

Bioinformatik

BLAT
QUASAR

Ulf Leser

Wissensmanagement in der
Bioinformatik



Exklusionsmethode BYP

- Alignment zweier Strings A, B dauert $O(n \cdot m)$
- K-Band Algorithmus benötigt $O(sn^2 - vn)$ für $|A| = |B|$
 - Gutes Verfahren, wenn nur sehr ähnliche Sequenzen interessant sind
- Anderes Vorgehen: Wir lassen nur eine feste Zahl k **Unterschiede** (Mismatch oder Leerzeichen) zu
- Definition
 - *Ein **k -Difference Vorkommen** von P in T ist ein Substring T' von T , der ein Alignment mit P mit Abstand höchstens k hat*
- Gesucht: Algorithmen, die alle **k -Difference Vorkommen** von P in T für geeignete k in **$O(m)$ Laufzeit im Average Case** finden

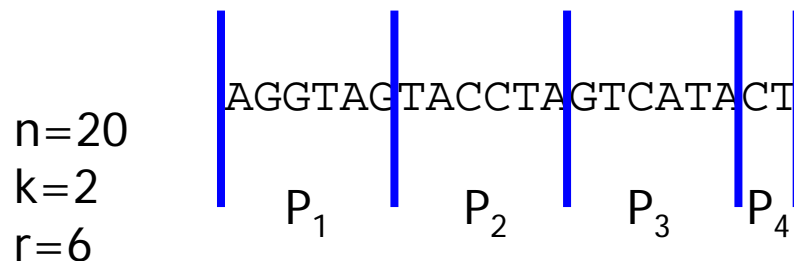
Exklusionsmethoden

- Es gibt eine ganze Reihe Algorithmen, die das erreichen
- Wir sehen uns näher an
 - BYP: Baeza-Yates, Perleberg: „Fast and practical approximative string matching“, LNCS 664, 1992
- Grundaufbau
 - Wir suchen k-Difference Vorkommen von P in T
 - **1. Partition**: Partitioniere P geschickt
 - **2. Search**: Suche Partitionen in T mit einem exakten Algorithmus
 - Partitionierung ist so gewählt, dass jedes potentielle k-Difference Vorkommen von P in T um die Fundstelle einer Partition liegen muss
 - Aber: Nicht jede Fundstelle ist Kern eines k-Difference Vorkommens
 - **3. Check**: Überprüfe alle Vorkommen von Partitionen von P in T
- Warum sparen wir?
 - Im **Average Case** müssen nur wenige Bereiche von T mit (teurer) dynamischer Programmierung untersucht werden
 - Die anderen Bereiche überspringt man bei der (linearen) Search-Phase

1. BYP - Partition

- Partition

- Zerlege P gleichmäßig in Teilstrings der Länge $r = \text{floor}(n/(k+1))$
- Damit haben wir $k+1$ Substrings von P der Länge r und einen kürzeren
 - Das sei der letzte



BYP – Komplexität

- Theorem

*Für Strings P und T mit $|P|=n$ und $|T|=m$ und $k \sim n/\log_s(n)$ findet der BYP Algorithmus im **Average Case** alle k -Difference Vorkommen von P in T in $O(m)$*

