Algorithms for studying the structure and function of genomes

Michael Schatz

Feb 6, 2015

JHU Dept. of Computer Science

Genome Biology

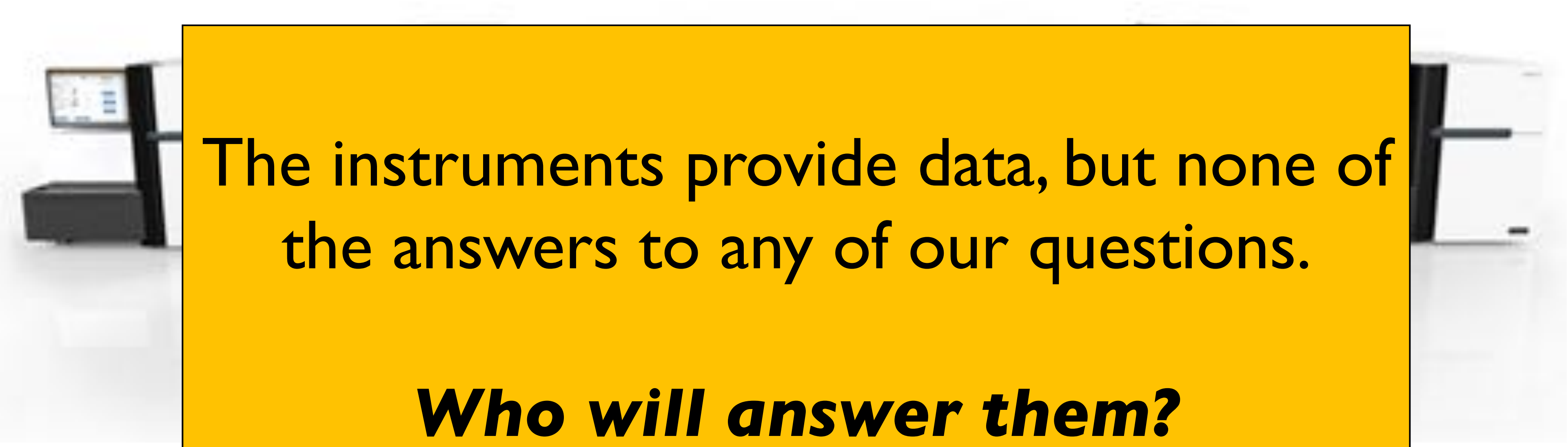
The double helix is a sheet of paper that genetic messages can be written upon.

The particular sequence of nucleotides in your genome, along with your environment and experiences, shapes who you are:

- Physical traits: Height, hair color, skin color, …
- Behavioral traits: Intelligence, Personality, …
- Susceptibility to disease, stress, and toxins
- Response to drug treatments

Finding changes to genome structure can provide powerful clues to its function.

Genomic Data



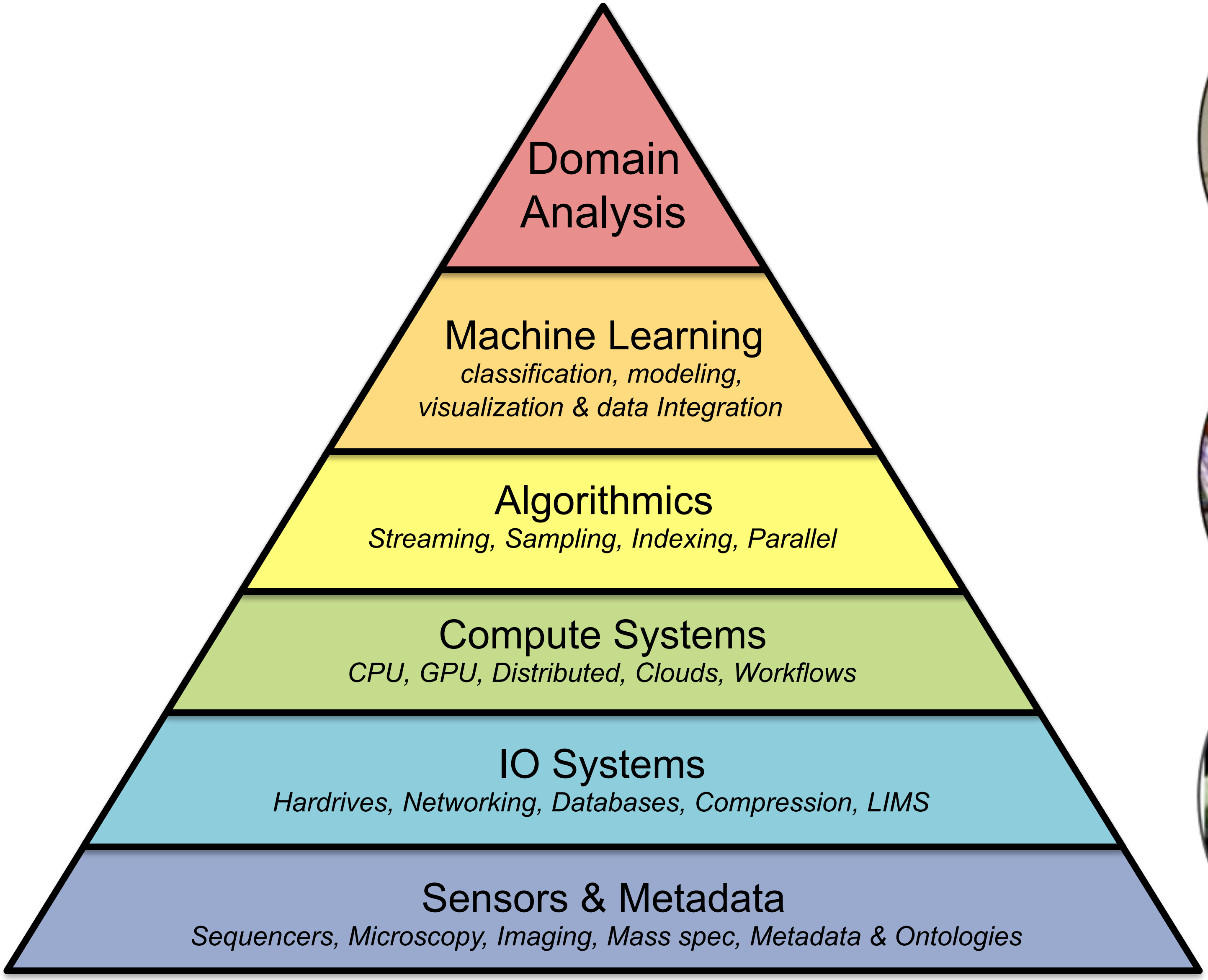
The instruments provide data, but none of the answers to any of our questions.

Who will answer them?

How will they do it?

Worldwide capacity exceeds 35 Pbp/year

Data Science Technologies



Domain Analysis

Machine Learning
classification, modeling, visualization & data Integration

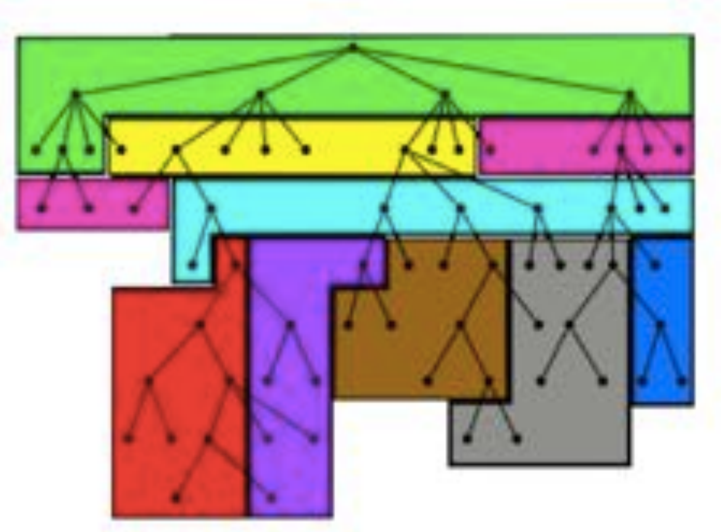
Algorithmics
Streaming, Sampling, Indexing, Parallel

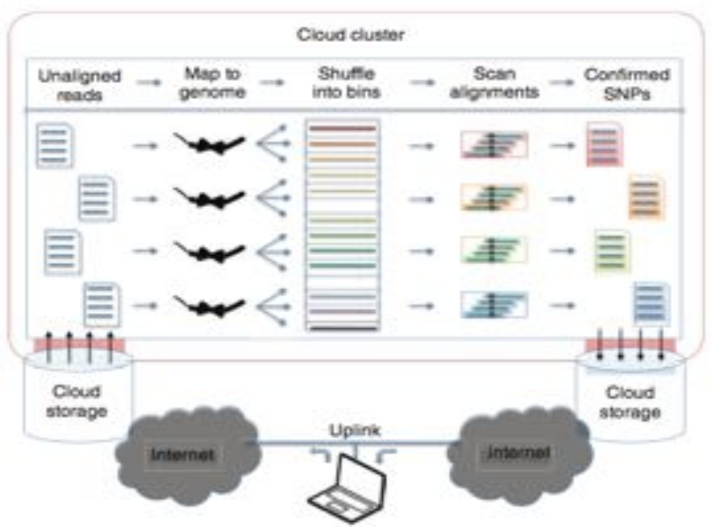
Compute Systems
CPU, GPU, Distributed, Clouds, Workflows

IO Systems
Hardrives, Networking, Databases, Compression, LIMS

Sensors & Metadata
Sequencers, Microscopy, Imaging, Mass spec, Metadata & Ontologies

System Level Advances





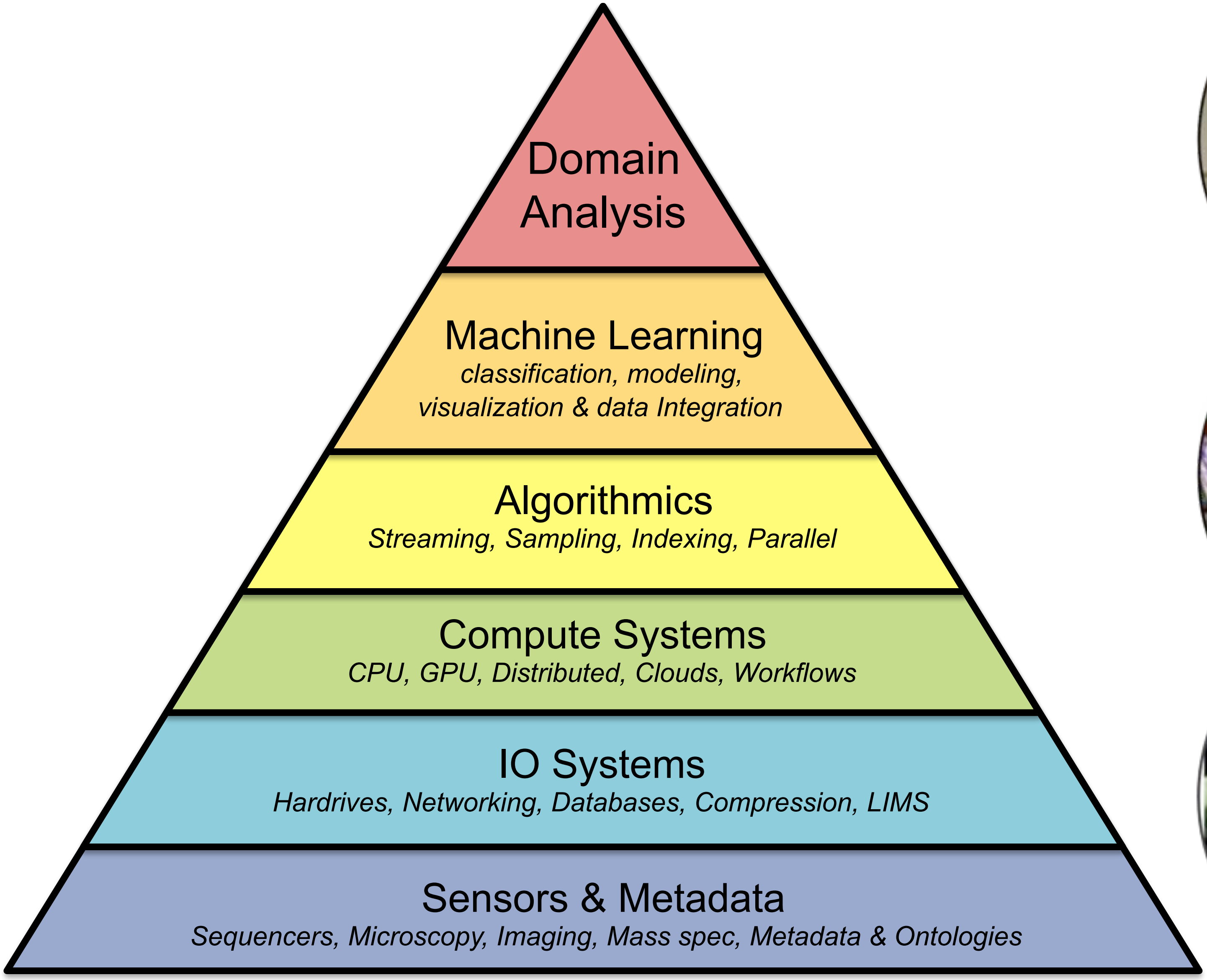
Optimizing data intensive GPGPU computations for DNA sequence alignment
Trapnell, C, Schatz, MC (2009) *Parallel Computing*. 35(8-9):429-440.

CloudBurst: Highly Sensitive Read Mapping with MapReduce.
Schatz, MC (2009) *Bioinformatics* 25:1363-1369.

