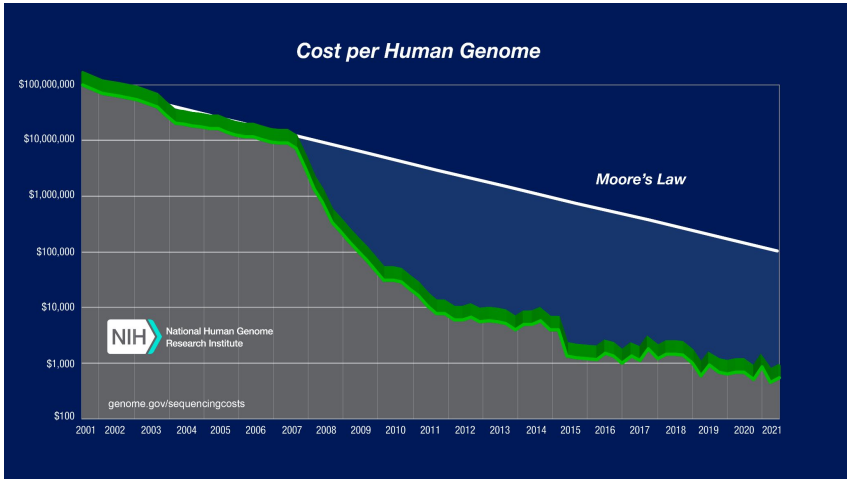
Sequence-read extraction from Counting de Bruijn Graphs

Dmytro Horyslavets

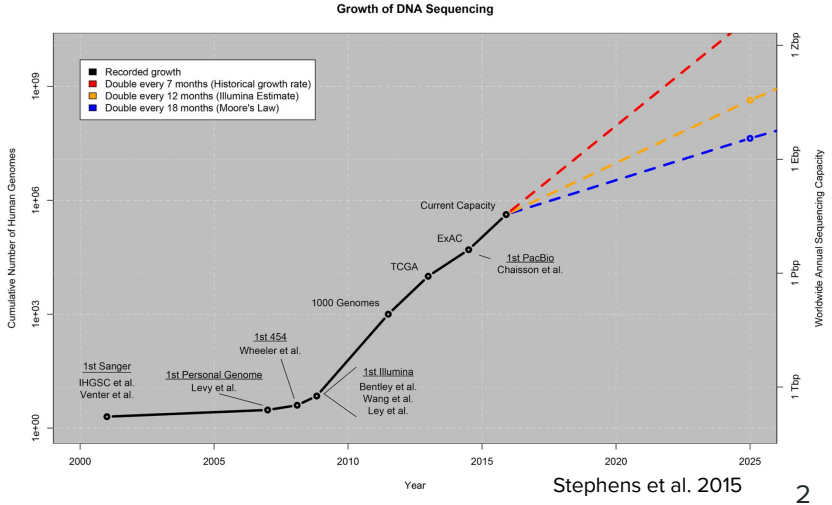
Mentors: Andre Kahles, Harun Mustafa (ETH Zürich)

Motivation

- both amount and sequencing capacity for biomedical sequencing data show **exponential growth**
- broad need for **indexing of and search in (raw) sequencing data** at petabase scale
- **compressed k-mer graphs** are a promising technology to meet these demands



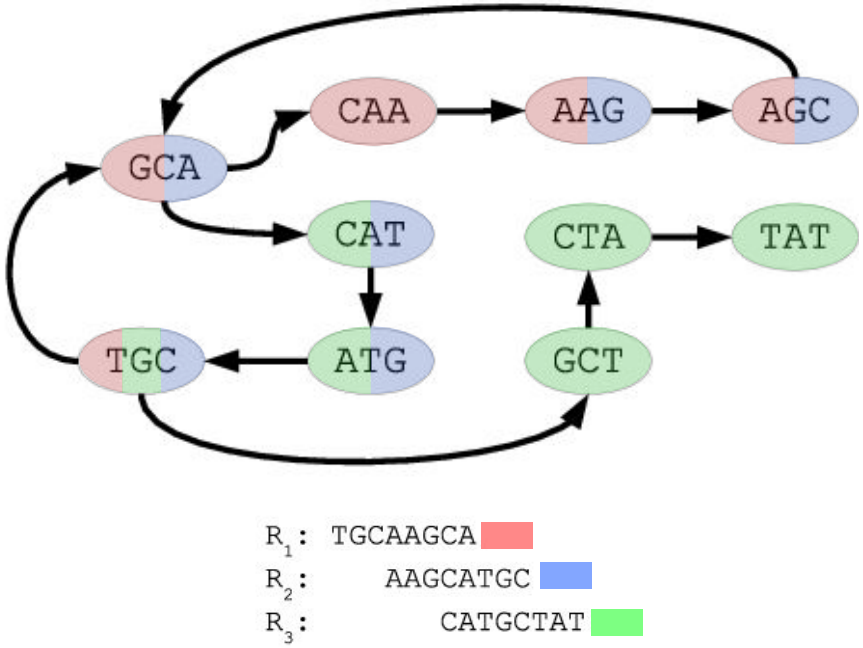
https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data



Stephens et al. 2015

Background

- **De Bruijn graph** is a directed graph of overlaps between sequences of symbols.
- A **Colored** (annotated) DBG is a generalization to distinguish multiple samples.
- A **Counting** DBG is a notion generalizing annotated DBG by supplementing each node-label relation with additional attributes.

