Algorithmische Bioinformatik

Exklusionsmethoden

BLAST

BLAT

Ulf Leser

Wissensmanagement in der Bioinformatik

Inhalt dieser Vorlesung

- Exklusionsmethode BYP
  - Auf dem Weg zur schnellen Datenbanksuche
- BLAST: Basic Local Alignment Search Tool
- BLAT: BLAST-like Alignment Tool

Bisher

- Alignment zweier Strings dauert O(n*m)
- K-Band Algorithmus benötigt $O(sn^2-vn)$ für |A|=|B|
  - Nicht gut für unähnliche Sequenzen
  - Klappt nur für globales Alignment
- Wir wollen ein schnelleres lokales Alignment
- Wir beginnen mit einem verwandtem Problem: k-Difference Matching

K-Difference Matching

- Definition.
  Ein k-Difference Vorkommen von P in T ist ein Substring T' von T für den gilt: ed(T', P) ≤ k
- Gesucht: Algorithmus, der alle k-Difference Vorkommen von P in T im Average Case in O(m) für geeignete k finden
  - Geeignetes k: klein verglichen zu |P|
  - Also: Lineares approximatives Stringmatching für bestimmte Fälle

AAGCGTGATAGCGGAGTA **AAGCGTGATAGCGGAGTA**

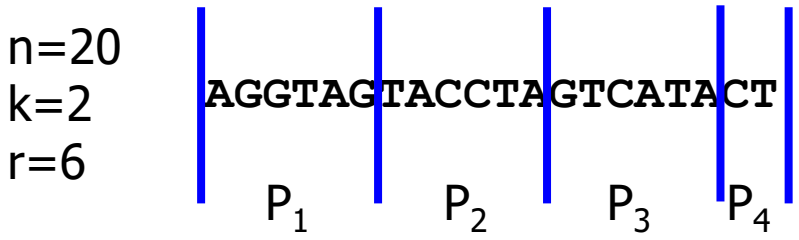
CGGA **CG_GA** **CGGA**

Exklusionsmethoden

- Baeza-Yates, Perleberg: „Fast and practical approximative string matching", LNCS 664, 1992
- Grundaufbau: Alle k-Difference Vorkommen von P in T
  - 1. Partition: Partitioniere P geschickt in Teilstrings P_i
  - 2. Search: Suche Teilstrings P_i exakt in T
    - Partitioniere so, dass jedes k-Difference Vorkommen um die Fundstelle mindestens eines Teilstrings liegen muss
    - Aber: Nicht jede Fundstelle ist Kern eines k-Difference Vorkommens
  - 3. Check: Prüfe alle potentiellen Vorkommen
- Einsparung: Im Average Case müssen nur wenige Bereiche von T näher (und teuer) untersucht werden

Partitionierung

- Lemma
  Sei T' ein k-Difference Vorkommen von P in T. Sei Z={P_1,...,P_{k+2}) eine Zerlegung von P in k+1 Teilstrings der Länge r = floor(n/(k+1)) und einen kürzeren Reststring. Dann muss T' mindestens eine Partition von P der Länge r exakt enthalten.



Beweis

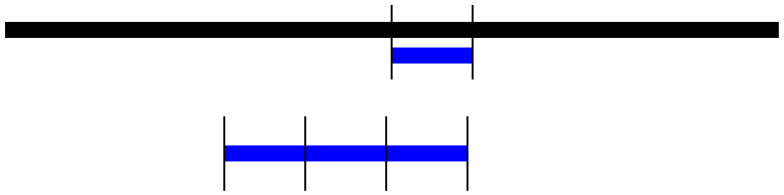
- Beweis
  - Nehmen wir an, dass T‘ existiert und P und T‘ optimal aligniert sind
  - Jede der k+1 Partitionen P_i von P der Länge r aligniert mit einem Substring T_i‘ von T‘
  - Wenn in jedem dieser k+1 Teilalignments mindestens ein Unterschied wäre, dann wären im Alignment von P mit T‘ mindestens k+1 Unterschiede
  - Dann wäre T‘ kein k-Difference Vorkommen
  - Also muss mindestens eines der Teilalignments fehlerfrei sein
  - qed.
- Bemerkungen
  - Die letzte Partition ignorieren wir
  - r ist so gewählt, dass mindestens ein Teilstück fehlerfrei sein muss

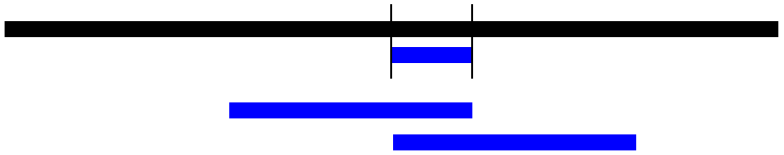
BYP – Search

- Wir suchen alle exakten Vorkommen der k+1 ersten Partitionen von P in T
- Wie machen wir das geschickt?
- Möglichkeit 1: Suffixbäume
  - Baue einen Suffixbaum für T in O(m)
  - Durchsuche den Baum mit allen Partitionen
  - Zusammen: O(m+n+k) [k: Zahl Matches]
  - Aber – dafür brauchen wir viel Platz
- Möglichkeit 2: Aho-Corasick Algorithmus
  - Baue einen Keyword Tree für die k+1 Partitionen
  - Durchsuche damit T in O(m+k) [k: Zahl Matches]
  - Zusammen: O(n+m+k)

BYP – Check Phase

- Sei i die Startposition einer Partition P‘ von P in T
- Wir müssen testen, ob um T[i; i+r-1] herum ein k-Difference Vorkommen von P liegt
- Ein solches Vorkommen kann im schlimmsten Fall n+k Zeichen lang sein





- Also alignieren wir T[i-n-k+r, i+n+k] mit P
  - Beachtet man die Position von P‘ in P, kann man das verbessern
- Komplexität: $O(n*(2*(n+k-r))) = O(n^2+nk) \sim O(n^2)$
  - Für sehr kleine k – unser Szenario

Alles zusammen

- Komplexität (für kleine k)
  - Partitionierung ist O(n)
  - Search ist O(n+m)
  - Check ist $O(n^2)$ für jedes Vorkommen einer Partition
- Der letzte Schritt ist entscheidend bzgl. unserer Wunschkomplexität
- Average Case Analyse
  - Annahme: Sei T eine zufällige Sequenz über Alphabet der Größe s
  - Dann ist die Wahrscheinlichkeit, dass ein konkreter String der Länge r an einer Position i in T beginnt gleich $1/s^r$