Design patterns for efficient graph algorithms in MapReduce.
Lin, J., Schatz, MC. (2010) *Proceedings of the 8th Workshop on Mining and Learning with Graphs*

The DNA Data Deluge
Schatz, MC and Langmead, B (2013) *IEEE Spectrum*. July, 2013

Data Science Technologies



Domain Analysis

Machine Learning

classification, modeling, visualization & data Integration

Algorithmics

Streaming, Sampling, Indexing, Parallel

Compute Systems

CPU, GPU, Distributed, Clouds, Workflows

IO Systems

Hardrives, Networking, Databases, Compression, LIMS

Sensors & Metadata

Sequencers, Microscopy, Imaging, Mass spec, Metadata & Ontologies

Genomic Data Structures

Strings

```
..TTGAATTACATG..
    |||  |||
    GAA--ACA
```

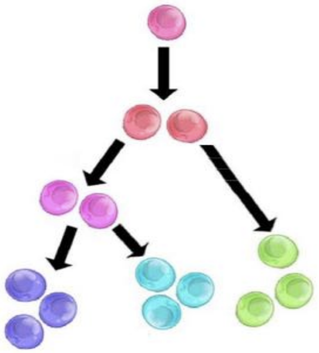
Alignment

Narzisi *et al.* (2014) *Nature Methods*
Lee & Schatz (2012) *Bioinformatics*

Autism Genetics

Iossifov *et al.* (2014) *Nature*
Fang *et al.* (2014) *Genome Medicine*

Trees



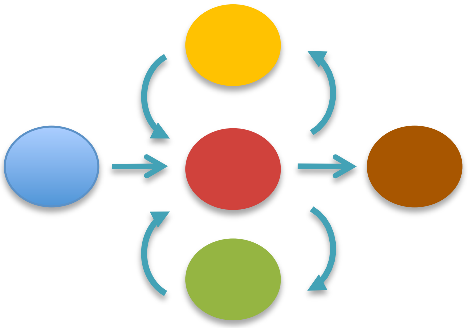
Suffix Trees

Marcus *et al.* (2014) *Bioinformatics*
Trapnell & Schatz (2009) *Parallel Computing*

Microbial Diversity

Donia *et al.* (2011) *PNAS*
Schatz & Phillippy (2012) *GigaScience*

Graphs



String Graphs

Narzisi *et al.* (2014) *Lecture Notes in CS.*
Koren *et al.* (2012) *Nature Biotechnology*

Plant Biology

Schatz *et al.* (2014) *Genome Biology*
Maron *et al.* (2013) *PNAS*

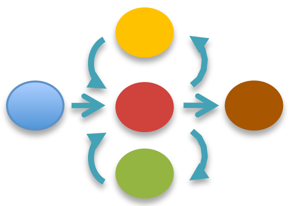
Genomics Graphs

1. **Error Correction and Assembly**

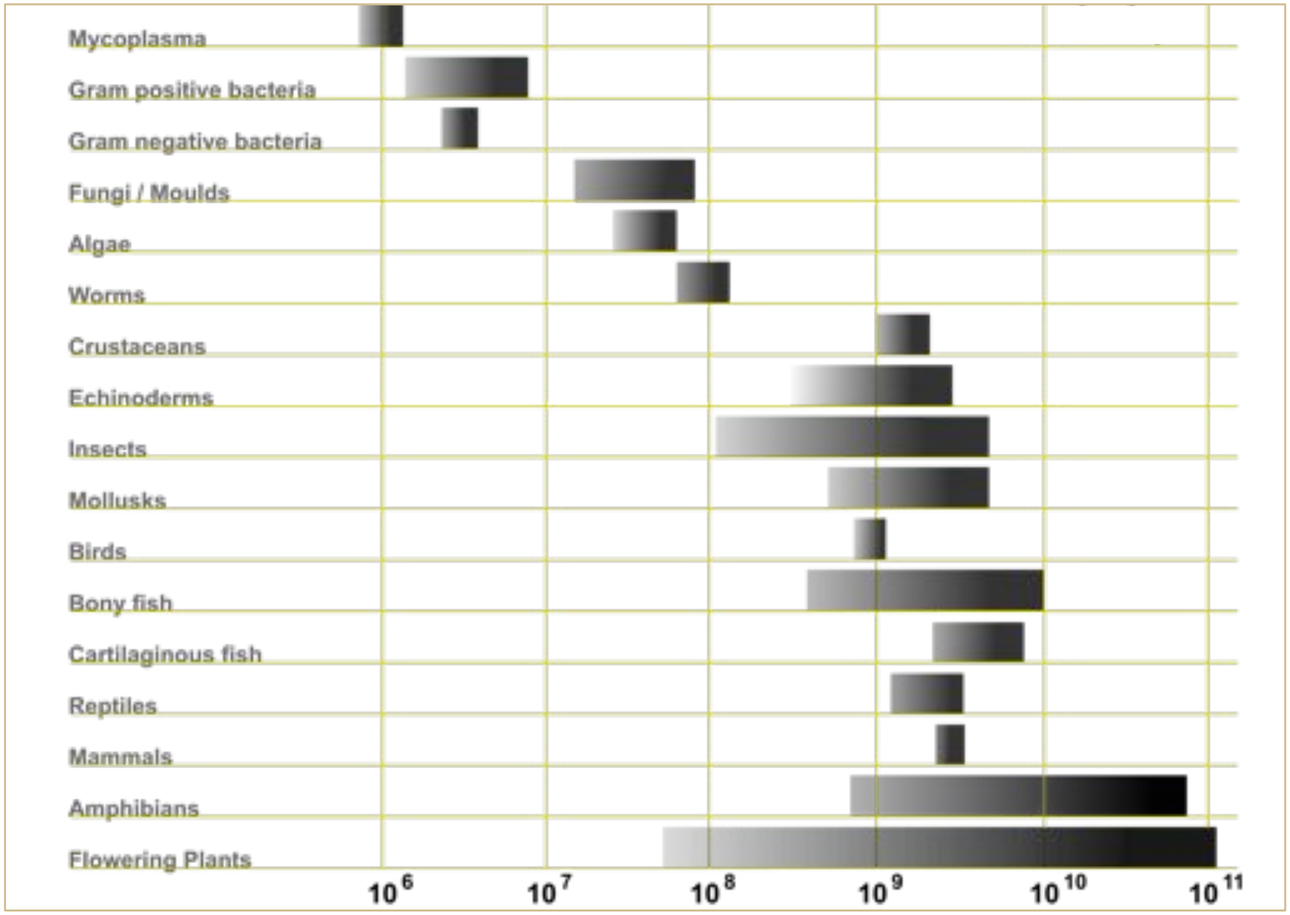
   Long Read Single Molecule Sequencing

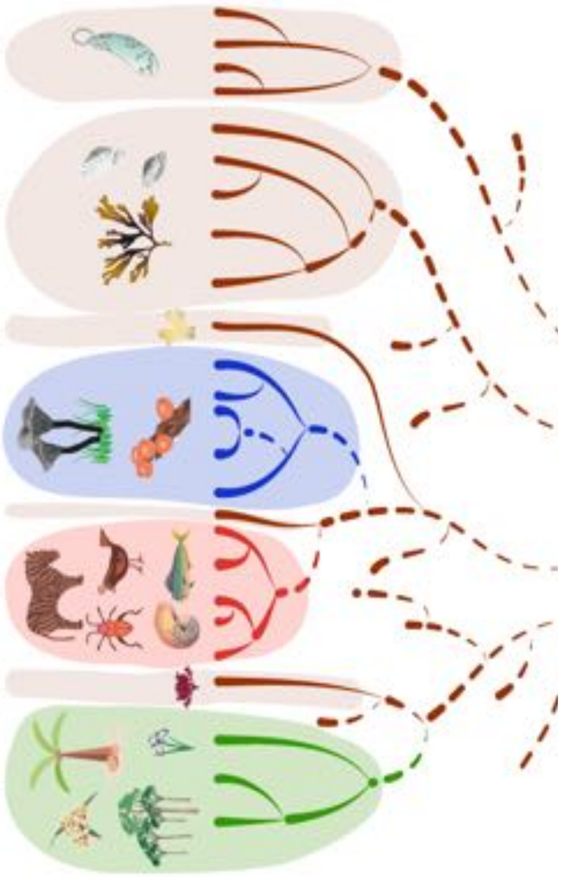
2. **Pan-Genomics**

   Sequence conservation and divergence



Genome Complexity

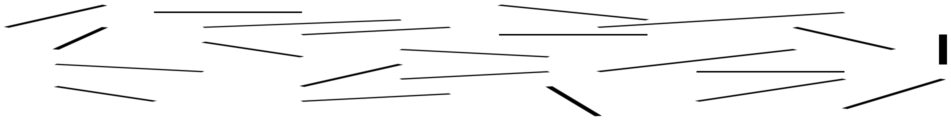




https://en.wikipedia.org/wiki/Genome_size

Sequence Assembly Problem

1. Shear & Sequence DNA



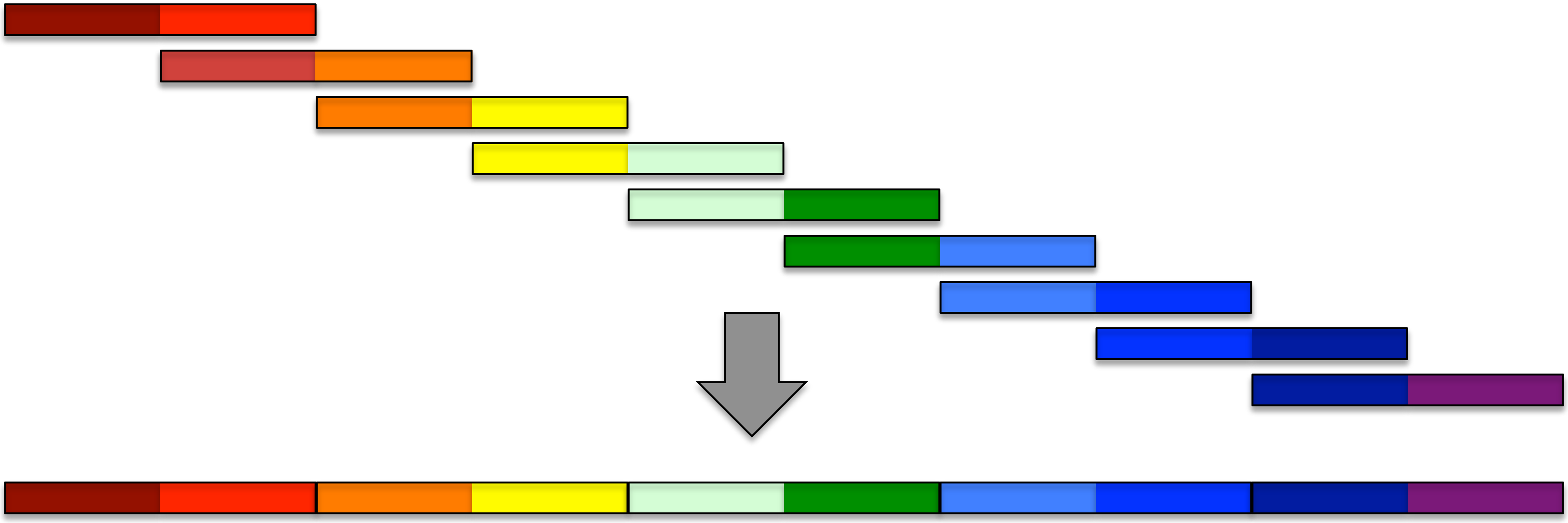
2. Construct assembly graph from overlapping reads

...AGCCTAGGGATGCGCGACACGT

GGATGCGCGACACGTCGCATATCCGGTTTGGTCAACCTCGGACGGAC

CAACCTCGGACGGACCTCAGCGAA...

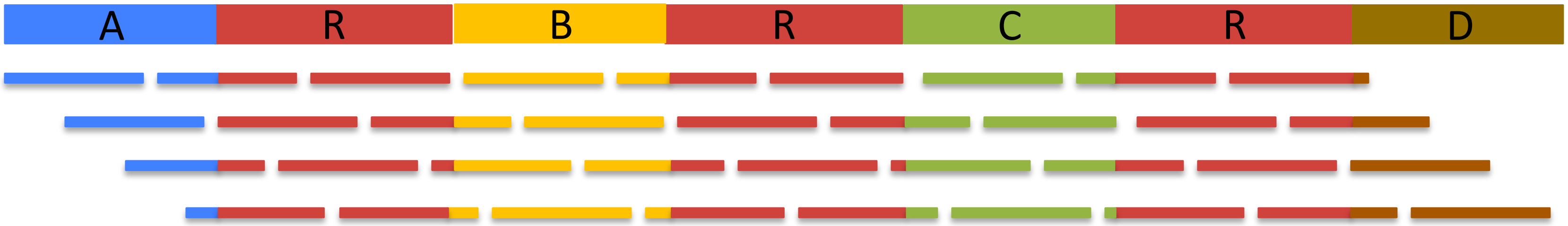
3. Simplify assembly graph

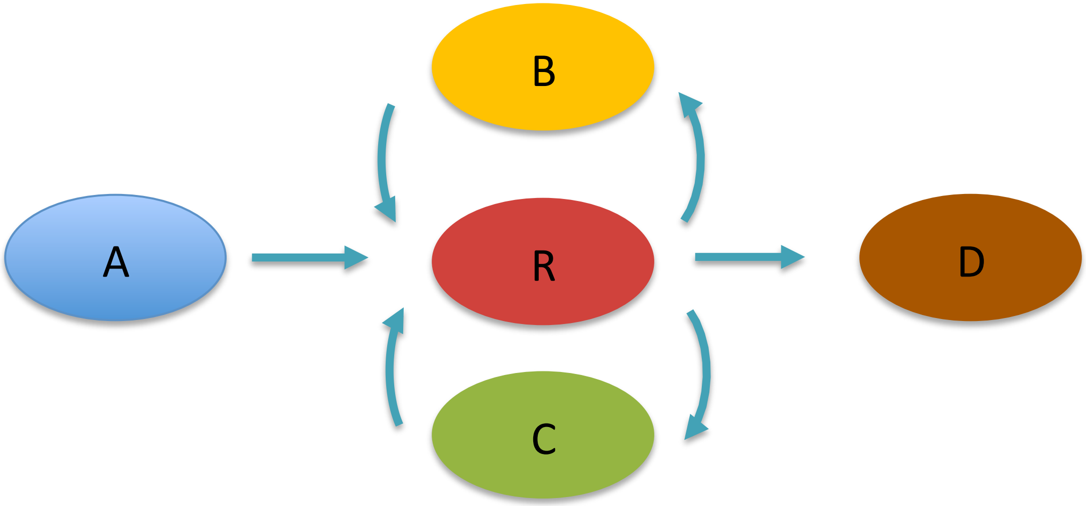


On Algorithmic Complexity of Biomolecular Sequence Assembly Problem

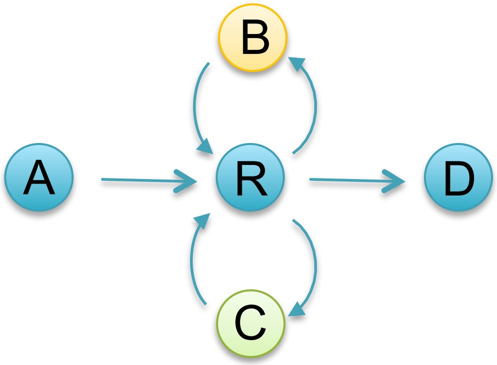
Narzisi, G, Mishra, B, Schatz, MC (2014) *Algorithms for Computational Biology*. Lecture Notes in Computer Science. *Vol. 8542*