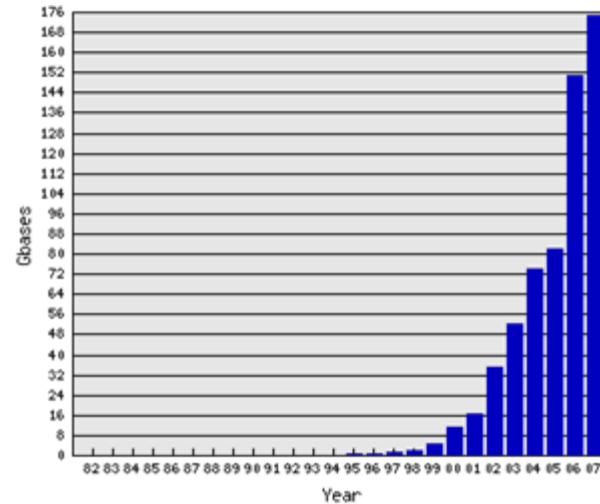
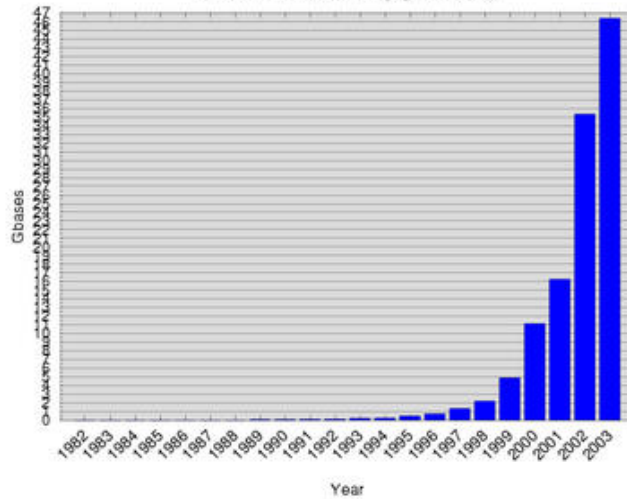
- Beweis

- Sei p eine der $k+1$ Partitionen von P der Länge r
 - T enthält 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 $(k+1) \cdot m/s^r$ Vorkommen einer Partition in T
- Für jede brauchen wir $O(n^2)$; die Gesamtlauzeit wollen wir aber in $O(cm)$ halten (c : beliebige Konstante). Also

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

Heuristische Alignierung

EMBL Database Growth
total nucleotides (gigabases)



- Annotation neuer Sequenzen basiert auf Suche nach homologen Sequenzen in [Sequenzdatenbanken](#)
- Datenmenge wächst **exponentiell** – selbst lineare Algorithmen sind zu langsam
- Gesucht sind schnelle Verfahren
 - Auch wenn wir dabei ein paar (schlechte?) Ergebnisse verlieren

BLAST Schritt 1 und 2

- Schritt 1
 - Bestimme alle **Teilwörter** P_1, \dots, P_m der Länge w in P
 - Mit Überlappung – keine Partitionierung
- Schritt 2
 - Suche nach **Hits** von P_1, \dots, P_m in DB mit Score über t
 - Hits müssen nicht exakt sein
 - Vergleiche alle P_i mit allen Teilwörtern der Länge w in T
 - Score wird ohne Leerzeichen mit Werten aus M berechnet
 - Durch t
 - werden auch Hits gefunden, die keine perfekten Matches sind
 - werden **statistisch insignifikante** Hits von vorneherein ausgeschlossen

BLAST Schritt 3

- Schritt 3
 - Für jeden Hit H zwischen DB-Sequenz S_j und P_i
 - **Verlängere Bereich** um H sowohl in P als auch in S_j
 - Alignment (gapfree) erst nach links, dann nach rechts wachsen lassen
 - Solange, bis
 - Sequenz P oder S_j ist zu Ende, oder
 - Alignmentsscore fällt unter geschätzten Schwellwert c , oder
 - Alignmentsscore fällt „signifikant“ unter bisherige beste Treffer
 - » „Signifikant“ heuristisch bestimmt, abhängig von c und v
 - Ergibt „**Maximal Segment Pairs (MSP)**“
 - Die besten v MSP sind das Ergebnis

Bemerkungen

- Proteinsuche
 - Expansion der Suchstrings – stärkeres Kriterium als $w=1$, aber schwächer als $w=2$
- BLAST ist eine **Exklusionsmethode**
 - Bestimme und suche nach Seeds = minimale Alignments, die (höchstwahrscheinlich) im Kern jedes optimalen Alignments stecken
 - Erweitere nur diese zu vollen Alignments
 - **Praktisch alle Alignmentheuristiken arbeiten nach diesem Muster**
 - Quasar, Biohunter, BLAT, FASTA, ...
- Heuristik
 - BLAST **kann nicht alle optimalen Alignments** finden
 - Keine Inserts/Gaps, Schwellwerte w und t

