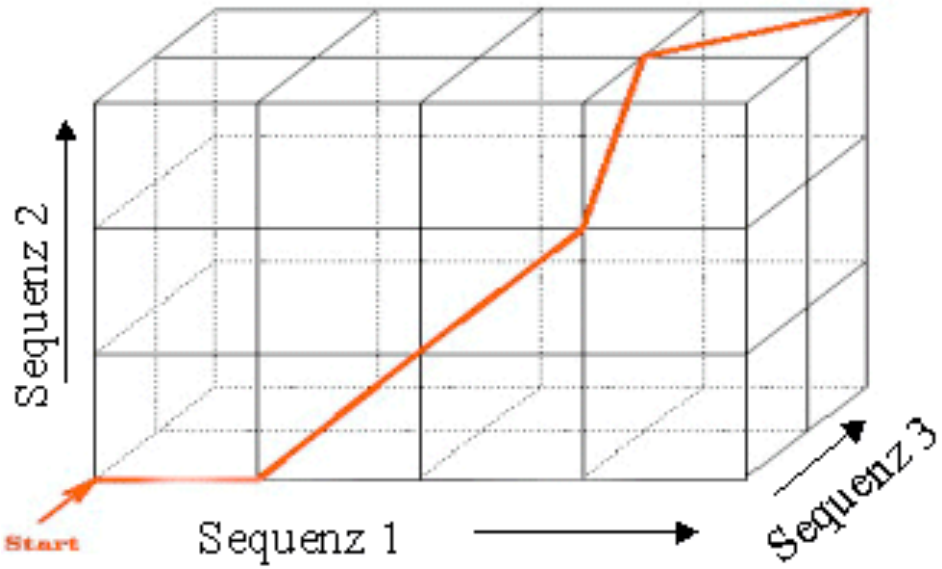

- Die Berechnung des SP-Scores für ein gegebenes MSA der Länge l über k Sequenzen ist einfach
  - Komplexität $O(l*k^2)$
- Aber wie findet man das MSA mit minimalem SP-Score?

# Dynamische Programmierung in k Dimensionen

- k = 2
  - 2-dimensionale Matrix

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| | | | w | r | i | t | e | r | s |
| 0 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | v | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | i | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 7 |
| 3 | n | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 6 |
| 4 | t | 4 | 4 | 4 | 4 | 3 | 4 | 5 | 6 |
| 5 | n | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 6 |
| 6 | e | 6 | 6 | 6 | 6 | 5 | 4 | 5 | 6 |
| 7 | r | 7 | 7 | 6 | 7 | 6 | 5 | 4 | 5 |

- k = 3
  - 3-dimensionale Matrix

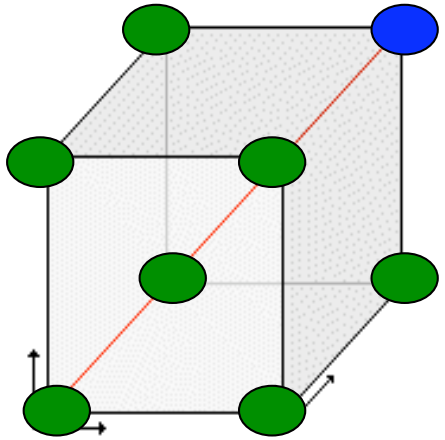


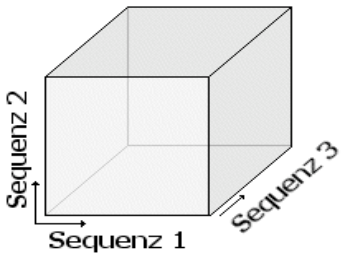
# Erinnerung

- Grundidee der dynamischen Programmierung für zwei Sequenzen $S_1$, $S_2$
  - Berechnung des Alignment d(i,j) von $S_1[1..i]$ und $S_2[1..j]$ für steigende Werte (i, j) bis i=$|S_1|$ und j=$|S_2|$
  - Berechnung von d(i,j) aus d(i-1,j-1), d(i,j-1), d(i-1,j)
    - Man verlängert d(i-1,j-1) um Match oder Mismatch
    - ... oder man verlängert d(i,j-1) um ein Insert
    - ... oder man verlängert d(i-1,j) um eine Deletion
  - Initialisierung der Werte d(i,0) und d(0,j)
- Wir betrachten im Folgenden nur den Fall k=3