Assembly Complexity





Counting Eulerian Tours



ARBRCRD
or
ARCRBRD

Often an astronomical number of possible assemblies

- Value computed by application of the BEST theorem (Hutchinson, 1975)

$$\mathcal{W}(G,t) = (\det L)\left\{\prod_{u\in V}(r_u - 1)!\right\}\left\{\prod_{(u,v)\in E} a_{uv}!\right\}^{-1}$$

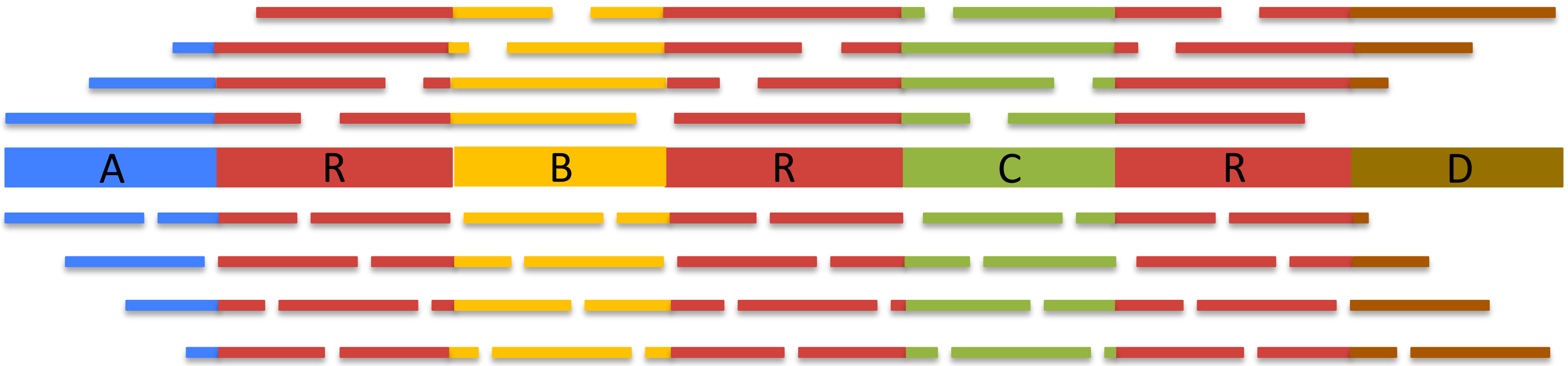
L = *n x n* matrix with r_u-a_{uu} along the diagonal and -a_{uv} in entry uv

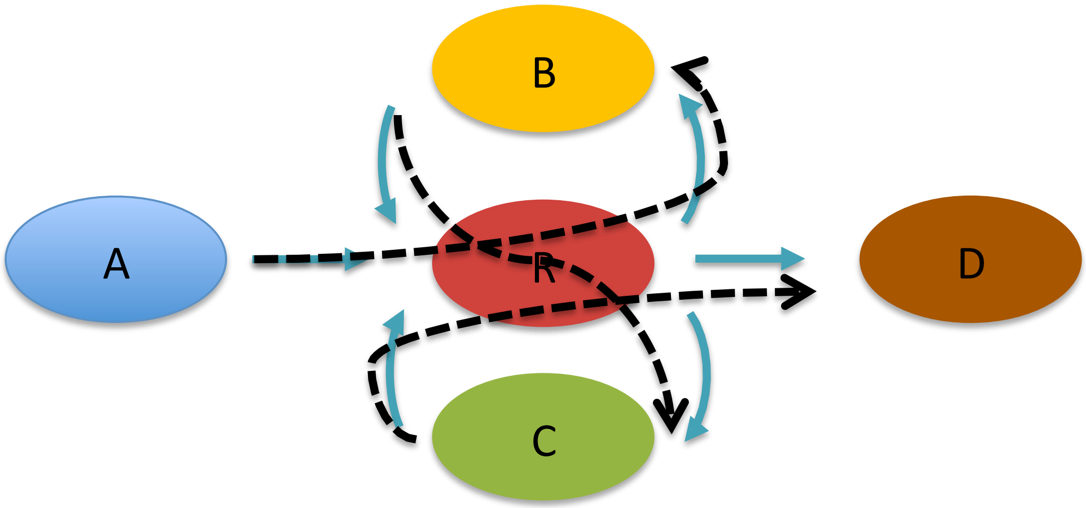
$r_u = d^+(u)+1$ if $u=t$, or $d^+(u)$ otherwise

a_{uv} = multiplicity of edge from *u* to *v*

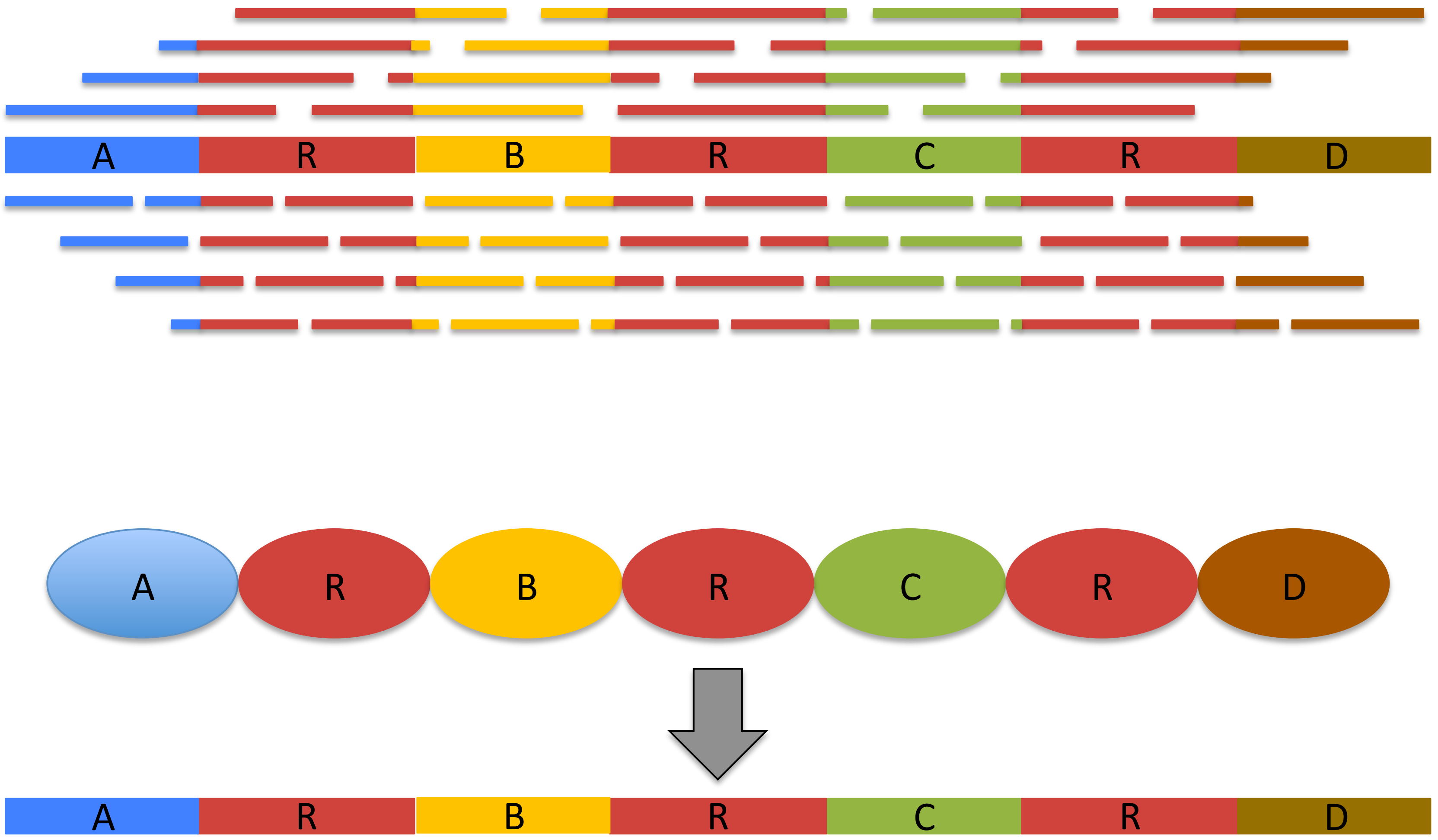
Assembly Complexity of Prokaryotic Genomes using Short Reads.
Kingsford C, Schatz MC, Pop M (2010) *BMC Bioinformatics*. 11:21.

Assembly Complexity





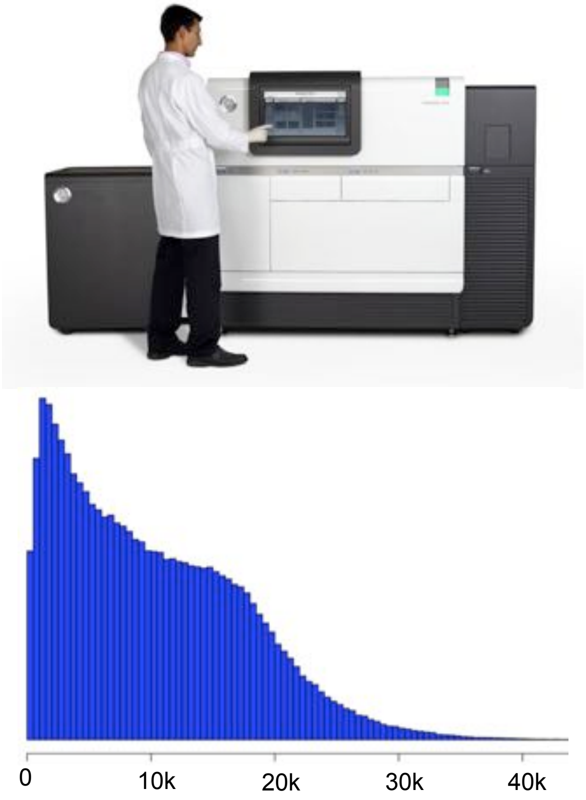
Assembly Complexity



The advantages of SMRT sequencing
Roberts, RJ, Carneiro, MO, Schatz, MC (2013) *Genome Biology.* 14:405