Average-Case Komplexität

- Theorem
  Für zufällige Strings P und T mit |P|=n und |T|=m und k~n/log$_s$(n) findet der BYP Algorithmus im Average Case alle k-Difference Vorkommen von P in T in O(m)
- Beweis
  - Sei P‘ eine der k+1 Partitionen von P der Länge r
    - T enthält O(m) potentielle Startpositionen von P‘
    - Damit ist P‘ im Schnitt m/s^r Mal in T enthalten
  - Über alle k+1 Partitionen erwarten wir damit (k+1)*m/s^r Vorkommen einer Partition in T
  - Für jede brauchen wir O(n^2); die Gesamtlaufzeit wollen wir aber in O(c*m) halten; wir suchen k so dass:
    - c: Konstante

$$\frac{mn^2(k+1)}{s^r} < cm$$

BYP – Komplexität Fortsetzung

- Beweis Fortsetzung
  - k ist immer kleiner als n; damit suchen wir den Punkt mit

$$\frac{mn^3}{s^r} = cm$$

  - Es folgt $s^r=n^3/c$ und damit $r = \log_s(n^3)-\log_s(c) \approx \log_s(n^3)$
  - Da r=floor(n/(k+1)) gewählt wurde, können wir einsetzen

$$\frac{n}{k+1} \approx \log_s n^3$$

  - Auflösen nach k ergibt

$$k \approx \frac{n}{\log_s n^3} \approx \frac{n}{3\log_s n} \in O\left(\frac{n}{\log_s n}\right)$$

  - q.e.d.

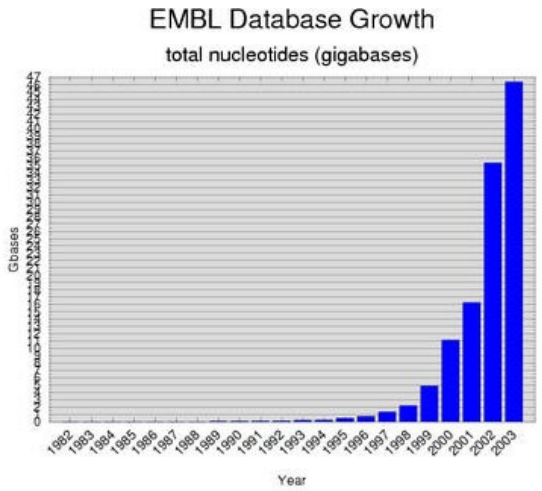
Zusammenfassung

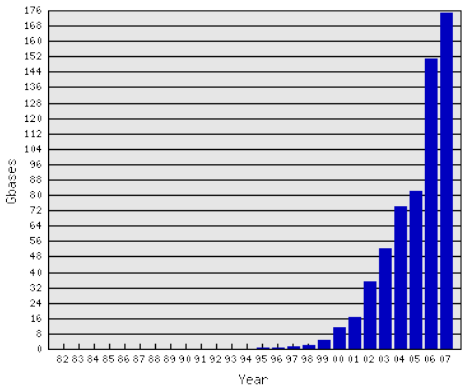
- Mit zusätzlichen Annahmen können wir schnellere Algorithmen für Stringmatchingprobleme bauen
- Grundidee (Partitionierung – exakte Suche – lokale Erweiterung) wird in vielen (allen?) heuristischen DB-Suchverfahren verwendet
  - Keine Fehlergrenze vorgegeben
  - Notwendigkeit zu noch höherer Geschwindigkeit
  - Preis: Nicht garantiert alle Matches finden

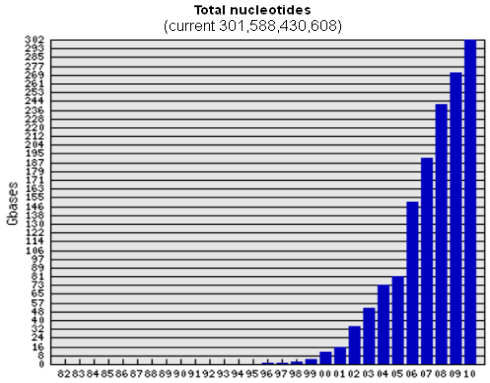
Inhalt dieser Vorlesung

- Exklusionsmethode BYP
- BLAST: Basic Local Alignment Search Tool
  - Man kann Googlen und Blasten
  - Altschul, Gish, Miller, Myers, Lipman: „Basic Local Alignment Search Tool", J Mol Bio, 1990.
  - Blast-1
  - Blast-2 (gapped Blast)
- BLAT: BLAST-like Alignment Tool

Heuristische Alignierung







- Annotation neuer Sequenzen basiert auf Suche nach homologen Sequenzen in Sequenzdatenbanken
- Datenmenge wächst exponentiell
- Selbst lineare Algorithmen sind zu langsam

Gedankenkette

- Gegeben P und Datenbank D mit Sequenzen S_1, ..., S_n
- Gesucht: Welche Sequenzen in D sind homolog zu P?
  - Kann nicht beantwortet werden
- Annäherungen
  - Welche Sequenzen in D sind evolutionär sehr ähnlich zu P?
    - Was heißt ähnlich?
  - Welche Sequenzen haben hohen lokalen Alignment-Score zu P?
    - Berechnung Alignmentscore dauert zu lange
  - Näherung: Finde die meisten Sequenzen in D mit hohem lokalen Alignmentscore zu P
    - BLAST
- Was heißt „meisten"?

Sensitivität und Spezifität

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

- Precision = TP/(TP+FP)
  - Wie viele der Treffer des Verfahrens sind wirklich welche?
  - Ähnlich wie Spezifität
- Recall = TP/(TP+FN)
  - Wie viele der echten Treffer findet das Verfahren?
  - Sensitivität

Trade-Off

- Immer eine Balance
  - Algorithmen berechnen typischerweise einen Score pro Sequenz
  - Hoher Score – Positiv; Niedriger Score – Negativ
  - Meist benutzt man einen Schwellwert t, um festzustellen, ab welchem Score man eine Sequenz als positiv ansieht
  - Wenn Score mit Wahrscheinlichkeit für korrekte Klassifikation korreliert, folgt
    - t hoch: Ergebnismenge klein, P=hoch, R=klein
    - t niedrig: Ergebnismenge groß, P=niedrig, R=hoch
- Über t kann man also das Verhältnis von P zu R beeinflussen, aber nicht beide gleichzeitig steigern
  - Dazu braucht man bessere Algorithmen

Beispiel

- Gegeben: Sequenz P und D mit 10.000 Sequenzen
- Algorithmus findet 15 Sequenzen homolog zu P
- In Wahrheit
  - 20 Sequenzen homolog zu P
  - 10 davon hat der Algorithmus gefunden

