Teilstring ═══ Teilsequenz - - - -

overlap ═══

String-Matching Problem: Text/Muster gg → beide Mustervorkommen

naive Alg. $O(m \cdot (n-m))$: $p = a^m$, $t = a^n$

String Matching Automat: $O(n)$ [Berechnung $O(|\Sigma| \cdot m^3)$ naiv / $O(|\Sigma| \cdot m)$ opt.]

Boyer-Moore Alg: Bad-Char-Regel: Funktion für alle $a \in \Sigma$ letzte Position. Von hinten matchen, verschieben lt. Pos.

opt.: $O(n/m)$
worst: $O(n \cdot m)$

Good-Suf-Regel:

einfacher Suffix-Baum $t \to t\$$: jedes Suffix von $t\$$ ab Wurzel eintragen

String-Suche: Von Wurzel Pfad mit Muster suchen, ab da alle Blätter ≙ Vorkommen

Größe: $O(|t|^2 \cdot |\Sigma|)$, z.B. $t = a^n b^n \$$

kompakter Suffix Baum: jeder innere Knoten mind. 2 Kinder, Kanten mit Strings oder Anfangs-/Endposition im Text
einfacher ⇒ Knoten mit einem Nachfolger eliminieren ⇒ Kompression
$\Rightarrow O(n \cdot \log n)$

Suche: $O(n \cdot \log n + m \cdot |\Sigma| + k)$, $k \triangleq$ overall Vorkommen pint, $|\Sigma| < \frac{n}{c}$, $c > 2$.

verallgemeinerter Suffix Baum: für n Texte mit Textsymbolen $\$_1 \dots \$_n \notin \Sigma$

Longest-Common-Substring: verallgemeinerter Suffix-Baum, mit Menge Text. an jedem inneren Knoten
⇒ maximale Tiefe finden dessen Knoten Blatt Indizes enthält
$O(n \cdot (\log n + N^2 + N \cdot |\Sigma|))$

pearwise Overlaps: verallgemeinerter Suffix-Baum, innere Knoten mit Menge Index inzidenter Kanten
$O(n \cdot (\log n + |\Sigma| + N))$

Alignments: Einfügung / Deletion / Match / Mismatch: Bewertungsfunktion (Match / Mismatch) g: Lücke für Lückensymbol, gap (min, max)
Edit-Distanz ($a/b = 1$, $a/a = 0$, $g = 1$, min), ($a/a = 1$, $a/b = -1$, $g = 2$, max)

global: dyn. Programmierung mit Ähnlichkeitsmatrix $O(n \cdot m)$
graphisch: Edit-Graph, längsten Weg finden (polynomiell)

lokal: Teilstrings betrachten, dyn. Programmierung

semi-global: Lücken am Anfang und Ende ignorieren
erste Zeile / Spalte mit 0 initialisieren / letzte Zeile / Spalte: finde max.

verallgemeinerte Bewertungsfunktionen: Lückenlänge berücksichtigen (Kosten fürs öffnen + Kosten die proportional zur Länge sind)
Bewertungsmatrizen 20×20, PAM-Matrizen, akzeptierte Mutationen
PAM-Einheit: $s \to t$ ohne Insertion/Deletions, eine akz. Mutation je 100 AS.
1: wähle ähnliche Sequenz von selben Vorfahren → „selbe" Länge, Lücken best. → PAM
2: $PAM_1 \to PAM_K$: $F = 20 \times 20$ $P(i,j) = $ w'keit $a_i \to a_j$ in einer PAM-Einheit mutiert

mult. Alignment: kein Lückensymbol in der selben Stelle bei allen Strings.
Position des Zeichens nicht löschbar vorkommen (ohne Lückensymbol)

Abstand: Summe über alle vom Consensus abweichenden Symbole an jeder Pos.