# Dynamische Programmierung für SP-MSA

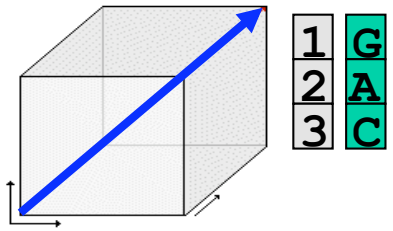
- Übertragung auf MSA
- Berechnung von d(i,j,k) aus
  - d(i-1,j-1,k-1)
  - d(i,j-1,k-1)
  - d(i,j,k-1)
  - d(i,j-1,k)
  - d(i-1,j,k)
  - d(i-1,j-1,k)
  - d(i-1,j,k-1)

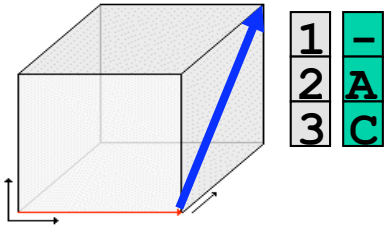


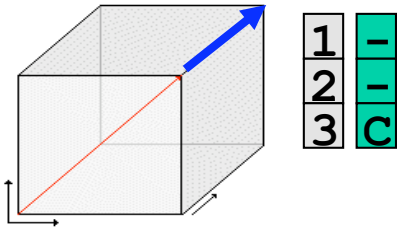


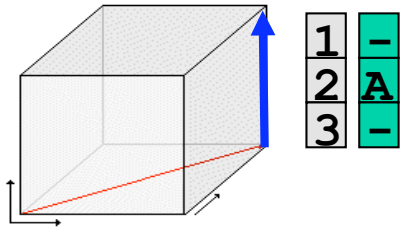

# Dyn Prog. für SP-MSA

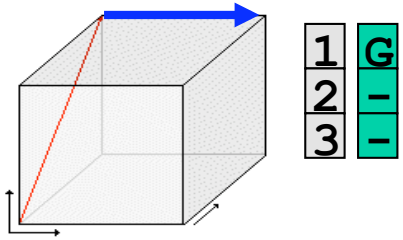
- d(i-1,j-1,k-1)
- d(i,j-1,k-1)
- d(i,j,k-1)
- d(i,j-1,k)
- d(i,j,k)
- d(i-1,j-1,k)
- d(i-1,j,k-1)

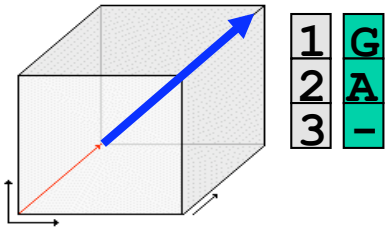


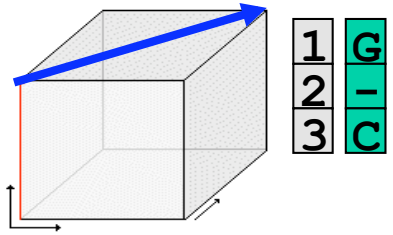



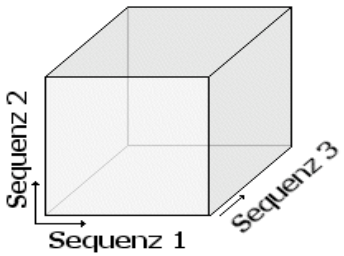



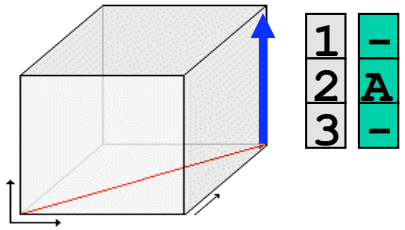





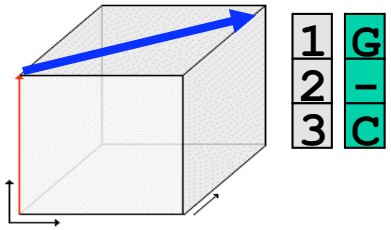

# Analogie

**d(i,j-1,k)**




- SP-Alignment von d(i,j-1,k) ist bekannt
- Wir erweitern dieses zu d(i,j,k)
- Dazu alignieren wir $S_2[j]$ zweimal mit Leerzeichen (Inserts)