3rd Gen Long Read Sequencing

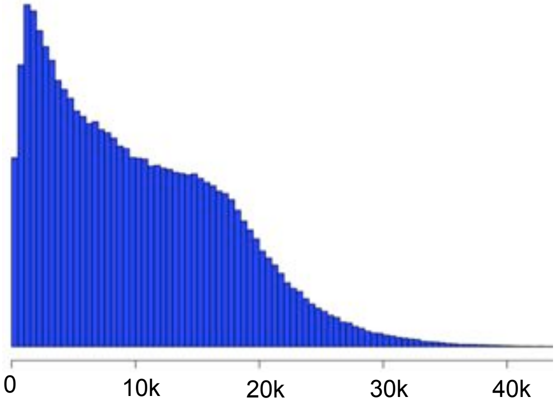
PacBio RS II



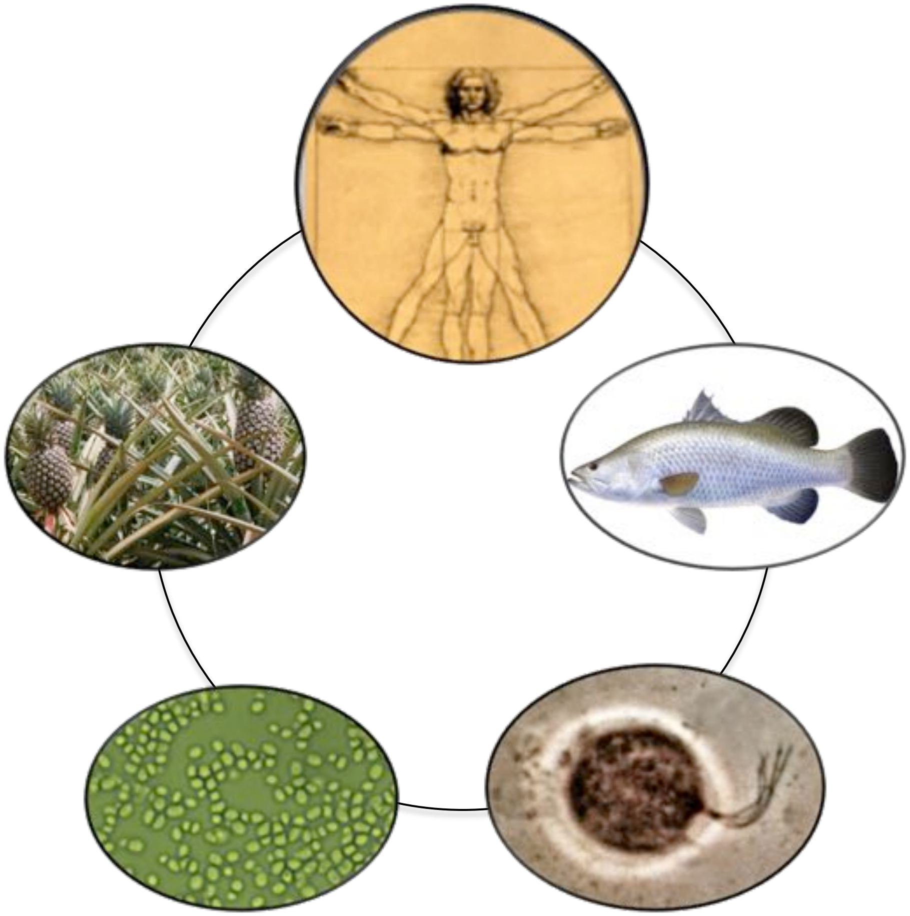
CSHL/PacBio

3rd Gen Long Read Sequencing

PacBio RS II

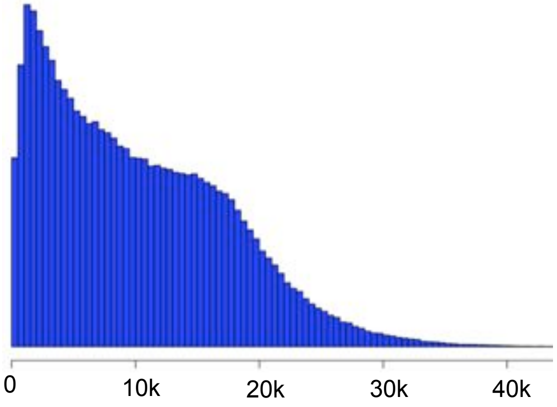


CSHL/PacBio



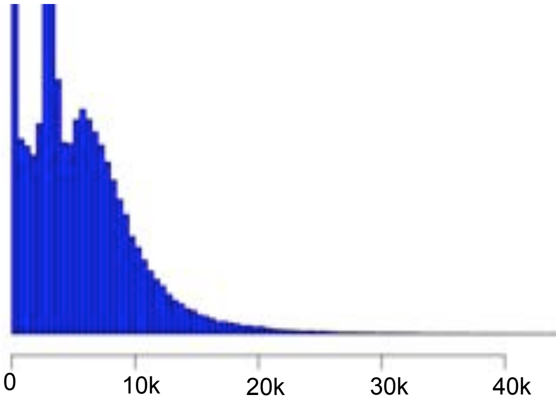
3rd Gen Long Read Sequencing

PacBio RS II



CSHL/PacBio

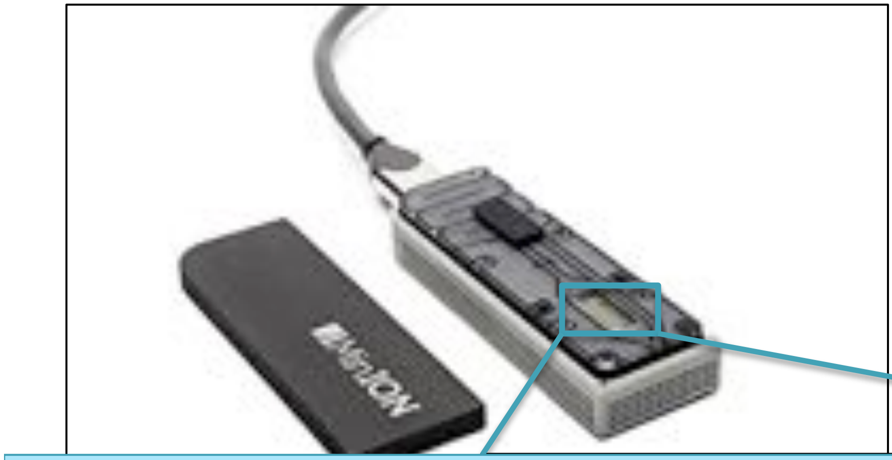
Oxford Nanopore

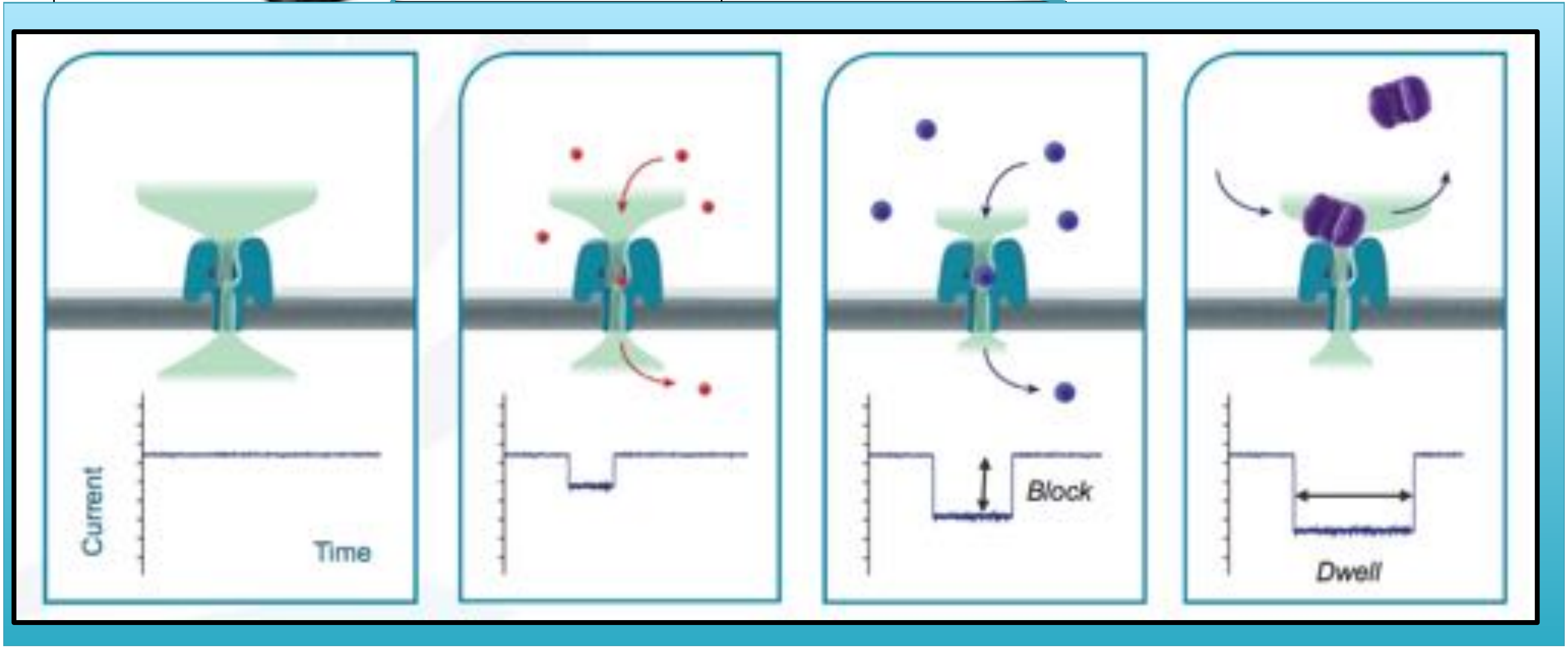


CSHL/ONT

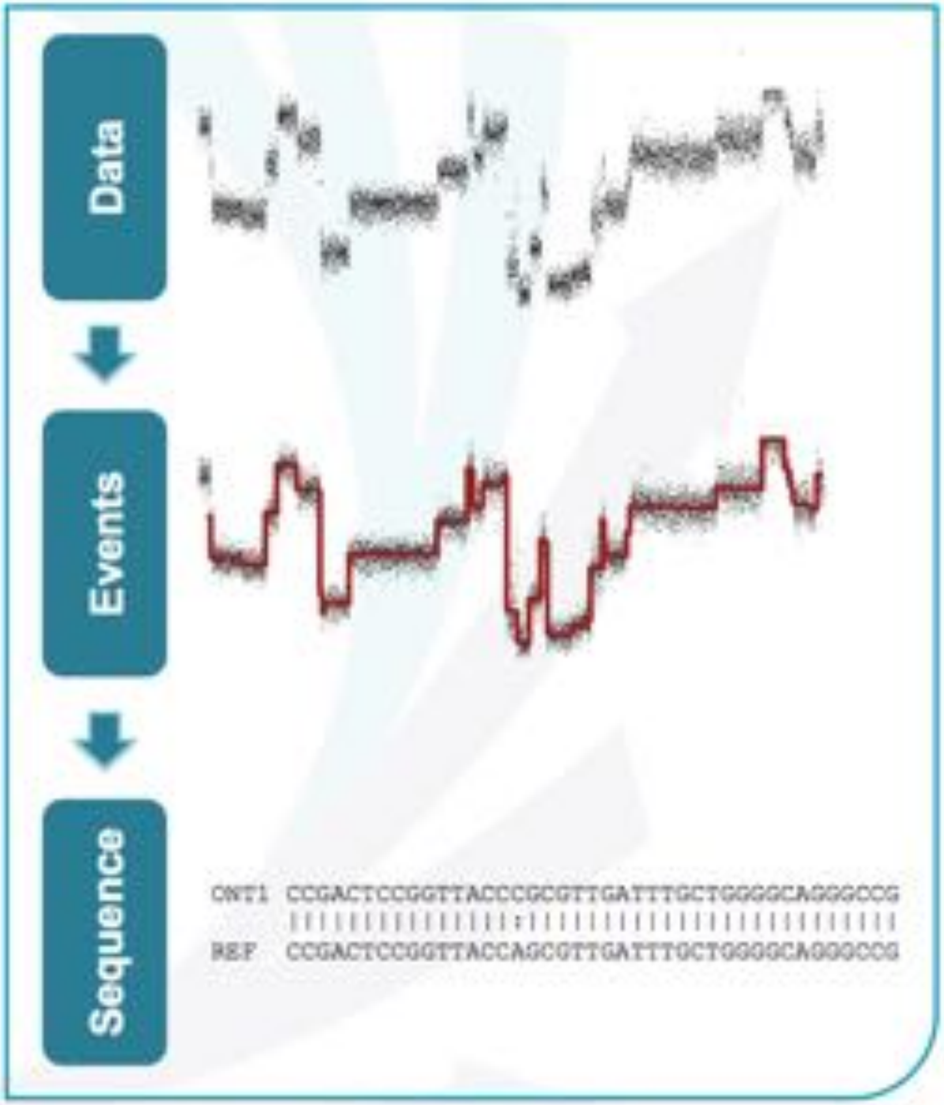
Oxford Nanopore MinION

- Thumb drive sized sequencer powered over USB
- Capacity for 512 reads at once
- Senses DNA by measuring changes to ion flow



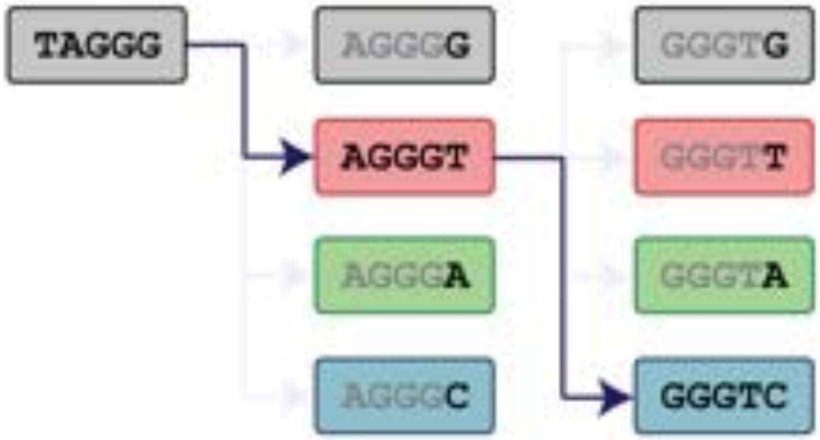


Nanopore Sequencing



```
ONT1  CCGACTCCGGTTACCCGCGTTGATTTGCTGGGGCAGGGCCG
      |||||||||||||||:|||||||||||||||||||||||||
REF   CCGACTCCGGTTACCAGCGTTGATTTGCTGGGGCAGGGCCG
```

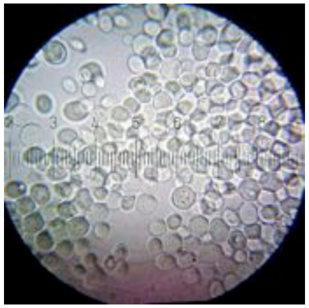
- Hidden Markov model
- Only four options per transition
- Pore type = distinct kmer length