Mult-Consensus-Align-Problem: Optimierungsproblem: mult. Align mit kleinstem Abstand
↳ Summe über alle Positionen und Spalten mit paarweisen Edit-Fkt.
dyn. Prog mit k-dim. Array → exp. Problem $O(k^2 \cdot 2^k \cdot n^k)$

Dis-Mult-Consensus-Align-Problem (DMSPAP): Entscheidungsvariante, mit Bewertungsfkt $\delta(\Sigma \cup \{-\})^2 \to \mathbb{R}$ und $d \in \mathbb{N}$. Liefert true wenn mult. Align. mit SP ≤ d, bzgl. δ

kürzeste gemeinsame Supersequenz über $\Sigma = \{0,1\}$ (D-(0,1)-SSP) $|t| \leq m$ → NP-vollst.
do it-time
↳ DMSPAP = NP-vollst.

paarweise Alignments: S Menge Strings, T Teilmenge S, A'(A'') mult. Align S(T).
↳ A' komp. A'' falls A' = A'', wenn Spalten der Strings mit nur '–' eliminiert werden.

Alignment-Baum: Knoten sind Strings, Kosten mit opt. Alignment zw. zwei Strings
T ist Stern ⟹ Star-Alignment

Bewertungsfkt. $\delta: (\Sigma \cup \{-\}^2) \to \mathbb{R}$ ist gut: $\delta(a,a) = 0$, $\delta(a,c) \leq \delta(a,b) + \delta(b,c) \Rightarrow \delta(a,b) \geq 0$

Heuristiken: FASTA: suche Muster in DB mit Sequenzen
1. Hot-Spots (k=6 DNA/ k=2 Protein) = exakte Matches der Länge k
2. Zusammenfassung: Matrix $M(i,j) = 1$ wenn $p_i = t_j$ für Muster p/Sequenz t
Suche 10 beste Läufe (diagonal) mit HS am Anfang/Ende
Bewertung: Anzahl HS positiv / Länge Lücken negativ
3. Teilalignments zu längeren zusammensetzen
4. Alternativlösung

BLAST: 1. Hits (Teilstrings die ähnlich sind) mit Länge w (11 DNA/3 Proteine)
2. Paare von Hits mit Abstand ≤ d
3. Paare an den Enden erweitern bis Alignment-Bewertung nicht mehr erhöht
wenn Bewertung > Schwellenwert S → High-Score-Pair.

physikalische Kartierung: Marker und Funktion die Positionen angibt
Double-Digest: zwei Enzyme A+B, bilde Teilverdau A, B, AB → Anordnung finden
naiv: $|A|! \cdot |B|!$
Reduktion auf Set-Partition → NP vollständig
Partial-Digest: teilweiser Verdau: $\binom{k}{2}$ Elemente (ideal)
Anordnung mit Backtracking (jeweils größtes verbleibendes Element am linken oder rechten Rand) finden.
Worst-case: $O(2^k \cdot k \log k)$ → Exponentiell, in der Regel aber gute Laufzeit
DDP einfaches Experiment / schwere Berechnung ⟺ PDP aufwendiges Experiment / einfache Berechnung

Hybridisierung mit DNA-Chips, Anordnung für n×m Matrix: Consecutive Ones Property (C1P)
→ PQ-Bäume: jedes Blatt ein Element, P-Knoten Kinder beliebig permutierbar, Q-Knoten Kinder invertierbar entsteht aus universellem Baum durch schrittweise Anwendung der Restriktionen (Zeilen). $O(n+m+k)$

Fehler: 0 zwischen Block von 1 (falsch negativ), 1 zwischen Block von 0 (falsch positiv), Block von 0 (chimären), Lücken ≙ Block von Nullen ⟹ Permutation zur Minimierung
⟹ Problem der minimalen Lückenzahl (durch Spaltenpermutation)
Spaltenabstandsgraph: Knoten ≙ Spalten, ungerichtete Kanten mit Gewichtung Hamming-Distanz (Anzahl Zeilen an denen Spalten unterschiedlich sind)
→ finde Pfad durch Graphen mit geringsten Kosten. Um unterschiedliche Lösungen mit beginnenden / endenden 1er/0er Blöcken vermeiden: Einfügen Spalte mit nur Oen und Kreis suchen → MinLN ≙ TSP (NP-vollständig)

da Dreiecksungleichung: Δ-TSP → 1,5-Approximation, bzw. 2 mit Spann... | BioInf 3

→ minimaler Spannbaum ⇒ Tiefensuche

Bestimmung Basensequenz: Gelelektrophorese (~ 1000 bp)
Shotgun Sequenzierung → Fragment-Assembly Problem (~ 100 Kbp)
1000 Fragmente à 500 bp, Überdeckung 7,5
→ Overlap-Bestimmung (paarweise), nicht exakt → Layout (Anordnung) → Consensus (Majoritätsregel)