**d(i-1,j,k-1)**




- SP-Alignment von d(i-1, j,k-1) ist bekannt
- Wir erweitern dieses zu d(i,j,k)
- Dazu alignieren wir ein Leerzeichen mit $S_1[i-1]$ und mit $S_3[k-1]$

# Formal

- Wir nehmen ein einfaches Kostenmodell (I/D/R=1, M=0)
- Theorem
  - *Gegeben Sequenzen $S_1$, $S_2$, $S_3$.*
    - *Sei d(i,j,k) der Score des SP-optimalen Alignments der Strings $S_1[1..i]$, $S_2[1..j]$, $S_3[1..k]$*
    - *Sei $c_{ij} = 0$, wenn $S_1(i) = S_2(j)$, sonst 1*
    - *Sei $c_{ik} = 0$, wenn $S_1(i) = S_3(k)$, sonst 1*
    - *Sei $c_{jk} = 0$, wenn $S_2(j) = S_3(k)$, sonst 1*
  - *Dann berechnet sich d(i,j,k) als*

$$d(i,j,k) = \min \begin{cases} d(i-1,j-1,k-1) + c_{ij} + c_{ik} + c_{jk} \\ d(i-1,j-1,k) + c_{ij} + 2 \\ d(i-1,j,k-1) + c_{ik} + 2 \\ d(i,j-1,k-1) + c_{jk} + 2 \\ d(i-1,j,k) + 2 \\ d(i,j-1,k) + 2 \\ d(i,j,k-1) + 2 \end{cases}$$

# Randbedingungen

- Theorem Fortsetzung
  - *...*
  - *mit Initialisierung*
    - *Sei $d_{a,b}(i,j)$ der optimale Alignmentscore von $S_a[1..i]$ mit $S_b[1..j]$*
    - *$d(0, 0, 0) = 0$*
    - *$d(i, j, 0) = d_{1,2}(i, j) + (i+j)$*
    - *$d(i, 0, k) = d_{1,3}(i, k) + (i+k)$*
    - *$d(0, j, k) = d_{2,3}(j, k) + (j+k)$*
- Bemerkung
  - Beweis analog zum paarweisen Alignment
  - Alignment eines Leerzeichen mit einem Leerzeichen ist im induzierten paarweisen Alignment nicht enthalten

# Algorithmus

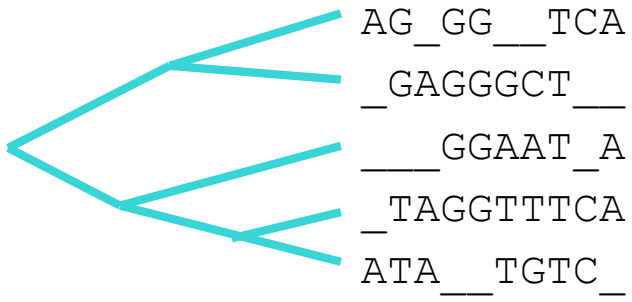
```
initialize matrix d;
for i := 1 to |S1|
    for j := 1 to |S2|
        for k := 1 to |S3|
            if (S1(i) = S2(j)) then cij := 0; else cij := 1;
            if (S1(i) = S3(k)) then cik := 0; else cik := 1;
            if (S2(j) = S3(k)) then cjk := 0; else cjk := 1;

            d1 := d[i - 1,j - 1,k - 1] + cij + cik + cjk;
            d2 := d[i - 1,j - 1,k] + cij + 2;
            d3 := d[i - 1,j,k - 1] + cik + 2;
            d4 := d[i,j - 1,k - 1] + cjk + 2;
            d5 := d[i - 1,j,k) + 2;
            d6 := d[i,j - 1,k) + 2;
            d7 := d[i,j,k - 1) + 2;

            d[i,j,k] := min(d1, d2, d3, d4, d5, d6, d7);
        end for;
    end for;
end for;
```