	Real: Positive	Real: Negative
Alg: Positive	TP = 10	FP = 5
Alg: Negative	FN = 10	TN= 9.975

- Precision = TP/(TP+FP) = 10/15 = 66%
- Recall = TP/(TP+FN) = 10/20 = 50%

BLAST

- Basic Local Alignment Search Tool (BLAST)
  - Heuristischer Suchalgorithmus
  - Datenbanksuche mit lokalem Alignment
  - Schnell, findet aber nicht garantiert die besten lokalen Alignments
- **Die** Erfolgsgeschichte der Bioinformatik
  - Für Biologen lange fast äquivalent zu „Bioinformatik"
  - Eingesetzt auf NCBI/EBI Server
  - Viele Varianten: Proteine/DNA, Reading-Frame, …
  - Software frei erhältlich
- Diverse Weiterentwicklungen
  - Gapped-BLAST, PSI-BLAST, MegaBlast, BLAST-ALL, PATHBLAST, Name-BLAST, …

BLAST Parameter

- Zunächst: Suche in DNA Datenbanken
- Gegeben
  - Suchsequenz P, Datenbank D={S_1,...,S_n}
  - Substitutionsmatrix MA
  - Parameter w: Länge der „Seeds“
  - Parameter t: Initialer Schwellwert
  - Parameter c: Gesamtschwellwert
    - Wird berechnet in Abhängigkeit von t, MA, |DB|, |P|
  - Parameter v: Erwünschte Anzahl Treffer
    - Blast versucht, die v Sequenzen mit bestem lokalen Alignment zu finden

BLAST: Drei Schritte

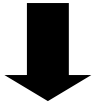
- Bestimme alle Teilwörter $P_1,...,P_m$ der Länge w in P
  - Mit Überlappung – keine Partitionierung
- Suche nach Hits von $P_1,...,P_m$ in D mit Score über t
  - Keine Indels, Verwendung von MA
- Erweitere Hits zu MSPs
  - Verlängere Bereich (ohne Indels) um jeden Hit H in P und S_j bis
    - Sequenz P oder S_j zu Ende ist, oder
    - Alignmentscore fällt unter Schwellwert c, oder
    - Alignmentscore fällt deutlich unter bisherige v beste Treffer
      - „Deutlich“: Geschätzt
  - Ergibt „Maximal Segment Pairs (MSP)“
- Die v besten MSP sind das Ergebnis

Example

w=5, t= 3, Matrix MA: M=+1, R=-1
P = ACGTGATA
S_j= GATTGACGTGACTGCTAGTGATACTATAT

P_1=ACGTG
P_2=CGTGA
P_3=GTGAT
P_4=TGATA

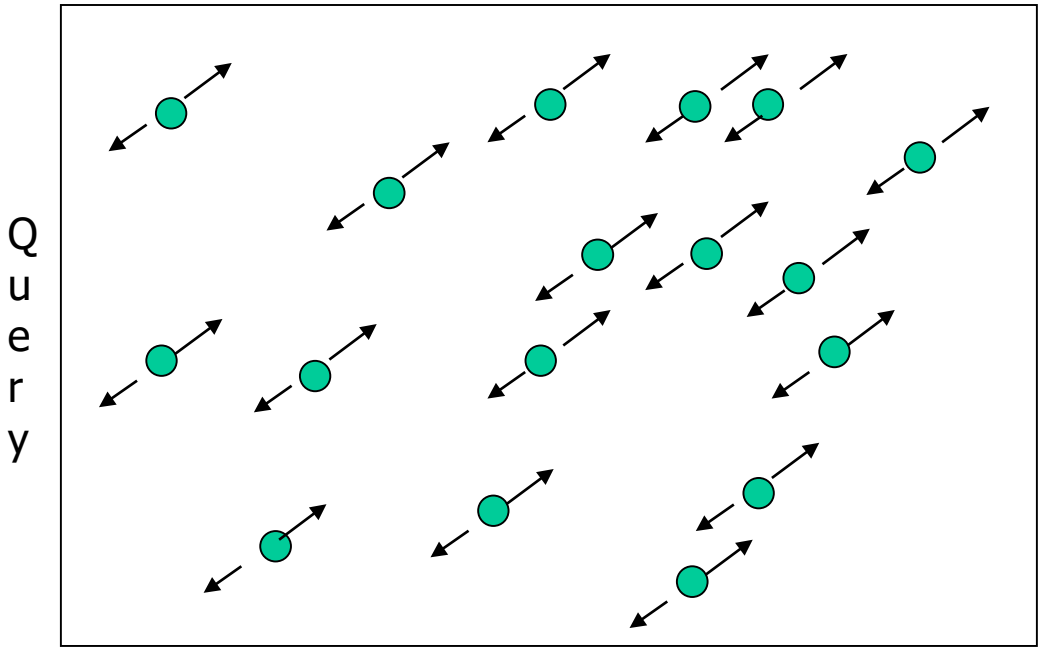
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCTAGTGATACTATAT



GATTGACGTGACTGCAAGTGATACTATAT

ACGTGATA	5
ACGTGATA	5+1=6
ACGTGATA	6-1=5
...	...

Veranschaulichung



- Original BLAST – kein Zusammenfügen mehrerer MSP
- Keine Beachtung von Gaps

BLAST für Proteinsequenzen

- Proteinsequenzen haben größeres Alphabet
- Auch kleinere Scores und kürzere Seeds sind aussichtsreich
- Suche nach „t-guten" Teilwörtern nicht sensitiv genug
  - Beispiel: MASGTLVWG und MTSDTSVRG
  - 50% identisch, aber selbst bei w=2 hätte kein Seed einen guten Hit
- Aber: Wenn w zu klein ist, geht der Filtereffekt verloren
- Stattdessen
  - Gegeben Seeds $P_1,...,P_m$ der Länge w aus P
  - Berechne eine Menge $Q=\{Q_1,...,Q_z\}$ von Teilwörtern der Länge w ($z>>m$)
    - In jedem P_i ersetze jeweils eine Position mit allen Aminosäuren
    - In Q aufnehmen, wenn Score zu P_i größer als t ist

Beispiel

```
w=2
P = qlnfsagw
Sj= ...
```

```
P1=ql
P2=ln
P3=nf
P4=fs
P5=sa
P6=ag
P7=gw
```

Teilwörter	Permutationen
ql:	ql,ml,nl,fl, … qe,ql,qm,qn, …
ln:	ln,mn, … lm,lw, …
nf:	nf, …