Fehlerquellen: Sequenzierungsfehler, Chimären, unvollst. Überdeckung, Orientierung, Repeats

Shortest-Common-Superstring Problem (SCS): Optimierungsproblem ⇒ fordere Teilstringfreiheit statt Längenmaß (Minimierung) → Kompressionsmaß (Maximierung): Maximum-Compression (MCCS)
Berechne Overlap und Distanzgraphen für alle Strings, suche kürzesten/längsten Pfad ⇒ TSP, auch Entscheidungsvariante NP-vollst. (Reduktion auf HP-Problem)
Greedy-Superstring (bzgl. Overlap): Größten Overlap suchen, Kanten löschen, etc.
→ Laufzeit $O(n \cdot \|S\|)$, 4-Approx., mind. 2-fach.
Bzgl. MCCS sogar 2-Approx. Beweis 3 Approx: Greedy-Merge verhindert max. 3 Merges der opt.
min Cycle Cover: gerichtete Kreise, sodaß jeder Knoten in genau einem Kreis → Polynomiell
berechne min Cycle Cover → wähle in jedem Kreis Repräsentanten → erneut Cycle-Cover →
Kante mit geringstem Overlap löschen → Konkatenation → 3-Approx für SCS

Reconstruction-Modell: Teilstring-Edit-Distanz: direkte Fehlerberücksichtigung (jedoch nicht unvollst. Überdeckung, Chimären, Repeats), inkl. Orientierung → NP-schwer

SBH: Sequenzierung durch Hybridisierung: Spektrum DNA bestimmen für Länge l ≙ Alle Teilstrings der Länge l, die in DNA enthalten. Spektrum-Graph: Knoten für alle Teilstrings der Länge l-1, Kanten (gerichtet) mit Beschriftung des Zeichens, um hinzukommen → finde Eulerpfad, dann einfach kompatible String (enthält jeden Teilstring genau einmal) → Linearzeit
Problem: nur etwa 200 bp rekonstruierbar, was auch durch Kettenabbruchmethode erreicht. ist.

Signale in DNA-Sequenzen: finde "interessante" Regionen (Restriktions-/Bindungsstellen/Gene)
Finden des längsten gemeinsamen ähnlichen Teilstrings (Benutzung durch Hamming-Distanz)
→ Consensus-String Problem: NP-schwer
r≥3 Approx. Algorithmus mit Eingabe l (Länge): $1+O(\sqrt{\ldots})$ Approx, $O(m \cdot l + r^{\ldots})$

Finde häufige/seltene Strings. Autokorrelationspolynom: Potenz um die Längen von Suffix = Präfix
Zufallsstring $s = s_1 \ldots s_m$, $X = \Sigma_{i=1}^{m} X_i$, $p = 1/4^l$, $E[X] = m \cdot p$, $Var[X] = \ldots m \cdot (2\,corr(1/4) - (2l-1)p - 1)$

Hidden-Markov-Modell (HMM): W'keiten für bestimmte Ereignisse und Transitionswahrscheinlichkeit zu anderen Modell; Berechnung W'keit eines bestimmten Pfades zu einer Ausgabefolge
Suche besten Pfad zu einer Ausgabe: HMM-Dekodier-Problem: polynomiell mit dyn. Prog.
→ Viterbi-Alg.: String mit Länge n für k Zustände: $O(n \cdot k^2)$.
Finde CG-Inseln mit Hilfe dieses Verfahrens

Phylogenetische Bäume: Metrik (Entfernungen zw. unterschiedlichen Knoten > 0, Symmetrie, Dreieck)
Ultrametrik ($\forall a,b,c \in A$: (a,b), (a,c), (b,c) → zwei gleich, dritte kleiner)
UPGMA-Alg: Ultrametrik → phylog. Baum in $O(n^2)$: immer Knoten mit kürzestem Abstand verbinden.
additiver Baum: zulässiges Distanzmaß gegeben, kompakt: nur Knoten aus Metrik
Konstr. durch minimalen Spannbaum: nehme jeweils billigste Kante, falls nicht: Fehler
Parsimony: bin. Baum mit Blattbeschriftung DNA mit Länge k.
Fitch: Füge Wurzel ein, von den Blättern zur Wurzel: Schnittmenge der Beschriftung auf inneren Knoten, falls ∅ Vereinigung und Kostenmaß +1 für jede Position
Von der Wurzel zu den Blättern jeweils bel. Symbol wählen, falls ... nicht in Knoten
→ minimale Kosten M. Hamming-Abstand. $O(n \cdot k)$
Min-Par-Top-Problem: geg. Strings → finde ungew. phylog. Baum der Parsimony minimal
→ Exponentiell groß → daher Heuristik mit Quartett, jeweils 4 Strings wählen.