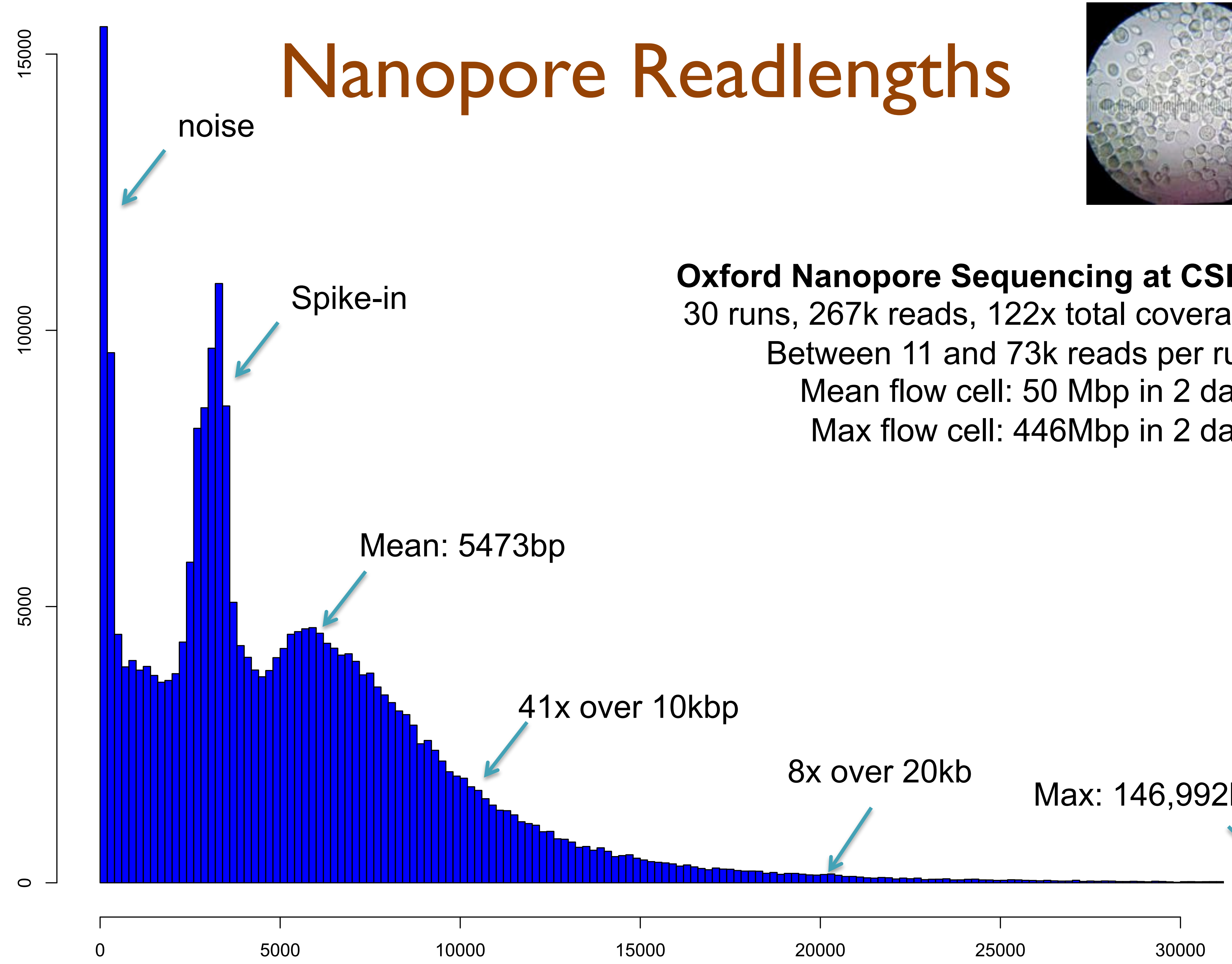


- Form probabilistic path through measured states currents and transitions
  - e.g. Viterbi algorithm

Basecalling currently performed at Amazon with frequent updates to algorithm

Nanopore Readlengths





noise

Spike-in

Mean: 5473bp

41x over 10kbp

8x over 20kb

Max: 146,992bp

Oxford Nanopore Sequencing at CSHL

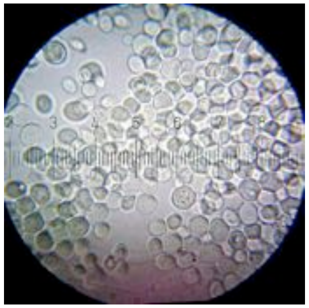
30 runs, 267k reads, 122x total coverage

Between 11 and 73k reads per run!

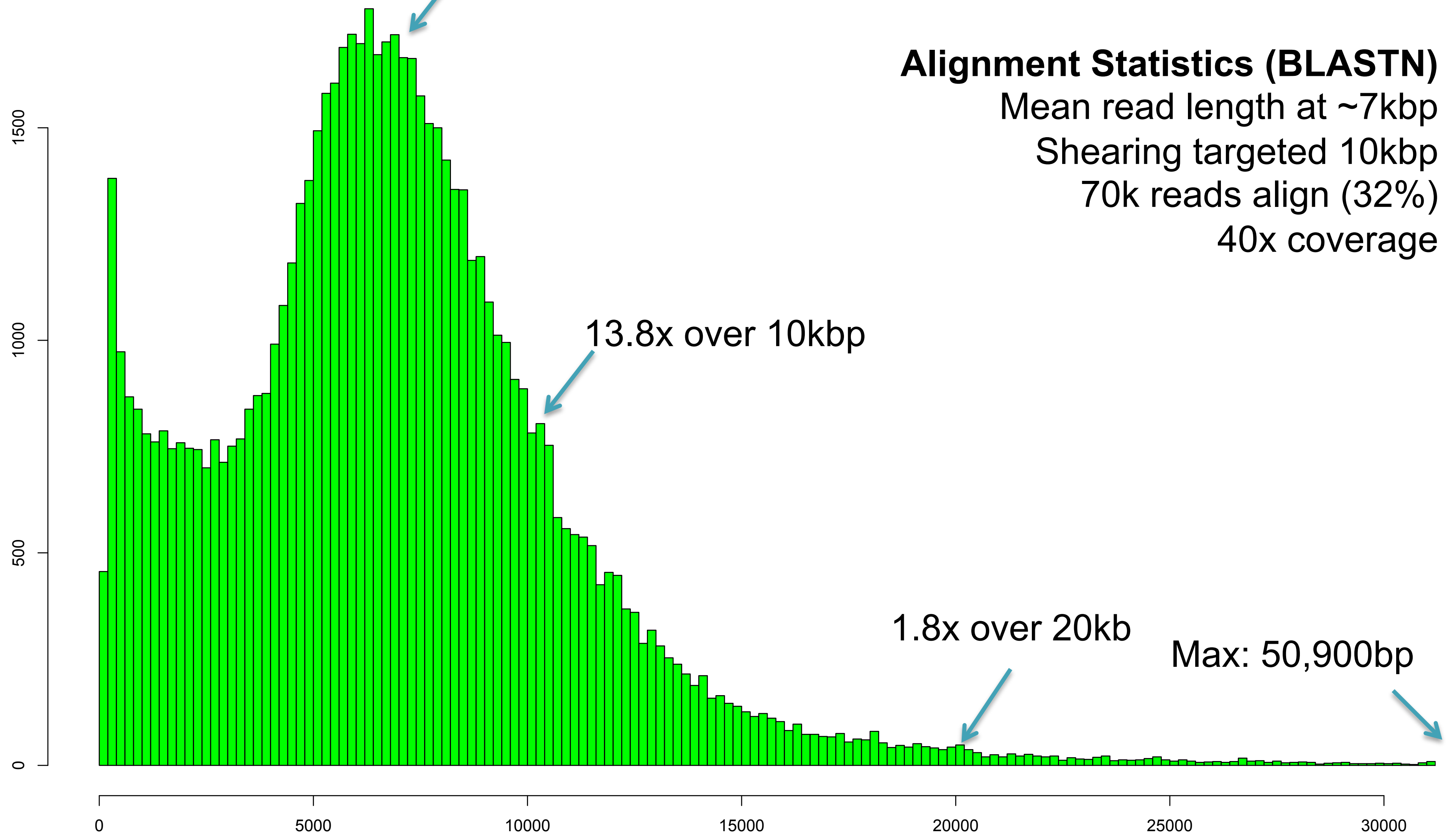
Mean flow cell: 50 Mbp in 2 days

Max flow cell: 446Mbp in 2 days

Nanopore Alignments



Mean: 6903bp



13.8x over 10kbp

1.8x over 20kb

Max: 50,900bp

Alignment Statistics (BLASTN)

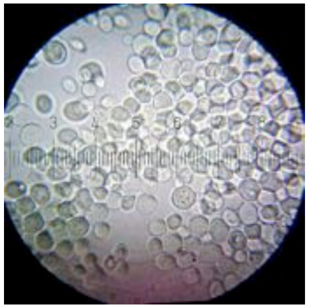
Mean read length at ~7kbp

Shearing targeted 10kbp

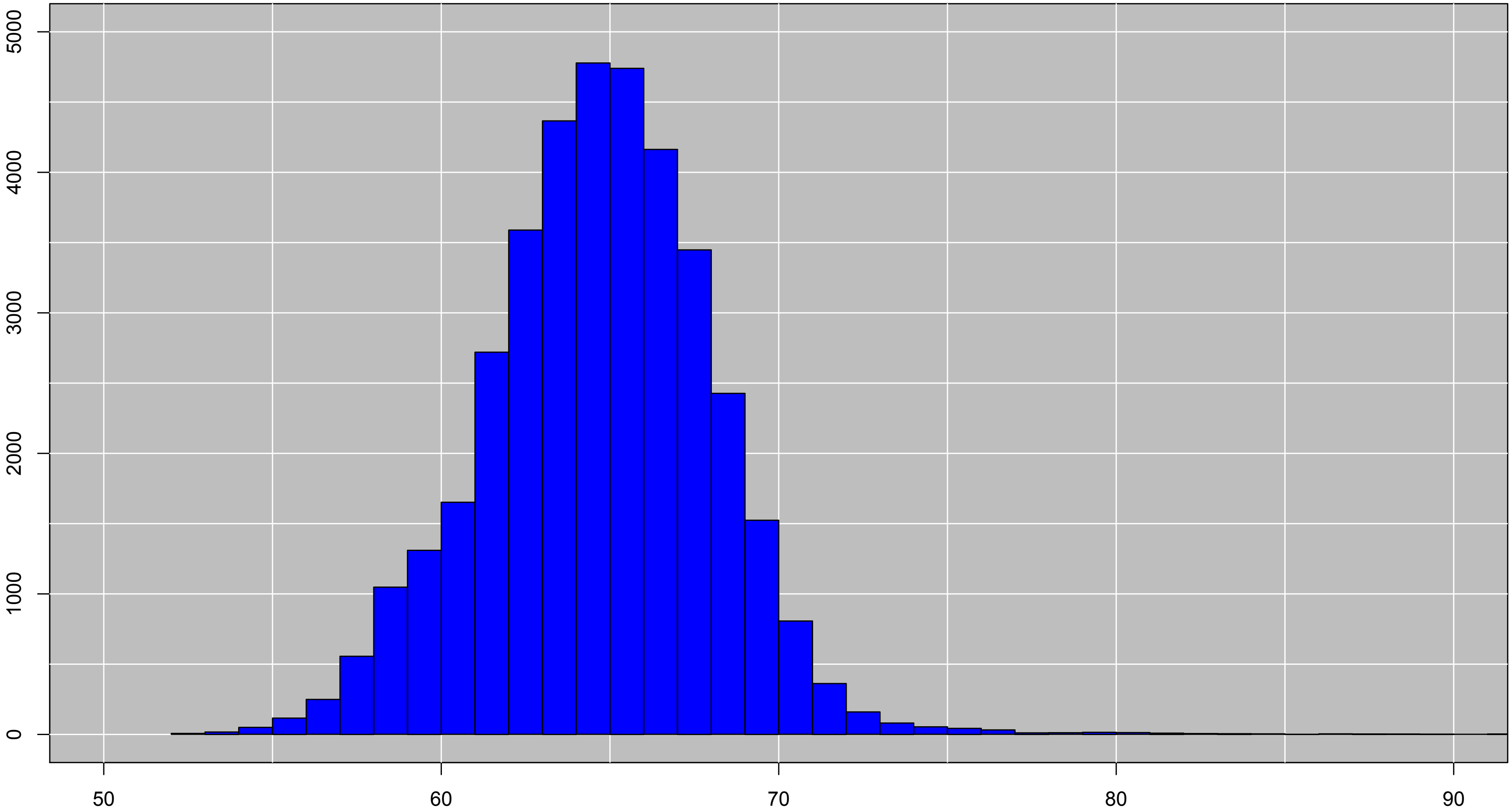
70k reads align (32%)

40x coverage

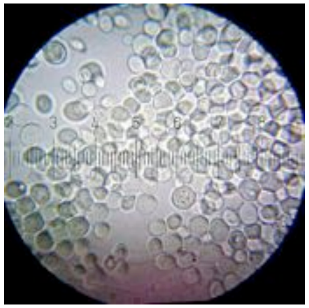
Nanopore Accuracy



Alignment Quality (BLASTN)
Of reads that align, average ~64% identity



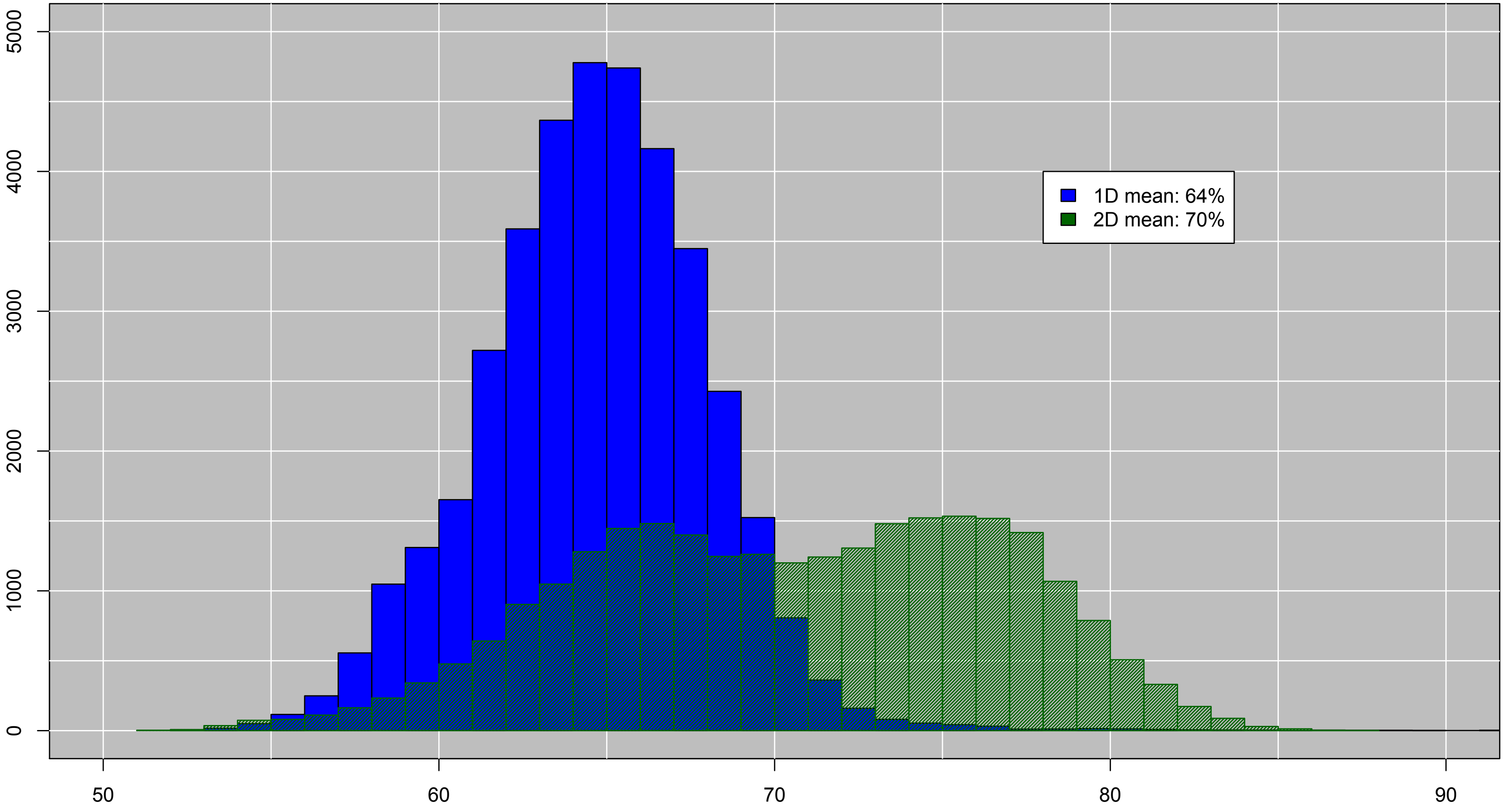
Nanopore Accuracy



Alignment Quality (BLASTN)