Scores (PAM 120, t=3)

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9																			
S	-1	3																		
T	-3	2	4																	
P	-3	1	-1	6																
A	-3	1	1	1	3															
G	-5	1	-1	-2	1	5														
N	-5	1	0	-2	0	0	4													
D	-7	0	-1	-2	0	0	2	5												
E	-7	-1	-2	-1	0	-1	1	3	5											
Q	-7	-2	-2	0	-1	-3	0	1	2	6										
H	-4	-2	-3	-1	-3	-4	2	0	-1	3	7									
R	-4	-1	-2	-1	-3	-4	-1	-3	-3	1	1	6								
K	-7	-1	-1	-2	-2	-3	1	-1	-1	0	-2	2	5							
M	-6	-2	-1	-3	-2	-4	-3	-4	-4	-1	-4	-1	0	8						
I	-3	-2	0	-3	-1	-4	-2	-3	-3	-3	-4	-2	-2	1	6					
L	-7	-4	-3	-3	-3	-5	-4	-5	-4	-2	-3	-4	-4	3	1	5				
V	-2	-2	0	-2	0	-2	-3	-3	-3	-3	-3	-3	-4	1	3	1	5			
F	-6	-3	-4	-5	-4	-5	-4	-7	-6	-6	-2	-4	-6	-1	0	0	-3	8		
Y	-1	-3	-3	-6	-4	-6	-2	-5	-4	-5	-1	-6	-6	-4	-2	-3	-3	4	8	
W	-8	-2	-6	-7	-7	-8	-5	-8	-8	-6	-5	1	-5	-7	-7	-5	-8	-1	-1	12

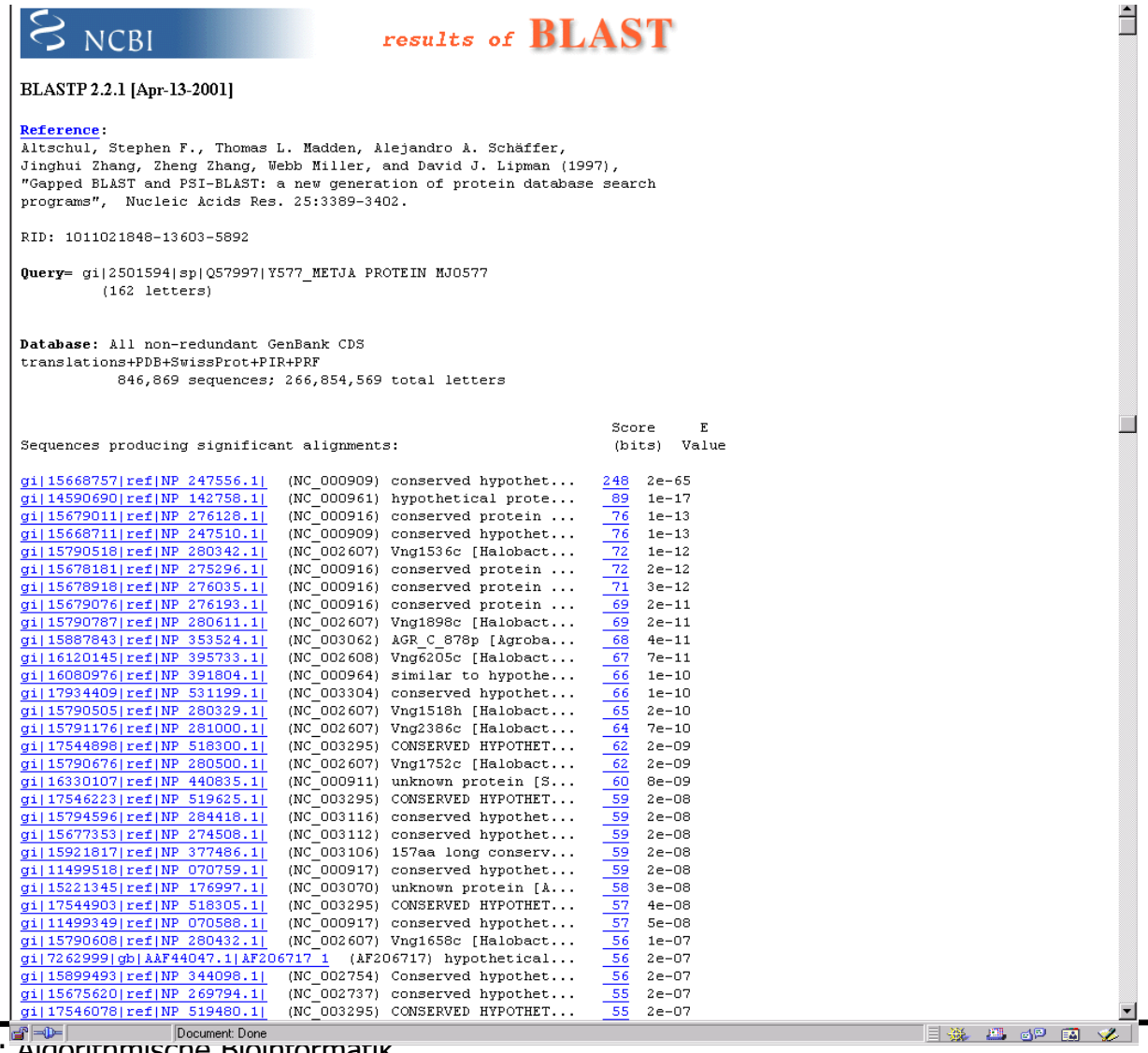
```
ql: 11
cl: -2
sl:  3
tl:  3
ml:  4
il:  2
…
qc: -1
qs:  2
qe:  2
…
```

Quelle: Zvelebil, Baum: ["Understanding Bioinformatics"]

Bemerkungen

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

BLAST Screenshot



NCBI results of BLAST

BLASTP 2.2.1 [Apr-13-2001]

Reference:
Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schäffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs", Nucleic Acids Res. 25:3389-3402.