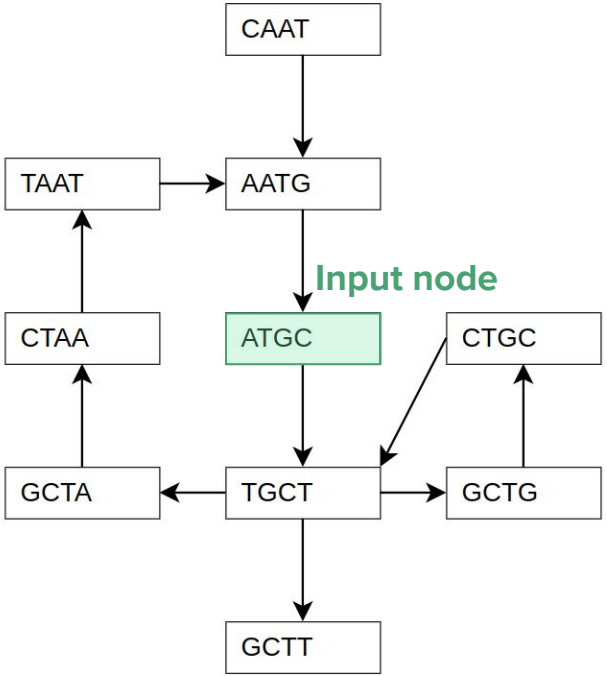


Source: https://www.researchgate.net/figure/Example-reads-and-the-corresponding-de-Bruijn-graph-for-k-3-The-edge-labels_fig1_260713899

Task

Given an annotated **Counting de Bruijn Graph** and a query sequence, return the set of **all input read sequences** that overlap with the query.