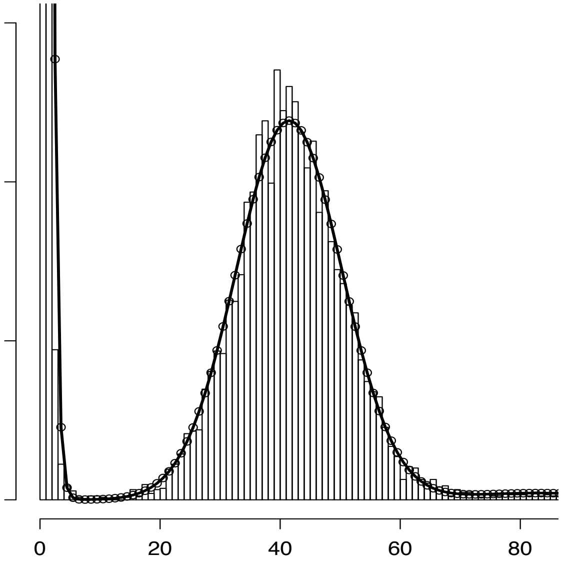
Of reads that align, average ~64% identity

“2D base-calling” improves to ~70% identity



Error Correction Methods

Quake

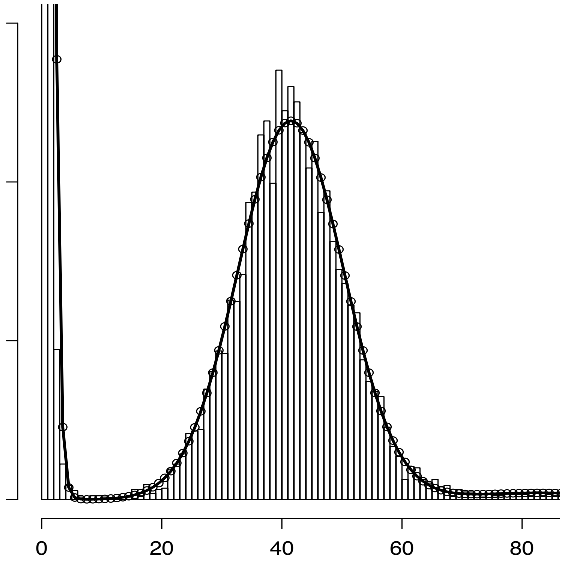


Word Analysis
of Illumina Reads

Kelly, Schatz, Salzberg (2010)
Genome Biology. 11:R116

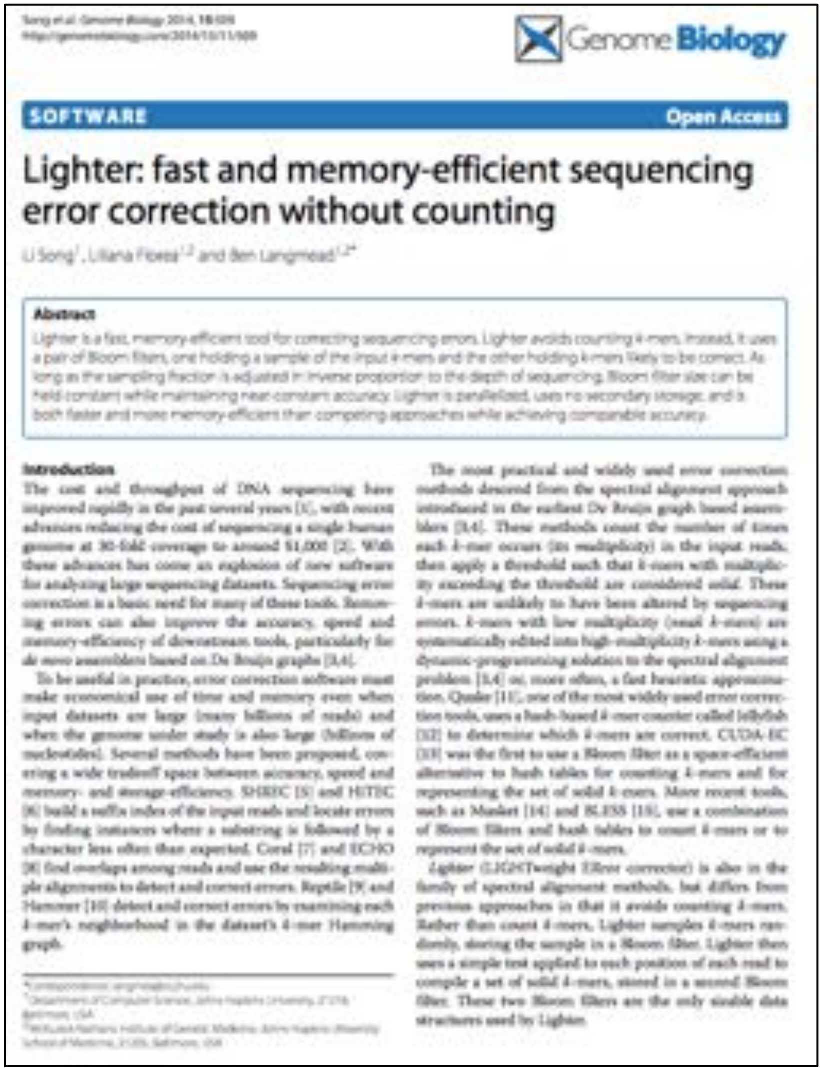
Error Correction Methods

Quake



Word Analysis of Illumina Reads

Kelly, Schatz, Salzberg (2010)
Genome Biology. 11:R116



Song et al. Genome Biology 2014, 15:509
http://genomebiology.com/2014/15/11/509

Genome Biology

SOFTWARE Open Access

Lighter: fast and memory-efficient sequencing error correction without counting

Li Song[1], Liliana Florea[1,2] and Ben Langmead[1,2*]

Abstract

Lighter is a fast, memory-efficient tool for correcting sequencing errors. Lighter avoids counting k-mers. Instead, it uses a pair of Bloom filters, one holding a sample of the input k-mers and the other holding k-mers likely to be correct. As long as the sampling fraction is adjusted in inverse proportion to the depth of sequencing, Bloom filter size can be held constant while maintaining near-constant accuracy. Lighter is parallelized, uses no secondary storage, and is both faster and more memory-efficient than competing approaches while achieving comparable accuracy.

Introduction

The cost and throughput of DNA sequencing have improved rapidly in the past several years [1], with recent advances reducing the cost of sequencing a single human genome at 30-fold coverage to around $1,000 [2]. With these advances has come an explosion of new software for analyzing large sequencing datasets. Sequencing error correction is a basic need for many of these tools. Removing errors can also improve the accuracy, speed and memory-efficiency of downstream tools, particularly for *de novo* assemblers based on De Bruijn graphs [3,4].

To be useful in practice, error correction software must make economical use of time and memory even when input datasets are large (many billions of reads) and when the genome under study is also large (billions of nucleotides). Several methods have been proposed, covering a wide tradeoff space between accuracy, speed and memory- and storage-efficiency. SHREC [5] and HiTEC [6] build a suffix index of the input reads and locate errors by finding instances where a substring is followed by a character less often than expected. Coral [7] and ECHO [8] find overlaps among reads and use the resulting multiple alignments to detect and correct errors. Reptile [9] and Hammer [10] detect and correct errors by examining each k-mer's neighborhood in the dataset's k-mer Hamming graph.

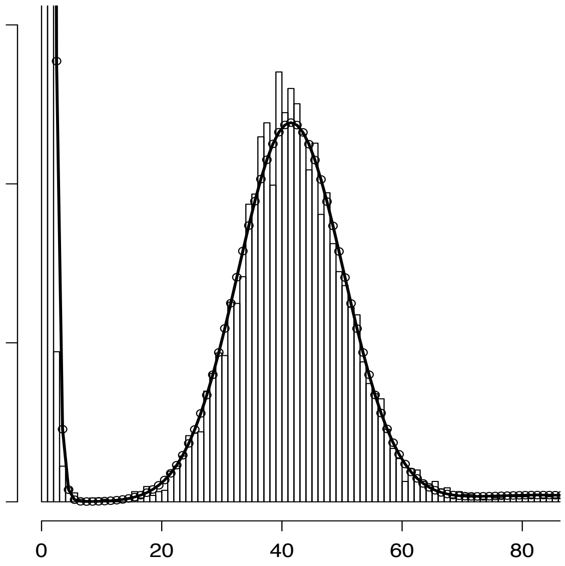
The most practical and widely used error correction methods descend from the spectral alignment approach introduced in the earliest De Bruijn graph based assemblers [3,4]. These methods count the number of times each k-mer occurs (its multiplicity) in the input reads, then apply a threshold such that k-mers with multiplicity exceeding the threshold are considered solid. These k-mers are unlikely to have been altered by sequencing errors. k-mers with low multiplicity (weak k-mers) are systematically edited into high-multiplicity k-mers using a dynamic-programming solution to the spectral alignment problem [3,4] or, more often, a fast heuristic approximation. Quake [11], one of the most widely used error correction tools, uses a hash-based k-mer counter called Jellyfish [12] to determine which k-mers are correct. CUDA-EC [13] was the first to use a Bloom filter as a space-efficient alternative to hash tables for counting k-mers and for representing the set of solid k-mers. More recent tools, such as Musket [14] and BLESS [15], use a combination of Bloom filters and hash tables to count k-mers or to represent the set of solid k-mers.

Lighter (LIGHTweight ERror corrector) is also in the family of spectral alignment methods, but differs from previous approaches in that it avoids counting k-mers. Rather than count k-mers, Lighter samples k-mers randomly, storing the sample in a Bloom filter. Lighter then uses a simple test applied to each position of each read to compile a set of solid k-mers, stored in a second Bloom filter. These two Bloom filters are the only sizable data structures used by Lighter.

*Correspondence: langmea@cs.jhu.edu
[1] Department of Computer Science, Johns Hopkins University, 21218 Baltimore, USA
[2] McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University School of Medicine, 21205, Baltimore, USA

Error Correction Methods

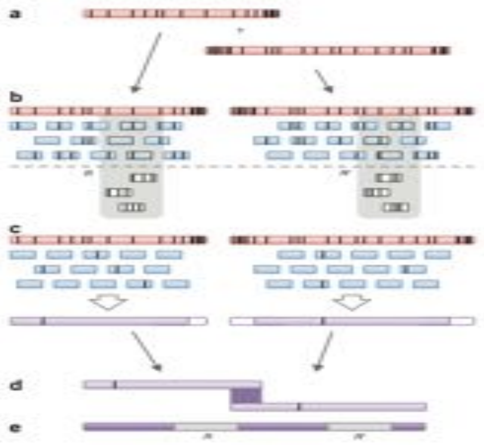
Quake



Word Analysis of Illumina Reads

Kelly, Schatz, Salzberg (2010)
Genome Biology. 11:R116

PacBioToCA & ECTools



Hybrid Correction Of PacBio using Illumina

Koren**,** Schatz, *et al* (2012)
Nature Biotechnology. 30:693–700

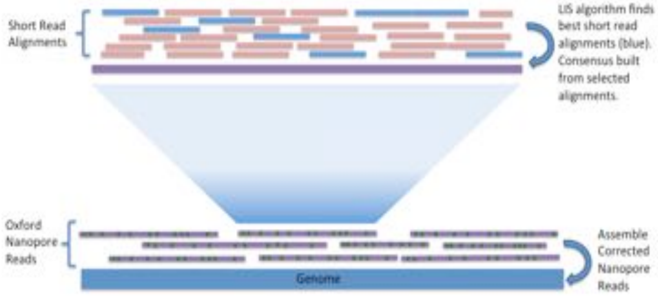
NanoCorr: Nanopore-Illumina Hybrid Error Correction