RID: 1011021848-13603-5892

Query= gi|2501594|sp|Q57997|Y577_METJA PROTEIN MJ0577
(162 letters)

Database: All non-redundant GenBank CDS
translations+PDB+SwissProt+PIR+PRF
846,869 sequences; 266,854,569 total letters

Sequences producing significant alignments:		Score (bits)	E Value
gi|15668757|ref|NP_247556.1|	(NC_000909) conserved hypothet...	248	2e-65
gi|14590690|ref|NP_142758.1|	(NC_000961) hypothetical prote...	89	1e-17
gi|15679011|ref|NP_276128.1|	(NC_000916) conserved protein ...	76	1e-13
gi|15668711|ref|NP_247510.1|	(NC_000909) conserved hypothet...	76	1e-13
gi|15790518|ref|NP_280342.1|	(NC_002607) Vng1536c [Halobact...	72	1e-12
gi|15678181|ref|NP_275296.1|	(NC_000916) conserved protein ...	72	2e-12
gi|15678918|ref|NP_276035.1|	(NC_000916) conserved protein ...	71	3e-12
gi|15679076|ref|NP_276193.1|	(NC_000916) conserved protein ...	69	2e-11
gi|15790787|ref|NP_280611.1|	(NC_002607) Vng1898c [Halobact...	69	2e-11
gi|15887843|ref|NP_353524.1|	(NC_003062) AGR_C_878p [Agroba...	68	4e-11
gi|16120145|ref|NP_395733.1|	(NC_002608) Vng6205c [Halobact...	67	7e-11
gi|16080976|ref|NP_391804.1|	(NC_000964) similar to hypothe...	66	1e-10
gi|17934409|ref|NP_531199.1|	(NC_003304) conserved hypothet...	66	1e-10
gi|15790505|ref|NP_280329.1|	(NC_002607) Vng1518h [Halobact...	65	2e-10
gi|15791176|ref|NP_281000.1|	(NC_002607) Vng2386c [Halobact...	64	7e-10
gi|17544898|ref|NP_518300.1|	(NC_003295) CONSERVED HYPOTHET...	62	2e-09
gi|15790676|ref|NP_280500.1|	(NC_002607) Vng1752c [Halobact...	62	2e-09
gi|16330107|ref|NP_440835.1|	(NC_000911) unknown protein [S...	60	8e-09
gi|17546223|ref|NP_519625.1|	(NC_003295) CONSERVED HYPOTHET...	59	2e-08
gi|15794596|ref|NP_284418.1|	(NC_003116) conserved hypothet...	59	2e-08
gi|15677353|ref|NP_274508.1|	(NC_003112) conserved hypothet...	59	2e-08
gi|15921817|ref|NP_377486.1|	(NC_003106) 157aa long conserv...	59	2e-08
gi|11499518|ref|NP_070759.1|	(NC_000917) conserved hypothet...	59	2e-08
gi|15221345|ref|NP_176997.1|	(NC_003070) unknown protein [A...	58	3e-08
gi|17544903|ref|NP_518305.1|	(NC_003295) CONSERVED HYPOTHET...	57	4e-08
gi|11499349|ref|NP_070588.1|	(NC_000917) conserved hypothet...	57	5e-08
gi|15790608|ref|NP_280432.1|	(NC_002607) Vng1658c [Halobact...	56	1e-07
gi|7262999|gb|AAF44047.1|AF206717_1	(AF206717) hypothetical...	56	2e-07
gi|15899493|ref|NP_344098.1|	(NC_002754) Conserved hypothet...	56	2e-07
gi|15675620|ref|NP_269794.1|	(NC_002737) conserved hypothet...	55	2e-07
gi|17546078|ref|NP_519480.1|	(NC_003295) CONSERVED HYPOTHET...	55	2e-07

Document: Done

Laufzeit

- Relativ schnell
- Wie sucht man viele P_i in der ganzen DB?
  - (Siehe Übungsaufgabe)
  - Hashing der w-Gramme (oder invertiertes File)
  - Berechne alle möglichen Strings der Länge w (Hashkey)
  - Sortiere alle Teilwörter von D der Länge w in Hashbuckets
  - Sehr viele Buckets, Speicherung der Startpositionen auf Platte
  - Finde jedes P_i im Index
- Viele Tricks (wir sind ja schon heuristisch): Häufige Seeds weglassen, spaced Seeds, Seeds mergen, …

Inhalt dieser Vorlesung

- Exklusionsmethode BYP
- BLAST: Basic Local Alignment Search Tool
  - Blast 1
  - Gapped Blast / Blast 2
- BLAT: BLAST-like Alignment Tool

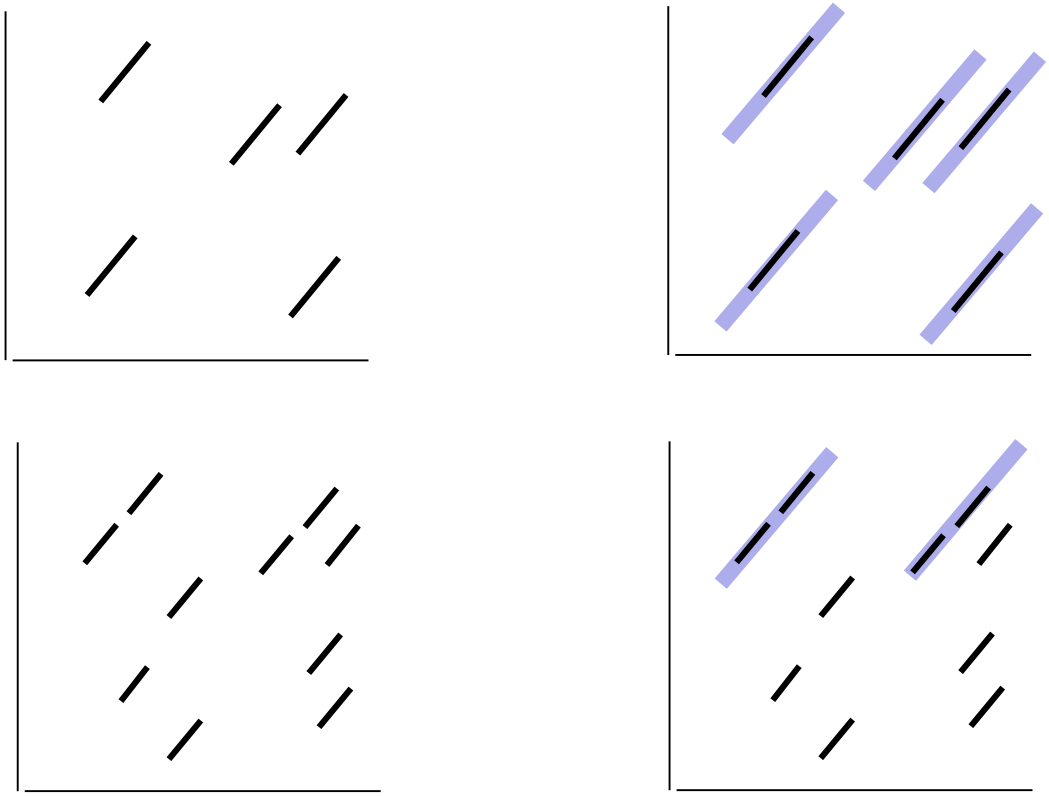
BLAST-2

- 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
    - Sequenzdatenbanken wachsen schneller als Rechnerperformance
  - Gaps beachten
    - Denn: Mehrere kurze Alignments mit Gaps werden von BLAST-1 übersehen, wenn kein Teilmatch einen Score über t schafft
    - Zusammen können diese Alignments aber signifikant sein
    - BLAST-1 liefert bei mehreren kurzen, nahe beieinander liegenden MSP mehrere kurze Ergebnisse (statt einem größeren)