# Komplexität

- Für drei Sequenzen der Länge n
  - Würfel hat $n^3$ Zellen
  - Für jede Zelle sind 7 Berechnungen notwendig
  - Zusammen $O(7*n^3)$
- Allgemeiner Fall: k Sequenzen der Länge n
  - Hyperwürfel hat $n^k$ Zellen
  - Für jede Zelle sind $2^k$-1 Berechnungen notwendig
    - Alle Ecken eines k-dimensionalen Würfels minus eins (Das ist die Ecke die gerade berechnet wird)
  - Zusammen $O(2^k * n^k)$
- Tatsächlich gilt
  - *Das SP-Alignment Problem ist NP vollständig*

# MSA also praktisch unlösbar?

- SP-Score für mehr als eine Handvoll Sequenzen nennenswerter Länge nicht berechenbar
- Außerdem: SP berechnet nicht die Menge an notwendiger Evolution




- Viele andere Zielfunktionen
  - Center-Star, MSA entlang des phylogenetischen Baums
- Viele Heuristiken (Branch&Bound, iterative, greedy, …)

# Inhalt dieser Vorlesung

- BLAT
  - Datenbanksuche für sehr ähnliche Sequenzen
- Multiple Sequence Alignment (MSA)
  - Problem und Lösung
  - Heuristik: Clustalw

# CLUSTAL W

- Greedy-MSA mit einem phylogenetischen Baum
- Lange Zeit das Standardprogramm zum multiplen Sequenzalignment
  - Higgins, D. G. and Sharp, P. M. (1988). "CLUSTAL: a package for performing multiple sequence alignment on a microcomputer." *Gene* **73**(1): 237-44.
  - Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994). "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." *Nucleic Acids Res* **22**(22): 4673-80.
- Heute ist die Situation unübersichtlicher
  - DAlign, T-Coffee, HMMT, PRRT, MULTALIGN, ...

# Progressives Alignment

- Grundproblem des SP-MSA
  - Ständige Betrachtung aller Sequenzen
  - Das sind zu viele Möglichkeiten = Dimensionen
- Grundidee der progressiven Verfahren
  - Berechne zunächst MSA für viele kleine Teilmengen von Sequenzen
  - Baue das Gesamt-MSA aus den Teil-MSA
- Das wirft Fragen auf
  - Wie wähle ich die Teilmengen?
  - Wie „aligniert" man zwei MSA?
    - Wir können MSA (Profil) und Sequenz, aber zwei MSA?
  - In welcher Reihenfolge verschmilzt man die Teil-MSA?