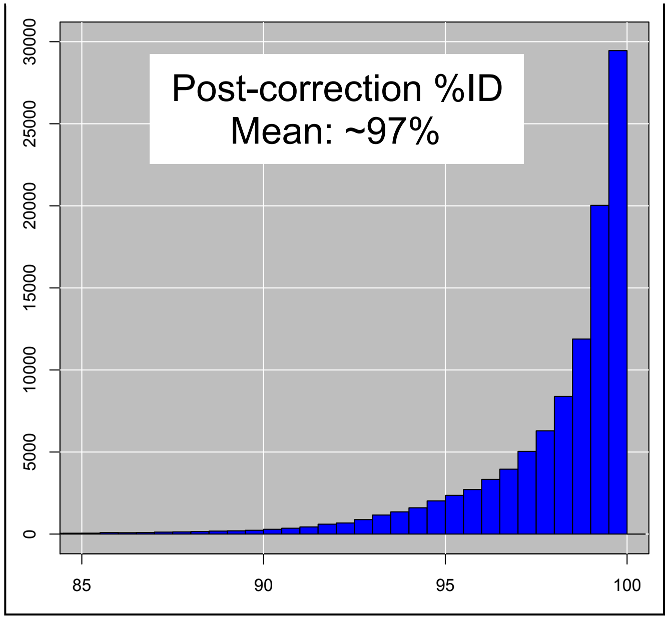
https://github.com/jgurtowski/nanocorr

1. BLAST Miseq reads to all raw Oxford Nanopore reads

2. Select non-repetitive alignments
   - ○ First pass scans to remove "contained" alignments
   - ○ Second pass uses Dynamic Programming (LIS) to select set of high-identity alignments with minimal overlaps

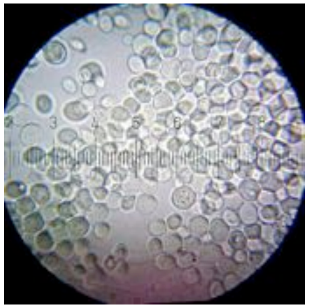
3. Compute consensus of each Oxford Nanopore read
   - ○ State machine of most commonly observed base at each position in read


Short Read Alignments
LIS algorithm finds best short read alignments (blue). Consensus built from selected alignments.
Oxford Nanopore Reads
Genome
Assemble Corrected Nanopore Reads



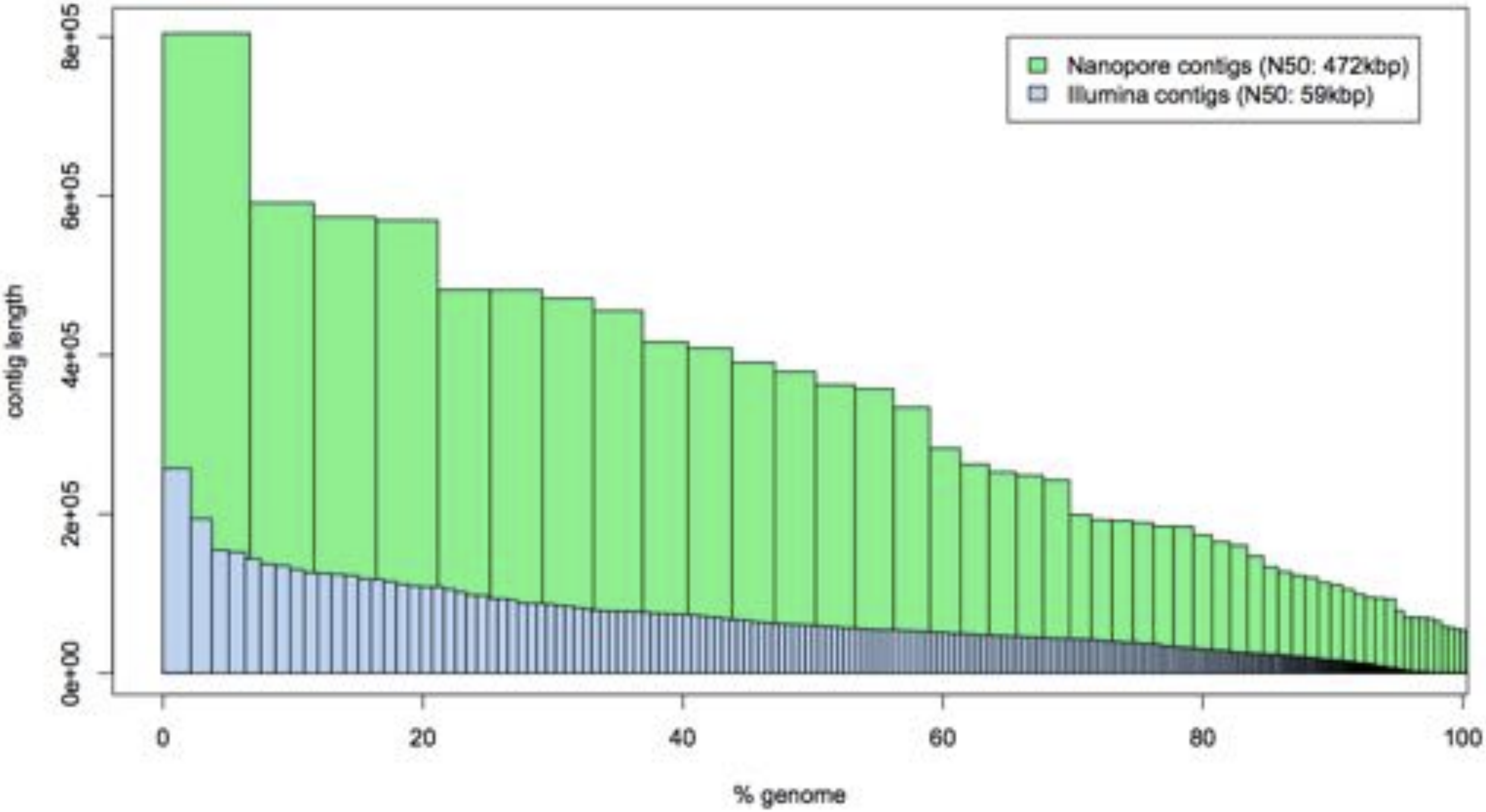
Oxford Nanopore Sequencing and de novo Assembly of a Eukaryotic Genome
Goodwin, S, Gurtowski, J *et al.* (2015) bioRxiv doi: http://dx.doi.org/10.1101/013490

Long Read Assembly



S288C Reference sequence

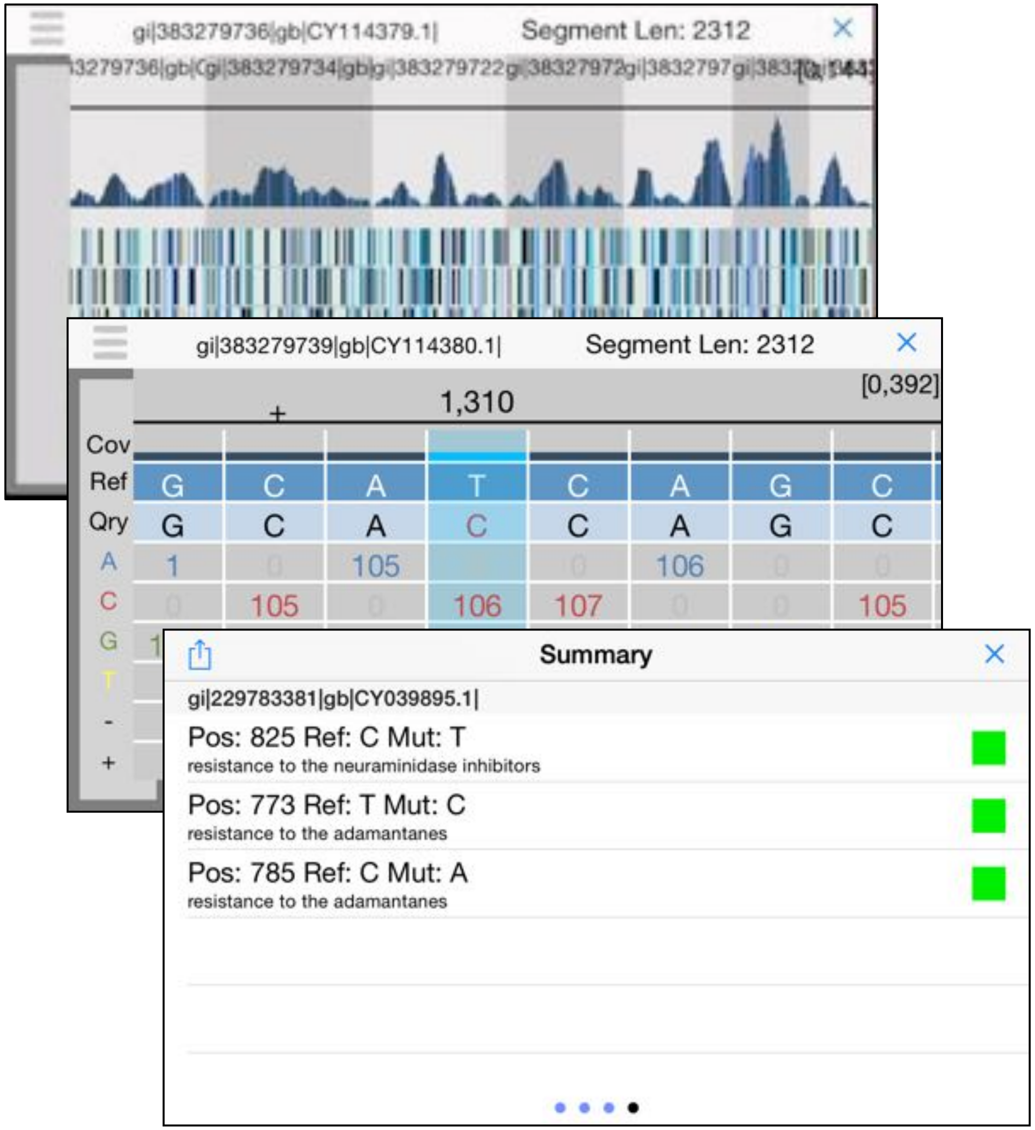
- 12.1Mbp; 16 chromo + mitochondria; N50: 924kbp



Genomic Futures?

iGenomics: Mobile Sequence Analysis

Aspyn Palatnick, Elodie Ghedin, Michael Schatz



gi|383279736|gb|CY114379.1|
Segment Len: 2312
3279736|gb|Cgi|383279734|gb|gi|383279722gi|38327972gi|3832797gi|3832

gi|383279739|gb|CY114380.1| Segment Len: 2312

1,310 [0,392]

Ref	G	C	A	T	C	A	G	C
Qry	G	C	A	C	C	A	G	C
A	1	0	105		0	106	0	0
C	0	105	0	106	107	0	0	105

Summary

gi|229783381|gb|CY039895.1|

- Pos: 825 Ref: C Mut: T 🟩
resistance to the neuraminidase inhibitors
- Pos: 773 Ref: T Mut: C 🟩
resistance to the adamantanes
- Pos: 785 Ref: C Mut: A 🟩
resistance to the adamantanes

The worlds first genomics analysis app for iOS devices

BWT + Dynamic Programming + UI

First application:

- Handheld diagnostics and therapeutic recommendations for influenza infections
- Coming soon to the App Store

Future applications

- Pathogen detection
- Food safety
- Biomarkers
- etc..

Genomics Graphs

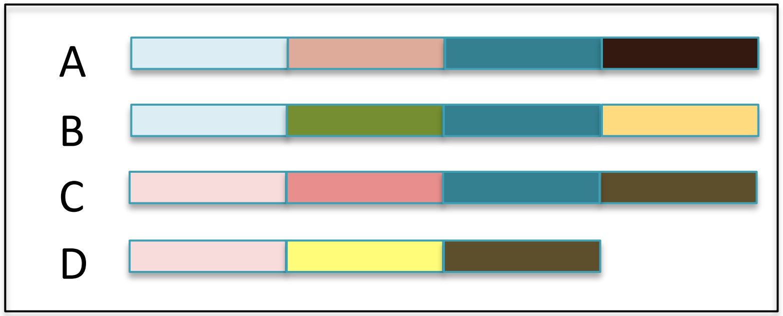
1. **Error Correction and Assembly**

   Long Read Single Molecule Sequencing

2. **Pan-Genomics**

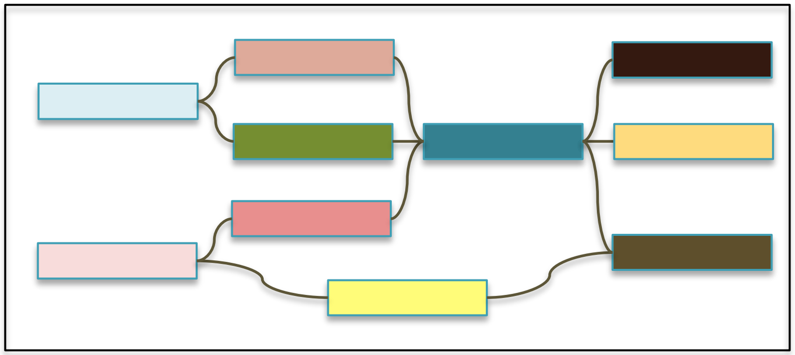
   Sequence conservation and divergence

Pan-Genome Alignment & Assembly



Time to start considering problems for which N complete genomes is the input to study the "pan-genome"

- Available today for many microbial species, near future for higher eukaryotes



Pan-genome colored de Bruijn graph

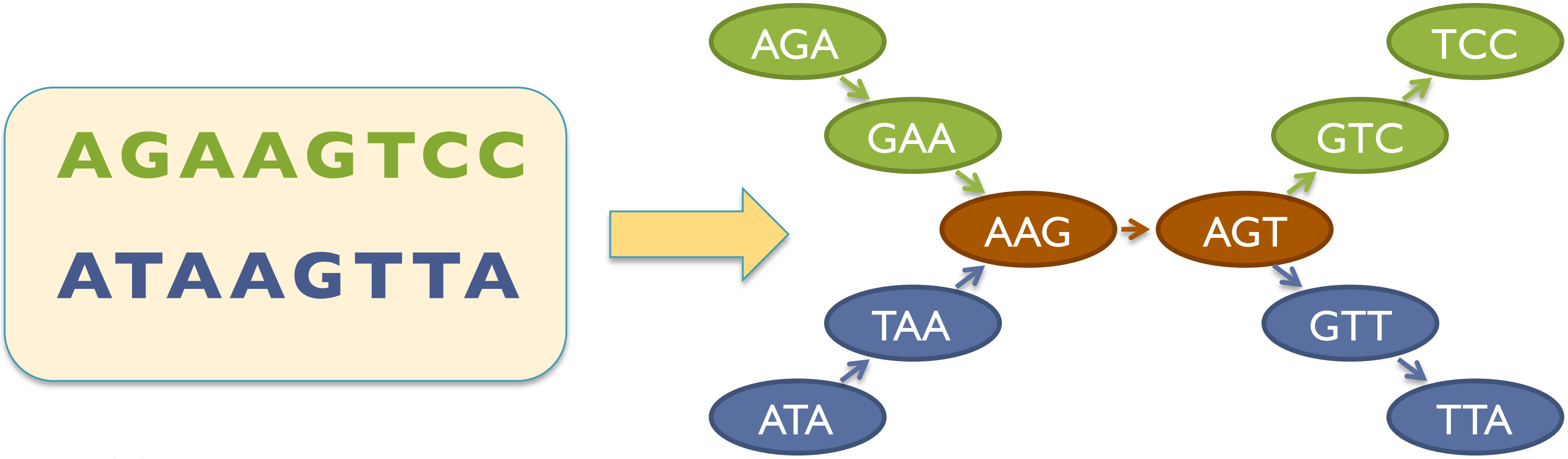
- Encodes all the sequence relationships between the genomes
- How well conserved is a given sequence?
- What are the pan-genome network properties?