Zwei-Hit-Strategie t

- BLAST-1: Alle Hits mit Score>t werden zu MSPs verlängert
  - Beobachtung: Extensionen fressen >90% der gesamten Laufzeit
  - Aber: Interessante Alignments sind wesentlich länger als ein Seed
- Was könnte man tun?
  - Seeds verlängern – verringert die Sensitivität stark
  - t vergrößern – verringert die Sensitivität stark
- BLAST-2: Zwei Hits verlangen
  - Extension nur, wenn zwei nicht-überlappende Hits auf einer Diagonale mit Abstand <a gefunden wurden
    - Keine INDELs hier: Hits müssen auf einer Diagonale liegen
  - Weniger Extensionen – Performancegewinn
  - Sensitivität behalten – mit kleineren w/t arbeiten
- Performance verdoppelt bei gleichbleibender Sensitivität

Illustration

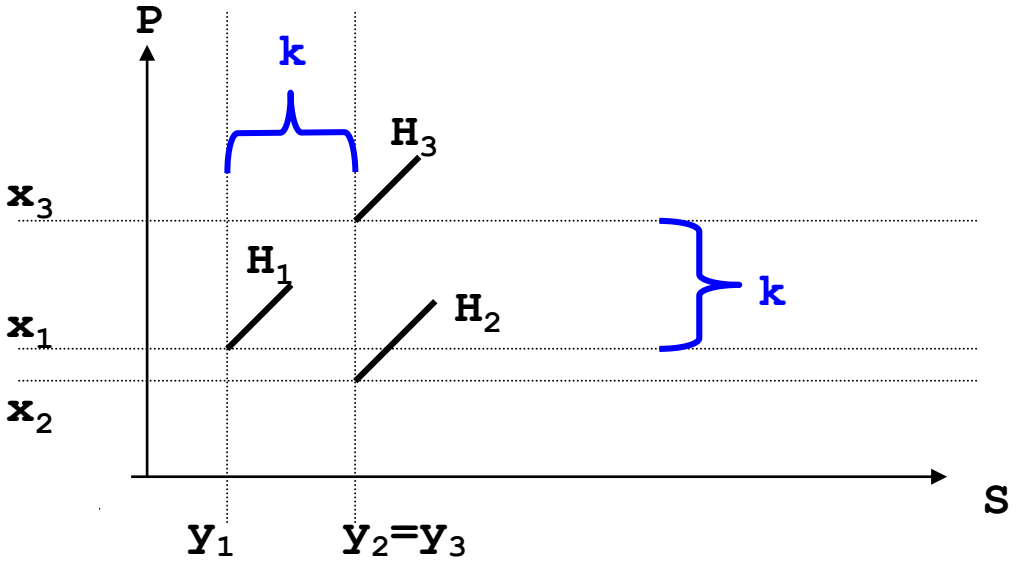


- Woher wissen wir, ob Hits auf derselben Diagonale liegen?

Berechnung der Diagonalen

Diagonale: Subtraktion der Koordinaten

$$\begin{aligned} diag(H_1) &= x_1 - y_1 \\ diag(H_2) &= x_2 - y_2 \\ diag(H_3) &= x_3 - y_3 = \\ &= (x_1+k)-(y_1+k) \\ &= d(H_1) \end{aligned}$$

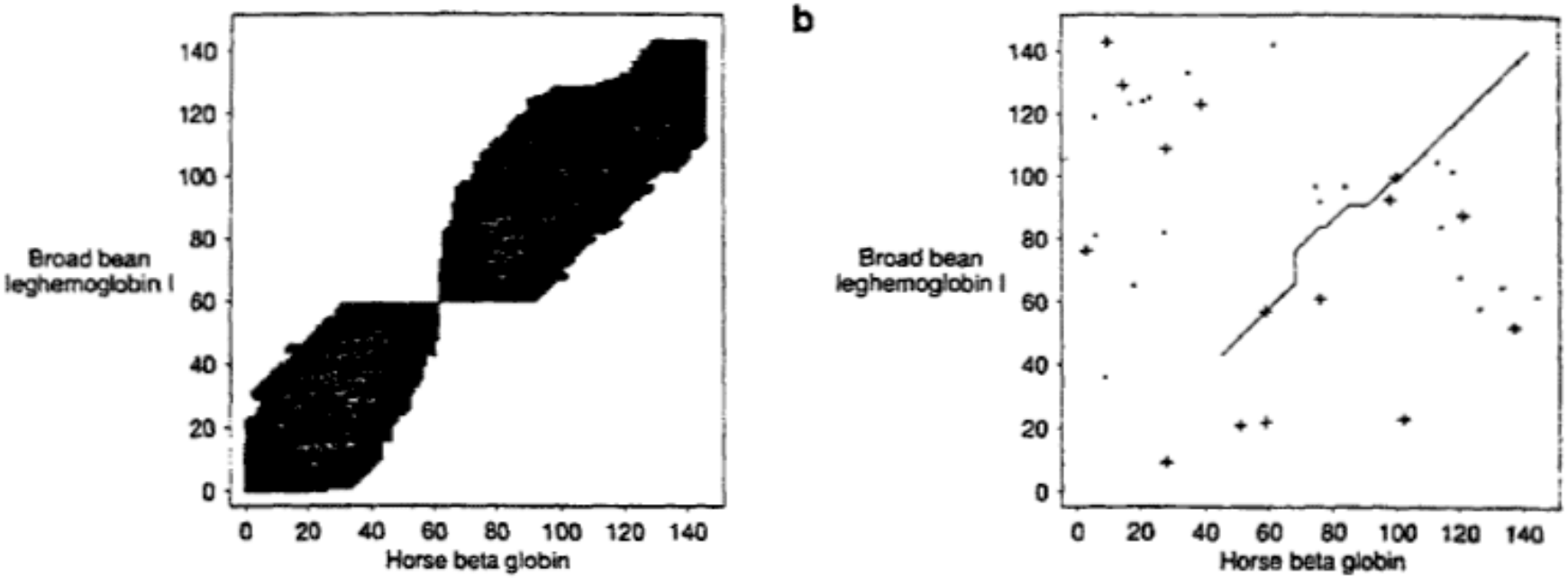


- Datenbank nach Hits durchsuchen
- Pro Hit merkt man sich die Diagonale (und P_i)
- Pro Diagonale merken wir uns das Ende des letzten Hits
- Wenn ein neuer Hit folgt, testen wir auf Abstand a
- Paarbildung erfolgt während der Hitsuche