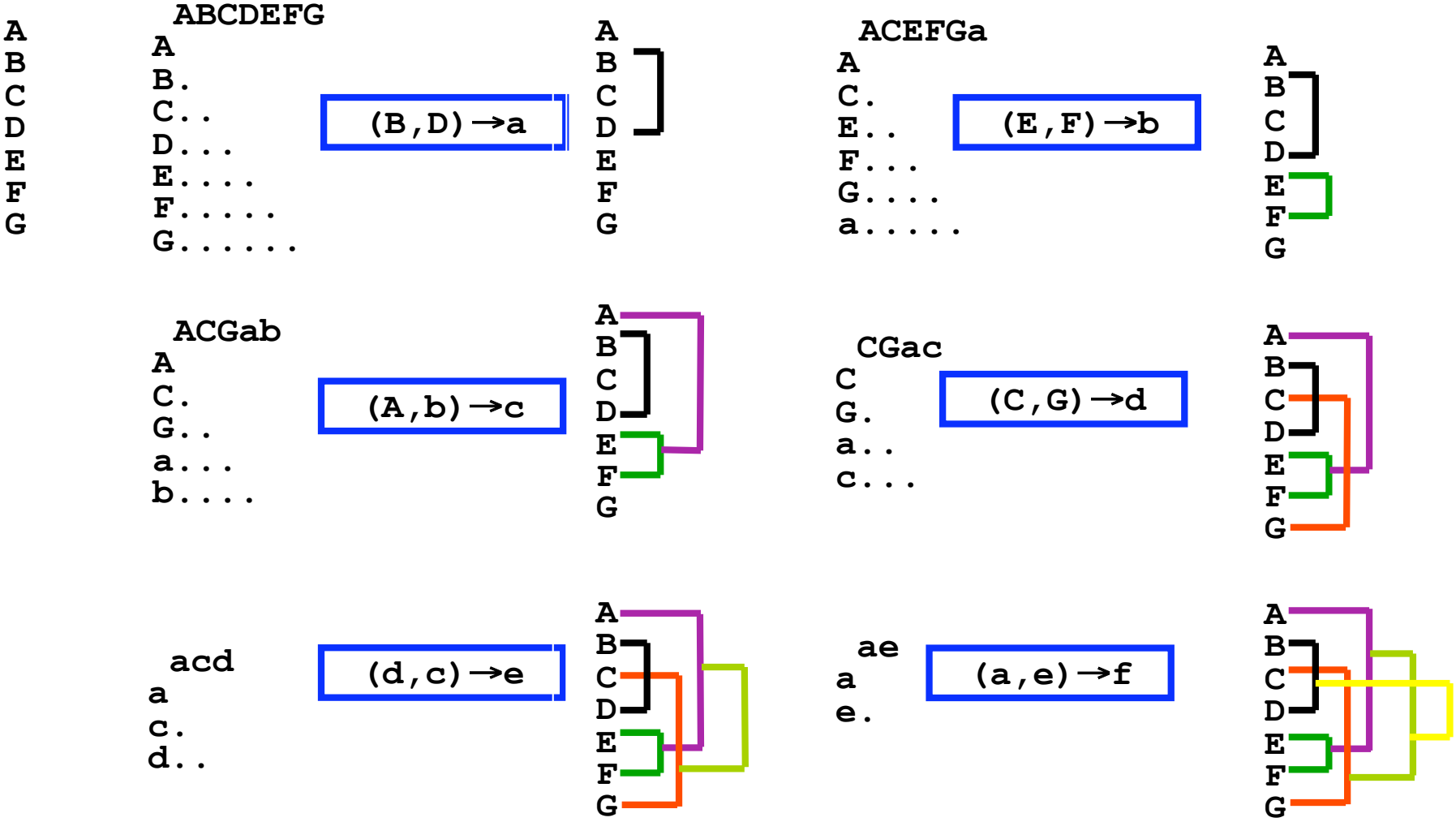
# CLUSTAL W: Grundaufbau

- Gegeben k Sequenzen
- Drei Schritte
  - Berechne alle paarweisen Alignmentscores
  - Konstruiere „Guide Tree" durch hierarchisches Clustering
  - Berechne und verschmelze Teil-MSA gemäß dem Guide Tree
- Idee dahinter
  - Aligniere erst sehr ähnliche Sequenzen – Signale werden verstärkt
  - Werden z.B. zwei sehr verschiedene Cluster von Sequenzen betrachtet, berechnet CLUSTAL automatisch erst zwei (homogene) MSA und verschmilzt diese am Ende
    - Hohe Chance, dass konservierte Blöcke erhalten bleiben
  - Außenseiter kommen erst spät dazu und können die Blockstruktur nicht mehr stören
  - Orientierung an der „tatsächlichen" Entstehungsgeschichte, dem phylogenetischen Baum

# Schritt 1 und 2

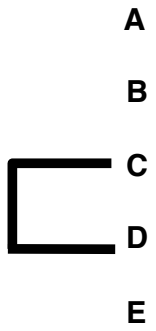
- Berechnen der Ähnlichkeitsmatrix M
  - Berechne die $O(k^2)$ paarweisen Alignmentscores für die k Sequenzen
- Hierarchisches Clustering
  - Wähle Zelle (i,j) mit kleinstem Abstand aus Matrix M
    - Das ist das erste Paar
  - Erzeuge M‘: Lösche die Sequenzen i und j aus M und füge neue Spalte/Zeile (ij) ein
  - Für alle $k \neq ij$: M‘[ij,k] = (M[i,k] + M[j,k]) / 2
    - Mittlerer Abstand zu i und j
  - Iteriere, bis Matrix nur noch 2x2 groß ist

