0.1 Schnelle Paarweise Alignments

- Eine Sequenz in langen Sequenzen suchen
- Viele Sequenzen in einer suchen
- Heute: Entweder Millionen Sequenzen (Genome) durchsuchen
- Oder Millionen Sequenzen auf eine mappen
- Für Beides: Seed und extend Verfahren

0.1.1 BLAST

- Basic Local Alignment Search Tool
- Für grosse Datenbanken (zB alle Genome, alle Proteine etc.)
- Ganz zuerst: look-up table von datenbank basteln
- Zuerst: Entfernung von low complexity regions
- K-mers als
- Hochscorende Wörter finden
- Die in Datenbank suchen
- Seed alignments verlängern. (ungapped, bis score tiefer fällt als threshold)
- Neuer: mit gaps
- Was mindestscore erreicht hat, ist HSP
- HSP werden nach expectation (E) value gescored
- E value: Wie oft erwarte ich zufällig ein solches Ergebnis

0.1.2 BLAT

- Blast Like Alignment Tool
- Oft (und genau) in einem Genom suchen
- Zuerst hash-tabelle der k-mere des Genoms
- 3 Approaches:
  1. Perfekte Übereinstimmung
  2. 1 mismatch in k-meren erlaubt (grössere k-mere)
  3. Mehrere beisammenliegenede perfekte Matches
- Nukleotides: variante 3 mit 2x11
- Proteine (3) mit 3x4 oder (1) mit 5 k-mer size
- Beieinanderliegende hits werden verbunden (splice sites)

0.2 Burrows wheeler Alignments

0.3 Burrow Wheeler Transformation: BWT

Schritt 1: Erstellen aller Zyklischen Permutationen (nach anhängen vom Endcharacter) eines Strings
Schritt2: Sortieren, alphabetisch:

EIT$FREIZ
EIZEIT$FR
FREIZEIT$
IT$FREIZE
IZEIT$FRE
REIZEIT$F
T$FREIZEI
ZEIT$FREI
$FREIZEIT

Die BWT ist nun die letzte Spalte dieser Sortierung: ZR$EEFIIT
BWT reicht, um String wiederherzustellen.
Suche Endcharacter in BWT, erster Buchstabe ist der, der in der ersten Spalte steht.

1 E Z
2 E R
3 F $
4 I E
5 I E
6 R F
7 T I
8 Z I
9 $ T

StepA Gehe von letzter zu erster Spalte, trage Buchstaben ab.

StepB Suche den entsprechenden Buchstaben in BWT, gehe zu dem Buchstaben

- Weiter mit StepA.
- Kommt ein Buchstabe öfter vor, dann entspricht der $x.$ in der ersten Spalte dem $x.$ in der BWT.

0.3.1 FM-Index

Um in BWT zu suchen, braucht man 2 Tabellen:
$C(c)$: wie oft kommen im String Buchstaben vor, die kleiner als c sind.
$Occ(c, k)$ Wie oft kommt Buchstabe c im Praäfix [1...k] der BWT vor
Dann kann in konstanter Zeit die Existenz eines Substrings untersucht werden:

- Substring wird von Hinten nach Vorne bearbeitet
- Es wird immer Beginn und Ende des Präfixes in BWT angegeben.
- Für den esten bearbeiteten Buchstaben a gilt:

$$\begin{aligned} \text{Beginn} &= C(a) + 1 \\ \text{Ende} &= C(a+1) \end{aligned}$$

- hierbei ist $a+1$ der in der Ordnung nächste Buchstabe aus der BWT
- Für die weiteren Buchstaben r_x gilt:

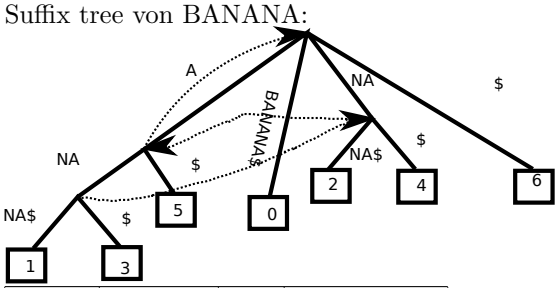
$$\begin{aligned} \text{Beginn} &= C(r_x) + Occ(r_x, Beginn - 1) + 1 \\ \text{Ende} &= C(r_x) + Occ(r_x, Ende) \end{aligned}$$

- Ist Beginn $>$ Ende, existiert der Substring nicht

0.4 Suffix Array Based Aligners

- Mit Longest common prefix und lcp intervallen equivalent zu s-tree
- Mittels Suffix links (xA zu A) schnelleres "falsch" mappen
- Wird auch zur Seed suche eingesetzt

Suffix tree von BANANA:



index	original	lcp	suffix
0	1	0	ANANA$
1	3	3	ANA$
2	5	1	A$
3	0	0	BANANA$
4	2	0	NANA$
5	4	2	NA$
6	6	0	$

0.5 Nanopore Sequencing

0.5.1 Minimizer k-mers

- k-mer (Alle) Substrings der Länge k
- $|S| - k + 1$ k-mers pro Sequence S
- Minimizer ist (für eine bestimmte Sortierung) der minimale k-mer innerhalb eines Fensters von k-mers
- Fenster haben Länge w
- if $w \leq k$ befindet sich jeder Buchstabe von S in mindestens einem Minimizer

0.5.2 Minimap2

Weitere Vereinfachung der minimizer:

- HPC(s) kontrahieren aller homopolymere: GGATTTTCCA GATCA

Seeds (minimizer) werden zusammengehängt (chaining), Chaining score mittels DP.