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?
- Oftmals **eine Balance**
 - Algorithmen berechnen einen Score pro Sequenz
 - Hoher 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



Laufzeit

- Keine exakte Komplexität bekannt
- In der Regel sublinear
- Wie sucht man viele P_i in sehr vielen S ?
 - Hashing
 - Für festes w
 - Berechne alle **möglichen Strings** der Länge w (Hashkey)
 - Sortiere alle Teilwörter von S der Länge w in Hashbuckets
 - Sehr viele Buckets, Speicherung der Startpositionen auf Platte
 - Für ein P_i
 - Berechne alle **Teilwörter der Länge w mit Score über t** für P_i
 - Für Proteine: Wie beschrieben; für DNA: auch mehrere Mutationen in einem Teilwort zulassen
 - Startpositionen findet man in den Hashbuckets
- Für „normal-seltene“ Teilwörter (keine Repeats etc.) ist das extrem schnell

Zwei-Hit-Strategie

- Original: Alle Hits mit $\text{Score} > t$ werden zu MSPs verlängert
 - Beobachtung: Extensionen fressen $>90\%$ der gesamten Laufzeit
 - Aber: Interessante Alignments sind wesentlich länger als ein Seed
 - Man sollte also mehr verlangen als nur ein (kurzes) Seed, bevor man die Extension beginnt
 - Aber: Lange Seeds verringern die Sensitivität
- Neue Strategie
 - Extension erfolgt nur, wenn **zwei nicht-überlappende Hits auf einer Diagonale** mit Abstand $< a$ gefunden wurden
 - Neuer Parameter: a
 - Dadurch werden weniger Extensionen ausgeführt – großer Performancegewinn
 - Andererseits sinkt Sensitivität – deswegen mit kleineren t arbeiten
- Ergebnis
 - Performance verdoppelt bei gleichbleibender Sensitivität

Gaps in BLAST-2

- Original: Hits werden verlängert ohne Gaps
- BLAST-2: Gapped Alignment
 - Wenn zwei Hits H_1 , H_2 in Abstand a gefunden werden, wird in dieser Diagonale 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“ aus lokales Alignment berechnen
 - Da **SW sensitiver ist als Extensionen ohne Gaps**, kann man t wieder erhöhen (musste man wegen der zwei-Hits Strategie verringern)
- Weitere **Performanceverbesserung** trotz besserer Sensitivität
 - 500x langsamere Extension durch SW, aber 4000x weniger Extensionen durch Erhöhung von t von 11 auf 13

Inhalt dieser Vorlesung

- BLAT
 - Kent, W. J. (2002). "BLAT - the BLAST-like alignment tool." *Genome Res* **12**(4): 656-64.
 - Viel schneller als BLAST, aber
 - nur bei sehr ähnlichen Alignments möglich
- QUASAR
 - Suffixarrays und Alignments

Genome Browser 1

Genomes Blat Tables Gene Sorter PCR DNA Convert Ensembl NCBI PDF/P

UCSC Genome Browser on Human May 2004 Assembly

move <<< << < > >> >>> zoom in 1.5x 3x 10x base zoom out 1.5x 3x 10x

position/search chr7:127,115,583-127,851,332 jump clear size 735,750 bp. configure

chr7 (q32.1)