Quartett: ung. phylog. Baum für 4 Taxa, optimale min. Parsimony

Wenn T ung. phylog. Baum für S (Menge von n Taxa) und Q Quartett (ab, cd)
Q ist Viererstück von P_1 und P_2 disjunkt, mit P_1 Pfad a–b in T und P_2 Pfad c–d.
aus Menge aller konsistenten Quartette → konsistenter Baum einfach mit polynom. Aufw.
↳ nicht bestimmbar, dass Parsimony für alle Teilmengen $\in O(n^4)$
finde ung. phylog. Baum mit max. viele konsistente Quartette → Max-Quartett-Consist → NP

Quartett-Puzzling: optimale Quartette für jede 4-elem. Teilmenge berechnen, dann zufällige Reihenfolge der Taxa wählen mit opt. Quartett der ersten 4 Elemente beginnen, für jedes weitere Element die Kante mit min. Kosten finden (jeweils für alle drei 3-Elementzyklen Knoten des Baums + neues Element die Kanten a_1, a_2 um eine erhöhen, wenn $Q(a_1 a_2 / a_3 b_i)$ und dort Element einfügen.

Genom-Vergleich: Reversals → Permutationen die einen Abschnitt umdrehen
MinSR: Permutation gegeben, ursprüngliche (Identität) mit min. Reversals wieder herstellen: NP
Breakpoints: Unterbrechungen an denen die Permutation nicht zusammenpassen,
max. zwei Breakpoints je Reversal entfernbar → opt. braucht min. $\lceil \frac{\text{Anz.}}{2} \rceil$ Reversals.
2-Approx: in jedem Schritt durch ein Breakpoint entfernen.
Erweiterung S_0, S_1 aufsteigend, andere ein-absteigend
opt. absteigender Strip: eliminiert einen Breakpoint / schafft zwei Breakpoints $O(n^2)$

Sekundärstrukturen: Minimierung freier Energie: primär → Sekundärstruktur, finde
homologe Basenpaare (Stems)
Alg. von Nussinov Colyn: Programmierung $\Theta(n^3)$ max. Anzahl Basenpaare
Erweiterung: freie Energie betrachte $O(n^3)$, Rekursion: kein Paarung, Paarung, Modell
neben betrachteten Positionen
Verbesserung durch Alg. von Zuker: berücksichtige Stem, Bulge, Haarnadel, Interior Loop
durch weitere Metrik: $O(n^4)$
Strukturvorlage

Protein-Faltung: betrachte HP-Modell (hydrophob 1 / hydrophil, polar 0) und Einbettung in 2 oder 3 dimensionales Gitter, maximiere 1-1 Paare ⇒ Max-1-1-Problem,
→ NP für 2 und 3 Dimensionen
verbrauchen / topologische Nachbarn
self avoiding walk
(1,1)-Paar (i,j) ⇒ i gerade, j ungerade oder umgekehrt: Faltung vorgenommen

4-Approx: finde in Mengen mit 1en auf, aufgesplittet nach geraden/ungeraden Positionen X die Menge mit weniger Elementen, Y die andere
opt. Lösung hat max. $2|X|$ (1,1)-Paare.
XP-Parametrisierung

Finde Faltungspunkt fp, so dass mind $\frac{|X|}{2}$ von X auf der linken und $\frac{|X|}{2}$ von Y auf der rechten Seite liegen, Ebenen von x' mit y' passen, $2|X|$ den rest links/rechts → y' anordnen.

Protein-Threading: Primärstruktur auf bekannte Tertiärstrukturen zurückführen. Daraus Datenbank mit Keimen von Tertiärstrukturen (Faltblatt/Helices) und minimale/maximale Längen von Schleifen daraus ableiten (als 5-Tupel) und Alignment finden, so dass Lücken nur in Schleifen auftreten und keine Keime zerstören.