

# Methoden der Hochdurchsatzsequenzierung

Vorlesung 03

Sommersemester 2021

# Assemblierung

~ das Zusammenführen von Nukleotidseq. zu längeren Fragment (durch Sequenzalignments).

Read  $\rightarrow$  Contig  $\rightarrow$  (Supercontig)  $\rightarrow$  Scaffold  $\rightarrow$  Genom

ACACACAC

ACACA $\times$ 5

## Greedy

- ① paarweise Aln. aller Reads
- ② Suche max. Score eines Aln.
- ③ Füge Fragmente dieser Aln. zusammen
- ④ Wiederhole Schritte ① + ③

## Nachteile

- kann nicht mit Repeats umgehen
- sehr langsam, sehr speicherintensiv
- Orientierung der Reads

bioinformatics  
STEVE

Capacity: fast  
- 8MB  $\rightarrow$  schnelle  
- 1MB  $\rightarrow$  langsam  
- 100MB  $\rightarrow$  für 1000  
64215768936

# Assemblierung

Mit den folgenden Problemen sollte ein "gutes" Assemblierungstool zu recht kommen

- Repeats
- Orientierung von Reads
- Speicher / Laufzeit
- Sequenzierungsfehler (Mutation, Kontamination, DNA-Modifikation)
- Lücken im Genom
- Chimärsequenz erkennen  
(Zusammenlagerung nicht zusammengehörender Sequenzen)

# Assemblierung

Gütekriterien für Genom-Assemblierungen

NSO-Wert  $\sim$  Länge des kürzesten Contigs, das in Summe mit allen längeren Contigs die Hälfte des Genomassembly abdeckt.

→ Contigs der Länge nach sortieren, dann von lang nach kurz die Contiglängen aufsummieren bis 50% der Basen des Gesamtassemblys erreicht sind → Nimm die Länge des zuletzt aufsummierten Contigs

LSO-Wert  $\sim$  Wie NSO, nur statt der Länge des zuletzt aufsummierten Contigs betrachten wir die Anzahl an Contigs um NSO zu erreichen.

N90 / L90

BSP. 10, 9, 8, 7, 6, 5, 4, 3, 2  $\Sigma 54$

10 + 9 + 8 = 27  $\xrightarrow{50\%}$

NSO = 8      LSO = 3

Bioinformatics  
STEVE

Carrying: pad  
- similar to sketching  
- for 1000

64215768936



- |             |   |
|-------------|---|
| 1. Greedy   | SSAKE                                     |
| 2. Hamilton | Newbler                                   |
| 3. Euler    | Velvet,<br>Trinity, Spades,<br>SoapDenovo |

# Assemblierung

## Hamilton

### Shortest - Superstring - Problem (SSP)

$\sim$  SS von zwei Strings  $X$  und  $Y$  ist ein kürzester String  $Z$  von dem sowohl  $X$  als auch  $Y$  Substrings sind.

→ einfach erweiterbar auf  $n$  Strings

- Das SSP ist NP-schwer (NP-vollständig)
- Es gibt nicht immer eine eindeutige Lösung für das SSP
- Kann in das TSP umgewandelt werden (durch Graphdarstellung)

→ Reduzierung des SSP zum TSP-Hamiltonpfad

bioinfo I → Statistik  
STEVE

Capping: post  
- 800000 → 800000  
- 100000 → 100000  
- 100000 → 100000

64215768936

# Assemblierung

## Hamilton-Pfad

ist ein geschlossener Kreis in einem Graphen, der jeden Knoten genau einmal enthält.

Wir definieren einen Overlap( $s_i, s_j$ ) als die Länge des längsten Präfixes von  $s_j$  der auch Suffix von  $s_i$  ist.

Overlap(ATCCA, CCAATT)

## Algorithmus-Idee

Hausaufgabe

$S = \{ATC, CCA, CAG, TCG, AGT\}$

- jeder Read ist ein Knoten in unserem Graph
- jedes <sup>Kantengewicht</sup> zwischen zwei beliebigen Knoten entspricht dem Overlap der dazugehörigen Reads (↻ der Graph ist gerichtet)
- finde den schwersten Hamiltonpfad

## Zwei große Probleme

↳ das liegt an der Problemdefinition

- ① Können nicht sinnvoll mit Repeats umgehen.
- ② Kann nicht mit Lücken umgehen.

Dominik  
STEVE

Copy: pad  
- solution  
- longest  
- find

6421576836