chr7: 127300000 | 127400000 | 127500000 | 127600000 | 127700000 | 127800000

STS Markers on Genetic (blue) and Radiation Hybrid (black) Maps

Gap Locations

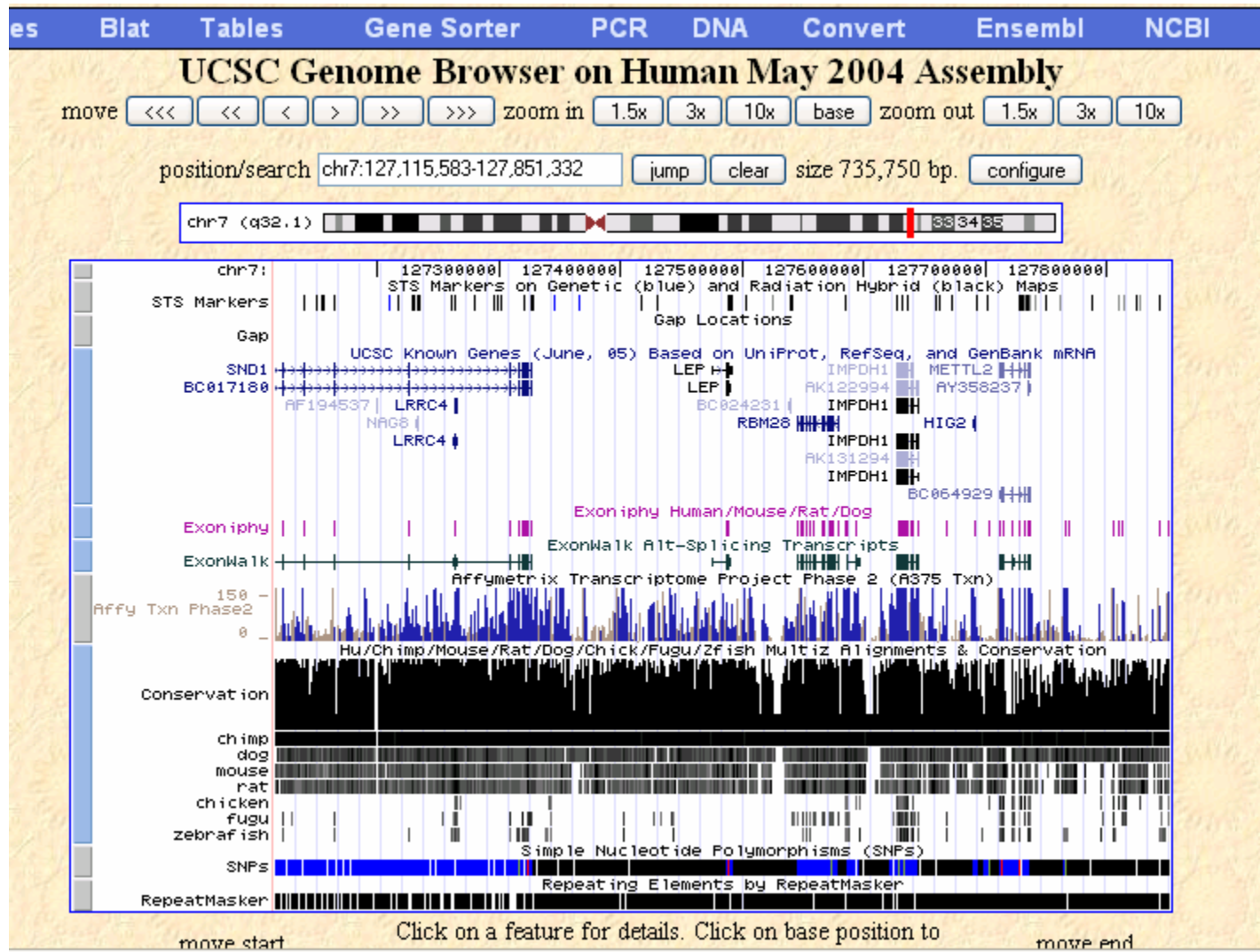
UCSC Known Genes (June, 05) Based on UniProt, RefSeq, and GenBank mRNA