# Beispiel: Graphisch

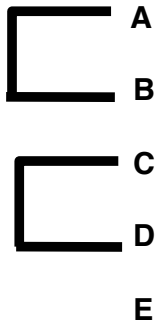


# Konstruktion des Guide Trees

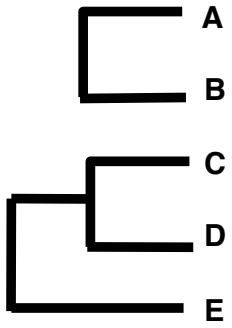
| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | | 17 | 59 | 59 | 77 |
| B | | | 37 | 61 | 53 |
| C | | | | 13 | 41 |
| D | | | | | 21 |

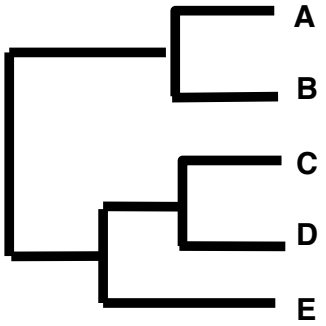



| | A | B | E | CD |
|---|---|---|---|---|
| A | | 17 | 77 | 59 |
| B | | | 53 | 49 |
| E | | | | 31 |




| | E | CD | AB |
|---|---|---|---|
| E | | 31 | 65 |
| CD | | | 54 |



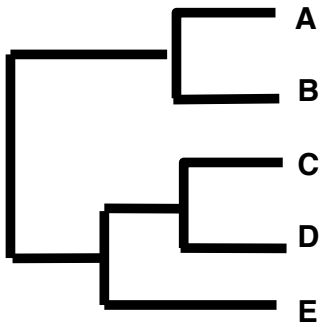
# Schritt 3: Progressive MSA Generierung

- Berechnung paarweiser Alignments in der Reihenfolge des Guide Trees
- Alignment eines MSA $M_1$ mit einem MSA $M_2$
  - Dynamische Programmierung mit linearem Gapscore
  - Wert eines Mismatches/Matches ist der Durchschnittsscore aller Paare mit einem Zeichen aus $M_1$ und dem anderen aus $M_2$
  - Gaps werden mit dem schlechtesten Score der verwendeten Substitutionsmatrix bestraft
  - Bei k Sequenzen sind das maximal k/2*k/2 = $O(k^2)$ Scores
- Beispiel

```
A   ...P...
B   ...G...
C   ...P...

D   ...A...
E   ...A...
F   ...Y...
```

- Score dieser Spalte
  - `(2*s(P,A)+s(P,Y)+`
    `2*s(G,A)+s(G,Y)+`
    `2*s(P,A)+s(P,Y) )/9`

# Beispiel




```
C   PADKTNVKAAWGKVGAHAGEYGA
D   AADKTNVKAAWSKVGGHAGEYGA

A   PEEKSAVTALWGKVNVDEYGG
B   GEEKAAVLALWDKVNEEEYGG

C   PADKTNVKAAWG_KVGAHAGEYGA
D   AADKTNVKAAWS_KVGGHAGEYGA
E   AA__TNVKTAWSSKVGGHAPA__A

A   PEEKSAV_TALWG_KVN__VDEYGG
B   GEEKAAV_LALWD_KVN__EEEYGG
C   PADKTNVKAA_WG_KVGAHAGEYGA
D   AADKTNVKAA_WS_KVGGHAGEYGA
E   AA__TNVKTA_WSSKVGGHAPA__A
```

Once a gap, always a gap

# Erweiterungen

- Viele weitere Tricks
  - Individuelle Scores für das Öffnen eines Gaps in Abhängigkeit der Umgebung, Abstand zu anderen Gaps, Länge der Sequenz, ...
  - Verwendung unterschiedlicher Substitutionsmatrizen, je nachdem wie hoch man schon im Baum ist
    - Denn damit steigt der evolutionäre Abstand, und PAM-X bzw. BLOSUM-X Matrizen werden nach dem geschätzten Abstand gewählt
  - Gewichten der Sequenzen im MSA/MSA Alignment je nach Ähnlichkeit zu anderen Sequenzen
    - Verhindert, dass sehr ähnliche Sequenzen das ganze MSA dominieren
  - Verwendung von Neighbour Joining statt hierarchischem Clustering
- Ergebnis: Deutlich verbesserte Ergebnisse
  - Durch Benchmarking gezeigt
  - Orientiert sich am 3D Strukturalignment als Goldstandard