Gaps in BLAST-2

- BLAST-1: Hits werden verlängert ohne Gaps
  - Schlechte Sensitivität, schon ein Indel zerstört den Match
- BLAST-2: Gapped Alignment
  - Wenn zwei Hits H_1, H_2 in Abstand a gefunden werden, wird ein Smith-Waterman Alignment berechnet
  - Dazu sucht man das Sequenzstück zwischen H_1 und H_2 der Länge w mit dem höchsten Score ohne Gaps
  - Von diesem „Seedpoint" wird lokal in zwei Richtungen aligniert
  - Da SW sensitiver ist als Extensionen ohne Gaps, kann man t wieder erhöhen (musste man wegen der zwei-Hit Strategie verringern)
- Performanceverbesserung trotz besserer Sensitivität
  - Beispiel: 500x langsamere Extension durch SW, aber 4000x weniger Extensionen durch Erhöhung von t von 11 auf 13

Zweiseitiges Smith-Waterman

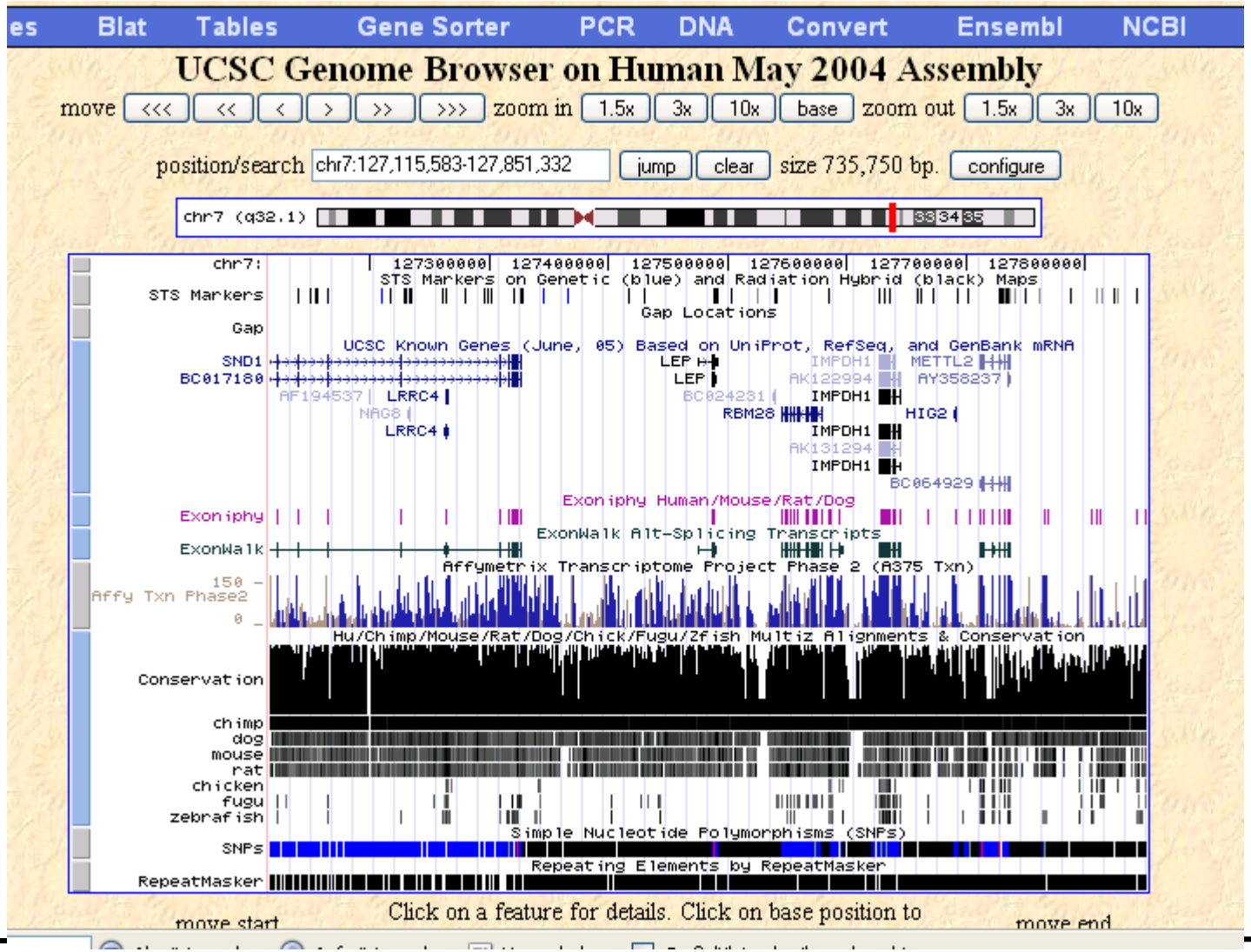


- SW ausgehend vom Seed-Point in beide Richtungen
- Abbruch bei Unterschreiten geschätzter Schranken
  - Wenn das Alignment zu schlecht wird, um es noch in die finale Treffermenge zu schaffen

Inhalt dieser Vorlesung

- Exklusionsmethode BYP
- BLAST: Basic Local Alignment Search Tool
- BLAT: BLAST-like Alignment Tool
  - Kent, W. J. (2002). "BLAT - the BLAST-like alignment tool." *Genome Res* 12(4): 656-64.
  - BLAST für spezielle Anwendungen mit etwas Statistik
  - Viel schneller als BLAST, aber findet nur hoch konservierte Sequenzbereiche