RefSeq Genes

ExonWalk

Human mRNAs from GenBank

Screenshot 2

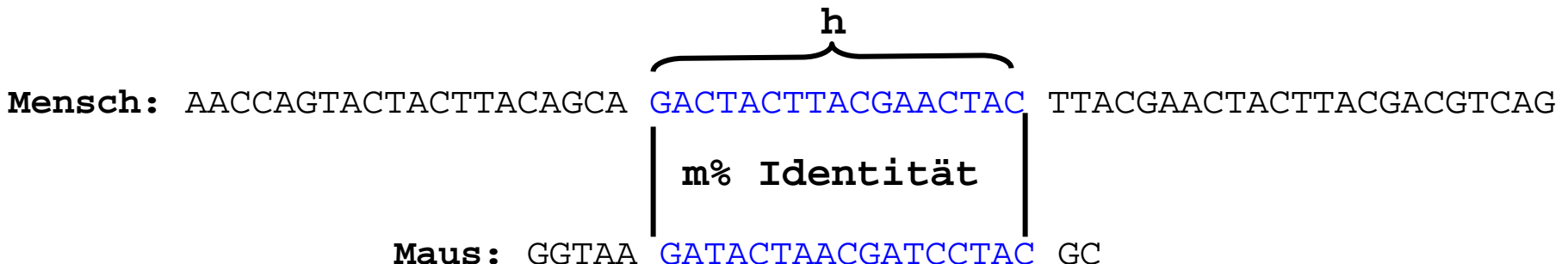


Motivation

- Warum sehr schnelle Alignments? Viel gegen Viel
 - Transcript Mapping
 - $2 \cdot 10^9$ humane EST-Basen gegen das humane Genom ($2.9 \cdot 10^9$ Basen)
 - Comparative Genomics
 - $1.7 \cdot 10^6$ Maus-Reads gegen das humane Genom
 - Alle Maus-Human ESTs miteinander verglichen
- 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**
 - Kann nur hoch konservierte Sequenzen finden
 - Das erlaubt deutlich höhere Anforderungen an Seeds und Gaps

BLAT Szenario

- Vergleich einer Maus-cDNA Q mit einer humanen cDNA P
- Hintergrundwissen über Menschen und Mäuse
 - Wenn es eine homologe Sequenz in der Maus zu einer Sequenz im Menschen gibt, wird diese 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 mit P finden**, wenn Q und P homolog sind?



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 -meren in jeder 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
 - Später

BLAT – Statistik

- Ziel: **Maximales k so, dass sehr wahrscheinlich nichts verloren geht**
 - 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 einer 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
 - Voraussetzung: $q > h$
- 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 mit einem Substring von Q

Falsch-positive Treffer

- 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

- Suche nach Hits mit **höchstens einem Mismatch**
 - Wahrscheinlichkeit, dass ein gegebenes k-mer in einer homologen Region perfekt oder mit einem Mismatch matched
 - $P_1 = k \cdot m^{k-1} \cdot (1-m) + m^k$
 - 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

Suchvariante 2 – mehrere Hits

- Was, wenn wir **n (perfekte) Hits** verlangen?
 - Siehe BLAST-2: Verlangt 2 Hits
- 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 man beim Vergleich **genau n perfekte** Matches (der Länge k) findet

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

Suchvariante 2 – mehrere Hits

- Wahrscheinlichkeit, dass man beim Vergleich **n oder mehr perfekte Matches** (der Länge k) findet
 - $p = p_n + p_{n+1} + \dots + p_z$
- Vorteil: 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

Indexierung und Suche

- Alle nicht-überlappende k-mere der DB werden indiziert
 - Entfernung zu häufiger k-mere (Repeats)
 - **Hashing**
 - Achtung: $k=15, a=4$: $4^{15} \sim 10^9$ potentielle Indexpositionen
- Suche mit allen überlappenden k-meren der Query
 - Speichern aller Treffer-Paare mit ihrer Diagonale
 - Abschließendes Sortieren aller Treffer nach Diagonale
 - Bestimmung von „**Proto-Clumps**“
 - Mindestens n Hits innerhalb Abstand w
 - Hinzufügung weiterer Basen Up- und Downstream
 - Zahl ergibt sich aus Durchschnittslängen von Exons
- **Suche mit Mismatch**
 - Expansion der Such-k-mere in alle Varianten mit einem Mismatch
 - Beispiel: $k=14, a=4 \Rightarrow$ Suche mit $14 \cdot (4-1) + 1 = 43$ Varianten

Zwischenstand

- Wo sind wir
 - Wir haben Proto-Clumps gefunden
 - Kern davon sind einer oder mehrere (fast) perfekte Hits
 - Diese Hits liegen auf einer Diagonalen und haben maximal Abstand w
 - Um die Hits herum wurde noch einiges an Sequenz hinzugefügt
- Was fehlt?
 - **Alignments** der Query mit den Sequenzen, in denen die Proto-Clumps gefunden wurden