Algorithm overview

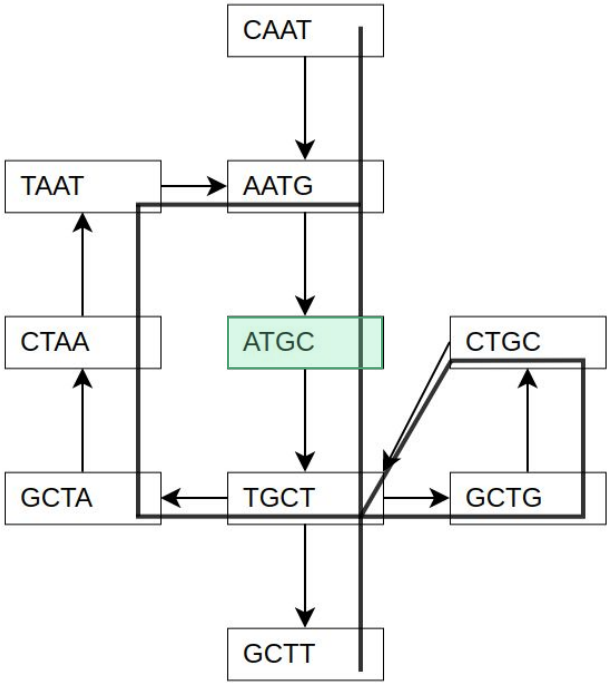


Input node: **ATGC**

1. Traverse the graph forward and backward from the input node.

Source: https://metagraph.ethz.ch/

Algorithm overview

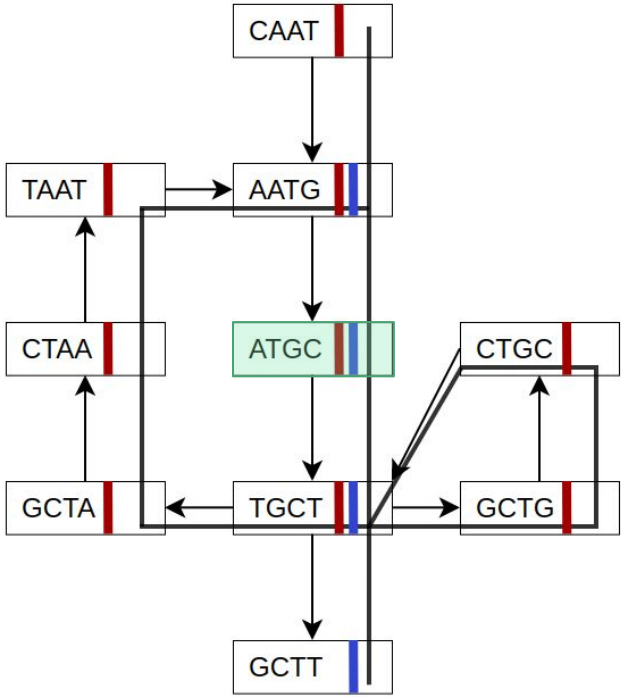


Input node: **ATGC**

1. Traverse the graph forward and backward from the input node.

Source: https://metagraph.ethz.ch/

Algorithm overview

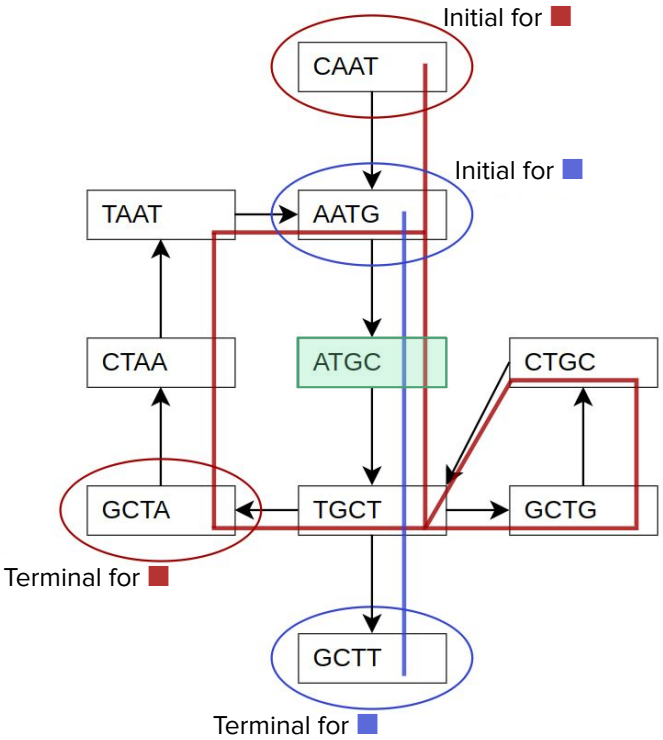


Input node: **ATGC**

1. Traverse the graph forward and backward from the input node.
2. Fetch the nodes' annotations stored in a compressed format.

Source: https://metagraph.ethz.ch/

Algorithm overview

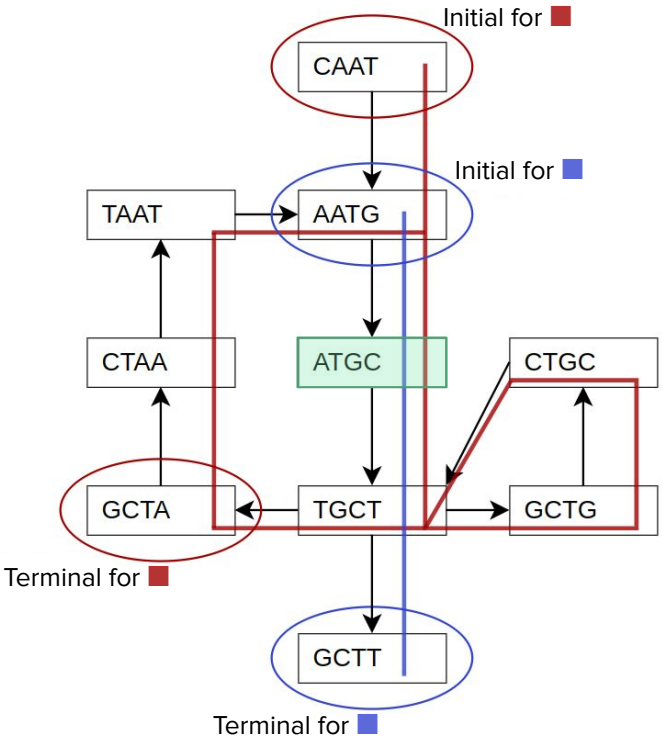


Input node: **ATGC**

1. Traverse the graph forward and backward from the input node.
2. Fetch the nodes' annotations stored in a compressed format.
3. Find walks in the graph that include the input node.

Source: https://metagraph.ethz.ch/

Algorithm overview



Input node: **ATGC**

1. Traverse the graph forward and backward from the input node.
2. Fetch the nodes' annotations stored in a compressed format.
3. Find walks in the graph that include the input node.
4. Reconstruct read sequences.

Result: **CAATGCTGCTAATGCTA**
AATGCTT

Source: https://metagraph.ethz.ch/

Further work

- Apply on real world data. Possible use cases:
  - Single-cell sequencing (search of transcript in the graph within different cell types).
  - Environmental metagenomics (search of unknown DNA, taxonomic profiling).
- Scalability:
  - Implement local graph decompression.
  - Implement batch mode.

Thank you!