Genome Browser

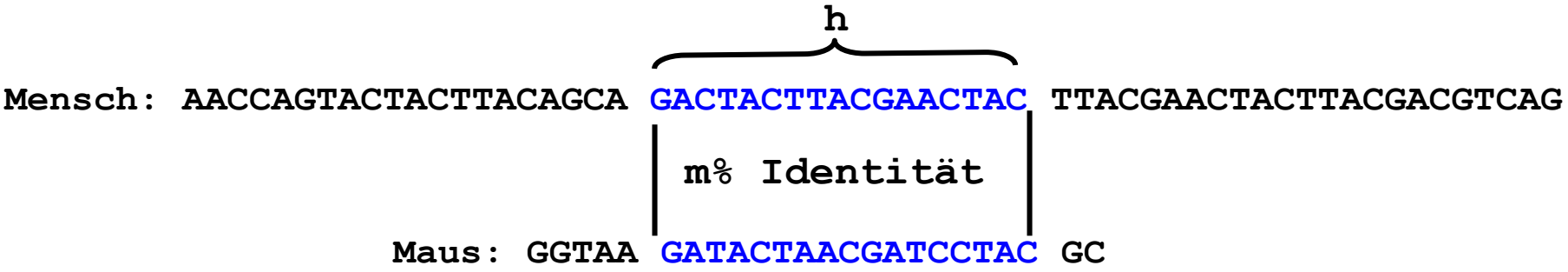


Motivation

- Warum noch schnellere Alignments?
  - „Viele gegen Viele" statt „Einer gegen Viele"
  - Transcript Mapping: $2*10^9$ humane EST-Basen gegen Genom
  - Comparative Genomics: $1.7*10^6$ Maus-Reads gegen Genom
- BLAT: 500-mal schneller als BLAST und genauso sensitiv in hoch konservierten Bereichen
  - Kann und will nur hoch konservierte Sequenzbereiche finden
  - Erlaubt höhere Anforderungen an Seeds (Länge, Matchqualität) und Gaps (Zahl, Länge, Abstand)

Szenario

- Vergleich einer Maus-cDNA P mit einer humanen cDNA D
- Hintergrundwissen über Menschen und Mäuse
  - Wenn eine Sequenz der Maus homolog zu einer Sequenz im Menschen ist, werden diese zu ~m% identisch sein und eine durchschnittliche Länge von h haben
  - Frameshifts (INDELs) sehr unwahrscheinlich



Keine Gaps

- Feste Seedlänge k
- BLAT vergleicht alle überlappenden k-mere von P mit allen nicht-überlappenden k-meren in jeder DB Sequenz
- Gesucht wird ohne Gaps
  - Gut geeignet für cDNA / Genom Mapping
    - Lange Gaps führen zu disjunkten Treffer-Bereichen
    - Kurze Gaps sind unwahrscheinlich - Frameshifts
  - Ungapped-Alignment ist viel schneller als Gapped-Alignment
- Wie lang müssen Seeds sein, damit man mit sehr hoher Wsk mindestens einen exakten Treffer in D findet, wenn P und D homolog sind?
  - Einer reicht – die Extension findet den ganzen Match

Etwas Statistik

m: %-Identität der zwei Sequenzen
h: Durchschnittliche Länge homologer Regionen
q: Länge der Querysequenz P
g: Größe der Datenbank (in Basen)
a: Größe des Alphabets (DNA oder Protein)

- Wähle k so, dass mit hoher Wsk kein gutes lokales Alignment übersehen wird
- Berechnung
  - Sei z~h/k: Zahl nicht-überlappender k-Mere in einer homologen Region
  - Wsk, dass ein beliebiges k-mer aus P mit seinem Gegenstück in D perfekt matched
    - $p_1=m^k$
  - Aber: Wir nehmen aus D ja nicht-überlappende k-mere; das Gegenstück zu einem beliebigen P muss nicht dabei sein
  - Wsk, dass mindestens ein nicht-überlappendes k-mer aus D mit dem entsprechenden k-Mer in P perfekt matched (wenn P, D 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

- Wahrscheinlichkeiten, dass mindestens ein perfekter Match in der Region vorkommt
  - q=100, m und k variabel (a und g hier nicht notwendig)
  - Ein Match reicht – Extensionsphase wird die ganze Region finden
- Gesprochen: Bei erwarteter Sequenzähnlichkeit von m=97% findet man in einer homologen Region in D der Länge h=100 Basen praktisch immer mindestens einen perfekten Match der Länge 13 mit einem Substring von P

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
  - Hohes k: Wenig falsch-positive, aber eventuell fehlende echte Hits
  - Niedriges k: Viele falsch-positive, aber weniger falsch-negative

Variante – Hits mit Mismatches

- Suche nach Hits mit höchstens einem Mismatch
  - Wahrscheinlichkeit, dass ein gegebenes k-mer in einer homologen Region 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

Variante – Mehrere Hits

- Was, wenn wir n (perfekte) Hits in der homologen Region verlangen
  - Wie BLAST-2; aber die Länge des Abstands zw. den Hits ist beliebig
- Berechnung
  - Wsk, dass ein beliebiges k-mer aus der Suchsequenz mit seinem Gegenstück in der homologen Datenbanksequenz perfekt matched
    - $p_1 = m^k$
  - Wsk, dass man beim Vergleich genau n perfekte Hits der Länge k findet

$$p_n = (p_1)^n * (1 - p_1)^{z-n} * \binom{z}{n}$$

Suchvariante 2 – mehrere Hits

- Wsk, dass man n oder mehr perfekte Hits findet
  - $p = p_n + p_{n+1} + \ldots p_z$
- Wsk für falsch-positive Matches berechnet sich wie vorher
- Ergebnis: Drastische Verringerung der erwarteten Anzahl falsch-positiver Hits
  - Intuition: Es müssen ja zusammen mehr Basen matchen; aber nicht alle hintereinander

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

Zusammenfassung

- Biologische Anwendungen operieren auf sehr grossen Datenmengen und brauchen sehr effiziente Algorithmen
- Eine sehr wichtige Anwendung ist approximative Suche in Sequenzdatenbanken
- Alle dafür ausreichend schnellen Verfahren sind Heuristiken
  - Tauschen Suchgenauigkeit für Laufzeit
- BLAST war lange Standard
  - Schwerpunkt auf hoher Sensitivität
  - Trade-Offs zwischen t/w Werten, Geschwindigkeit und Sensitivität
- BLAT ist besser für Suche in hochkonservierten Genomen
- Next-Generation Sequencing: Wir müssen noch schneller werden

Selbsttest

- Prinzip der Exklusionsmethoden
- Wie erreicht BYP lineare Laufzeit im Average Case?
- Wie realistisch ist dieser Average Case bei der Genomsuche?
- Wie funktioniert BLAST und BLAST-2
- Warum ist Proteinsuche in BLAST anders implementiert als DNA Suche?
- Wie man in BLAST t senkt – welche Auswirkungen hat das auf Spezifität, Sensitivität und Geschwindigkeit?
- Leiten Sie die Formel in BLAT für die Wsk eines Hit mit genau einem Mismatch her