Alignmentphase in BLAT

- Heuristisches, schnelles Verfahren
 - Ziel: Alignment für kompletten Proto-Clump finden
 - Suche nach **maximalen perfekten Hits** mit kleinen Seeds
 - Wir suchen jetzt nur im Clump, nicht mehr in der Datenbank
 - Zusammensetzen dieser zu komplettem Alignment
 - Gaps werden **rekursiv mit kleineren Seeds** gefüllt
- Findet und verklebt Bereiche perfekter Hits bis minimale Länge des kleinsten Seeds (5)
- Kein echtes Alignment, aber schnell
- Spezielle Behandlung von EXON-INTRON Signalen
- Nur gut für **sehr ähnliche Sequenzen** (>90%)

Quasar

- Burkhardt, S., Crauser, A., Ferragina, P., Lenhof, H.-P., Rivals, E. and Vingron, M. (1999). "q-gram Based Database Searching Using a Suffix Array (QUASAR)". RECOMB 1999.
 - BLAST ist zu langsam
 - Quasar ist auch für Sequenzen mit hoher Ähnlichkeit gemacht (wie BLAT)
 - Benutzt BLAST in der Alignment / Extension-Phase, aber hat **schnellere Methode zum Finden der Hits**

Quasar Grundidee

- Erste Phase sucht Regionen mit Länge w und hoher Ähnlichkeit
 - D.h. Regionen mit Editabstand $< k$
- q -Gramme sind Substrings der Länge q
 - q -Gramme = q -mere
- q -Gramm Lemma
 - Gegeben zwei Sequenzen X, Y der Länge w mit Editabstand $< k$. Dann haben X und Y mindestens $t = (w + 1) - (k + 1) * q$ q -Gramme gemeinsam.
- Beweis
 - X und Y besitzen jeweils $(w - q + 1)$ q -Gramme der Länge q
 - Wenn in einem Substring Z von X ein Mismatch mit Y auftaucht, dann ist dieser Mismatch in q q -Grammen enthalten
 - X enthält höchstens k Mismatches mit Y
 - Diese können höchstens $k * q$ q -Gramm-Matches „zerstören“
 - Damit bleiben mindestens $t = (w - q + 1) - k * q = (w + 1) - (k + 1) * q$ perfekt matchende q -Gramme übrig
 - qed.

Beispiel

ACACAGGT
ACAGAGGT

- Editabstand = $k = 1$
- Also müssen mindestens $t = (w + 1) - (k + 1) * q$ q -Gramme identisch sein
 - $w = 8; k = 1$
 - $q = 3: t = 3: ACA, AGG, GGT$
 - $q = 4: t = 1: AGGT$
 - $q = 5: t < 0$

Algorithmus

- Quasar verbessert nur die Suchphase von BLAST
 - Gegeben: Gewünschte Seedlänge w , Seedgüte k
 - Seedgüte benutzt hier keine Substitutionsmatrix
 - Berechne t
 - Schiebe ein **Window W der Länge w** über jede Datenbanksequenz und die Query Q
 - Zähle identische q -Gramme
 - Wenn W mehr als t q -Gramme enthält, ist W ein Hit für Blast
 - Diese werden BLAST zur weiteren Verarbeitung übergeben

Tricks & Komplexität

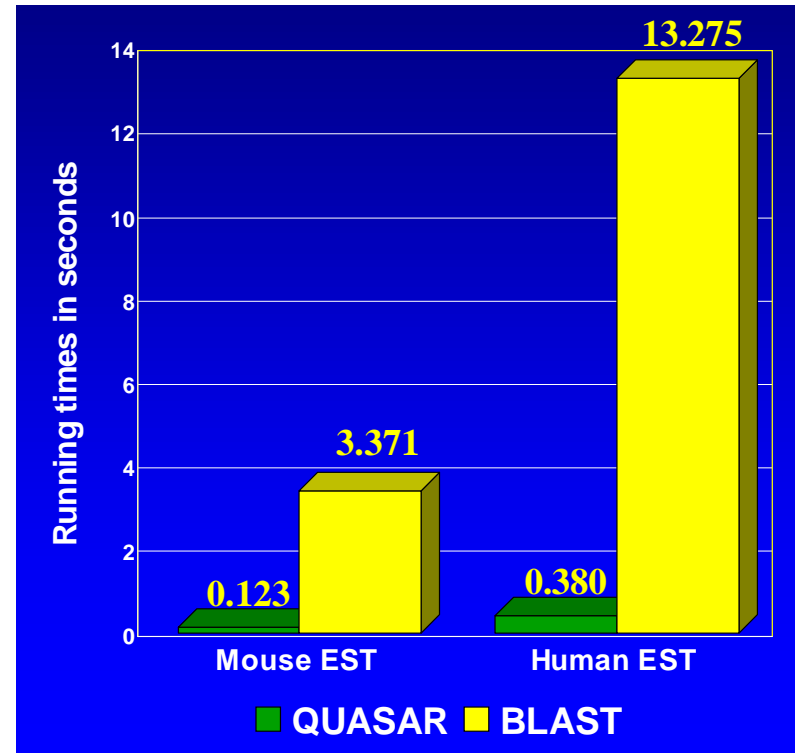
- Beachte: Wir suchen **exakte Hits**
- **Aufbau eines Suffixarrays** für alle Sequenzen in DB
 - Zeit: $O(|DB|)$
 - Nachteil: Platzbedarf $\sim 9 * |DB|$ laut Paper
 - Es gibt $|DB|$ Suffixe; wir brauchen jeweils einen Pointer in die Datenbank
- Aus diesem Array einen **Hashindex** für alle q-Gramme der DB erzeugen
 - Suffixarray linear scannen ($O(|DB|)$)
 - Alle Suffixe mit identischem Präfix der Länge q zu einem Hashbucket zusammenfassen
 - Inhalt: (q-Gram, ((seqid₁, pos₁), (seqid₂, pos₂), ...))
 - Test kostet $O(q)$ – man muss nur mit dem Vorgänger vergleichen
 - Trick: Vergleiche von Hinten anfangen
 - Zusammen: $O(|DB| * q)$

Suche

- Wir vergleichen nun nicht Q mit jeder Sequenz, sondern suchen **alle q -Gramme von Q nacheinander im Index**
 - Jedes Q -Gram ergibt eine Liste von Sequenz/Position Paaren
 - Wir sammeln alle Treffer zu einer Sequenz in einer nach Position sortierten Liste
 - Sequenzen mit weniger als T Einträgen sofort löschen
 - Schließlich wird über jede Sequenzliste iteriert und nachgesehen, ob innerhalb eines Intervalls der Länge W genügend Treffen vorlagen
- Suche nach allen Vorkommen eines q -Gramms ist konstant
- Suchphase zusammen: $O(|Q| * v)$
 - Sei v die maximale Anzahl der Vorkommen eines q -Grams in DB
 - Wir ignorieren den Aufwand für das Sortieren der Listen

Ergebnisse

- Parameter
 - 6% Error
 - $w = 50$, $q = 11$
 - Scan with BLAST
 - Averaged for 1000 queries
- ~30 Mal schneller als BLAST
 - Bei etwa gleichbleibender Sensitivität
 - Klappt nur für sehr ähnliche Sequenzen (k)
 - Nur schnell, wenn der q -Gramm Index komplett in Hauptspeicher passt



Weitere Suchheuristiken

- Locality-sensitive Hashing (Buhler, 2001)
 - Effiziente Suche nach k-meren mit höchstens n Fehlern
- Oasis (VLDB, 2003)
 - Smith-Watermann kombiniert mit A^* -Algorithmus
 - Als nächstes zu expandierende Zellen werden mit A^* Algorithmus gewählt
- Patternhunter, Biohunter (Bioinformatics, 2002)
 - Kommerzielle Software
 - Sucht nicht nach exakten Seeds der Länge k , sondern nach non-consecutive Areas der Länge w mit mindestens k Hits
 - Schneller als BLAST, bessere Sensitivität
- ... Ein sehr aktives Gebiet