SplitMEM: A graphical algorithm for pan-genome analysis with suffix skips
Marcus, S, Lee, H, Schatz, MC (2014) *Bioinformatics.* doi: 10.1093/bioinformatics/btu756

Graphical pan-genome analysis

Colored de Bruijn graph

- Node for each distinct kmer
- Directed edge connects consecutive kmers
- Nodes overlap by k-1 bp

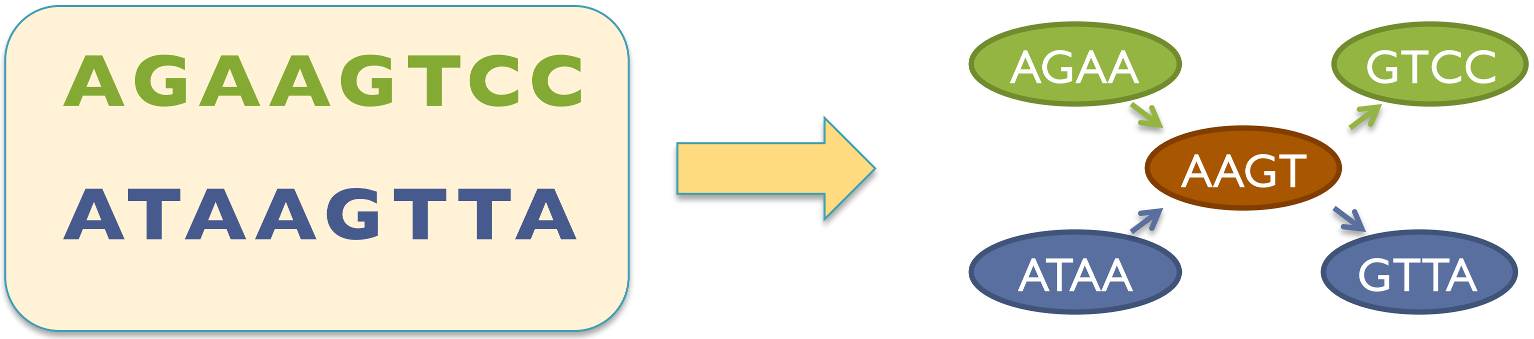


de Bruijn, 1946
Idury and Waterman, 1995
Pevzner, Tang, Waterman, 2001

Graphical pan-genome analysis

Colored de Bruijn graph

- Node for each distinct kmer
- Directed edge connects consecutive kmers
- Nodes overlap by k-1 bp



AGAAGTCC

ATAAGTTA

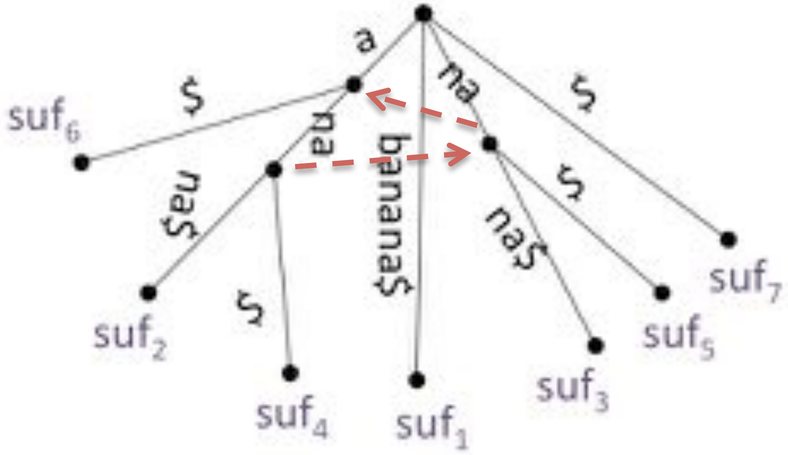
More specifically:

- We aim to build the *compressed* de Bruijn graph as quickly as possible without considering every distinct kmer

Suffix Trees

Elegant, widely used full text index

- Rooted, directed tree with a leaf corresponding to each suffix
- Path from root to leaf *i* spells suffix S[*i* . . . *n*].
- Each internal node has at least two distinct children except possibly the root
- Special *suffix links* navigate between internal nodes corresponding to consecutive substrings (xα -> α) without returning to root



S = banana$

Many important search problems can be solved in linear time and space

Linear pattern matching algorithms.
Weiner, P. (1973) *14th Annual IEEE Symposium on Switching and Automata Theory.*

On-line Construction of Suffix Trees
Ukkonen, E. (1995) *Algorithmica.*

Maximal Exact Matches (MEMs)

Definition:

A MEM is an exact match within a sequence that cannot be extended left or right without introducing a mismatch.

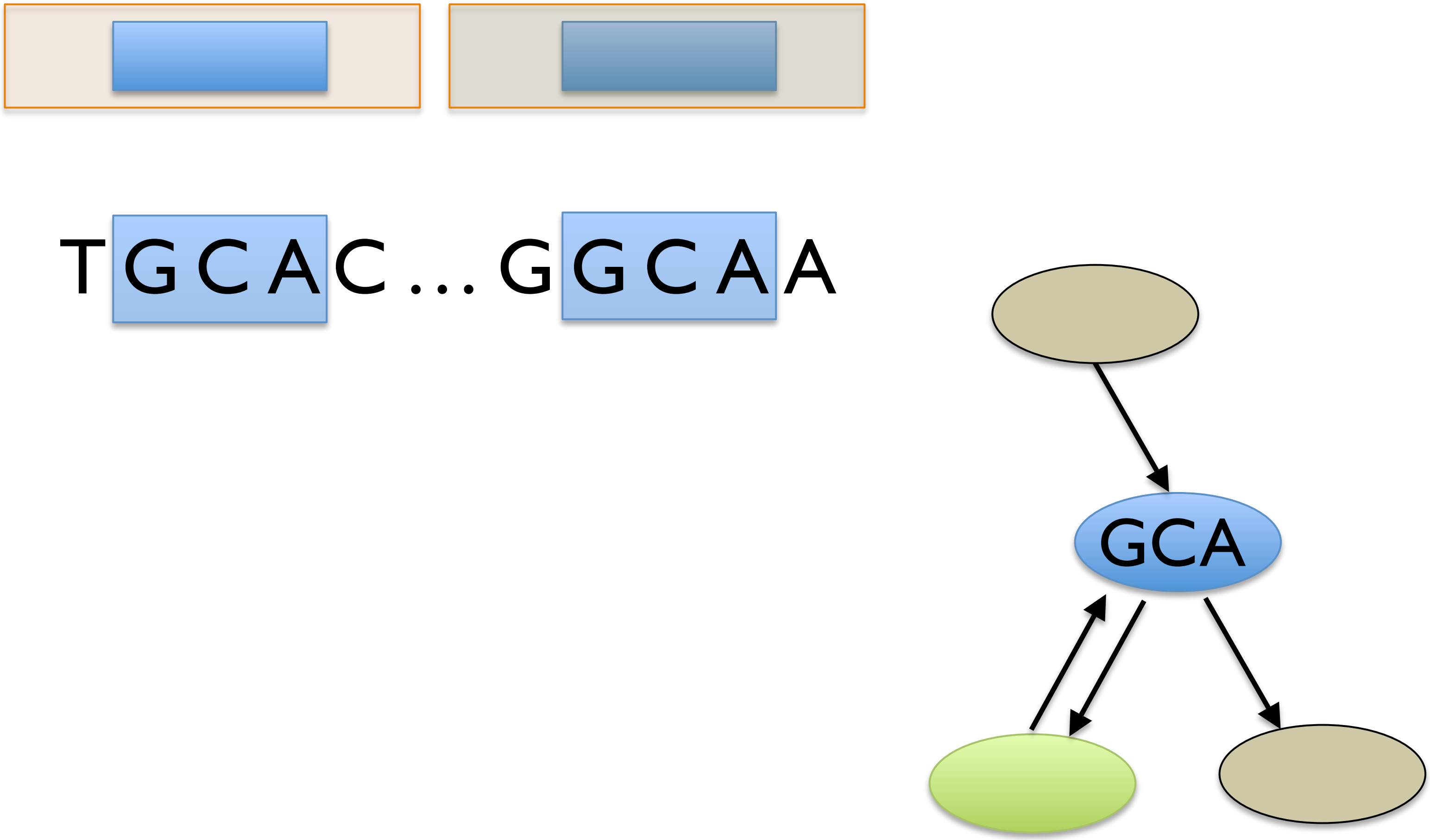
…XGATTACAW… …YGATTACAZ…

Key Properties:

- MEMs are internal nodes in the suffix tree that have left-diverse descendants.
- Have descendant leaves that represent suffixes with different characters preceding them

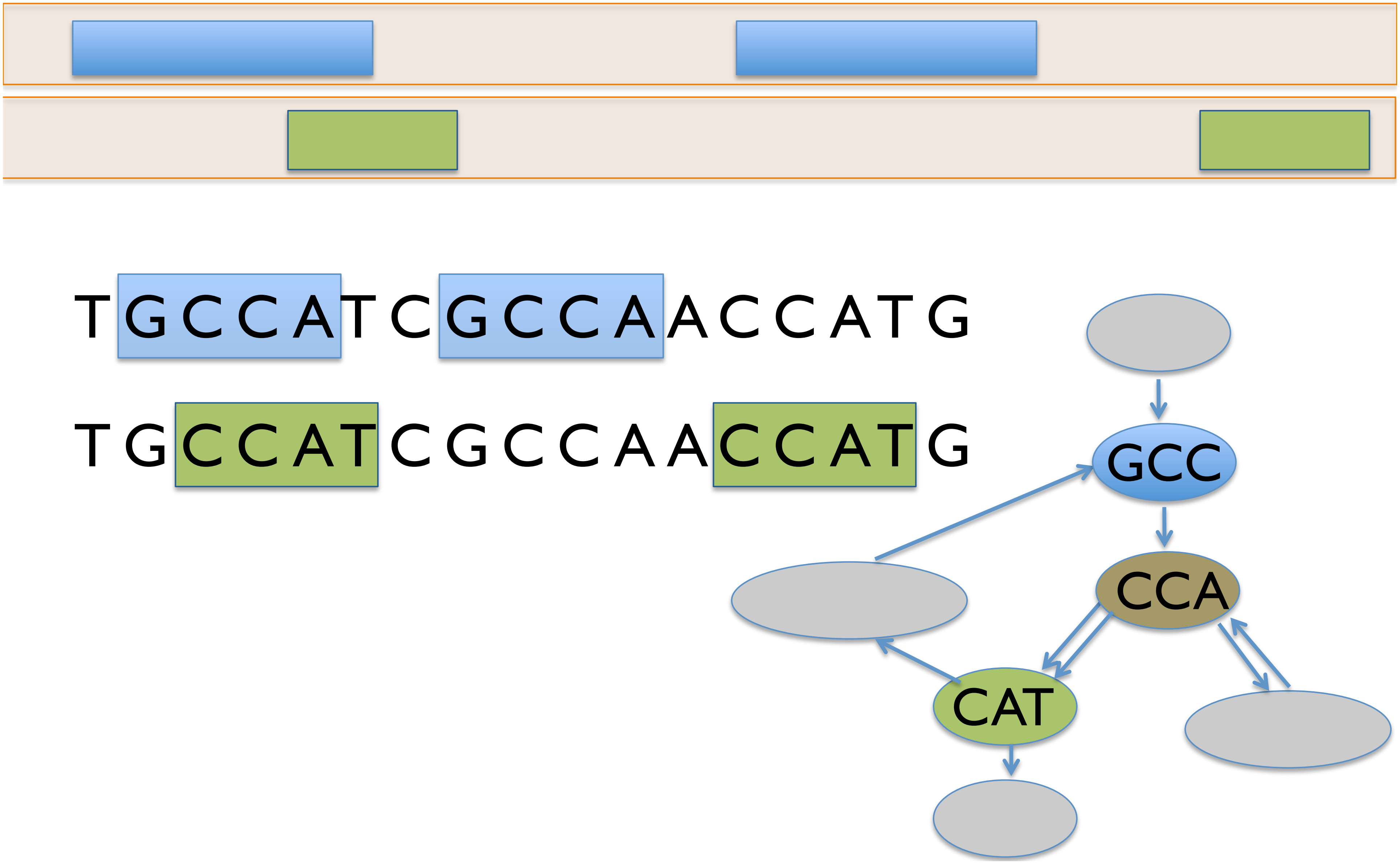
- ***Linear-time traversal of suffix tree to identify MEMs.***

MEMs to compressed de Bruijn Graphs



TGCAC…GGCAA

Overlapping MEMs



TGCCATCGCCAACCATG

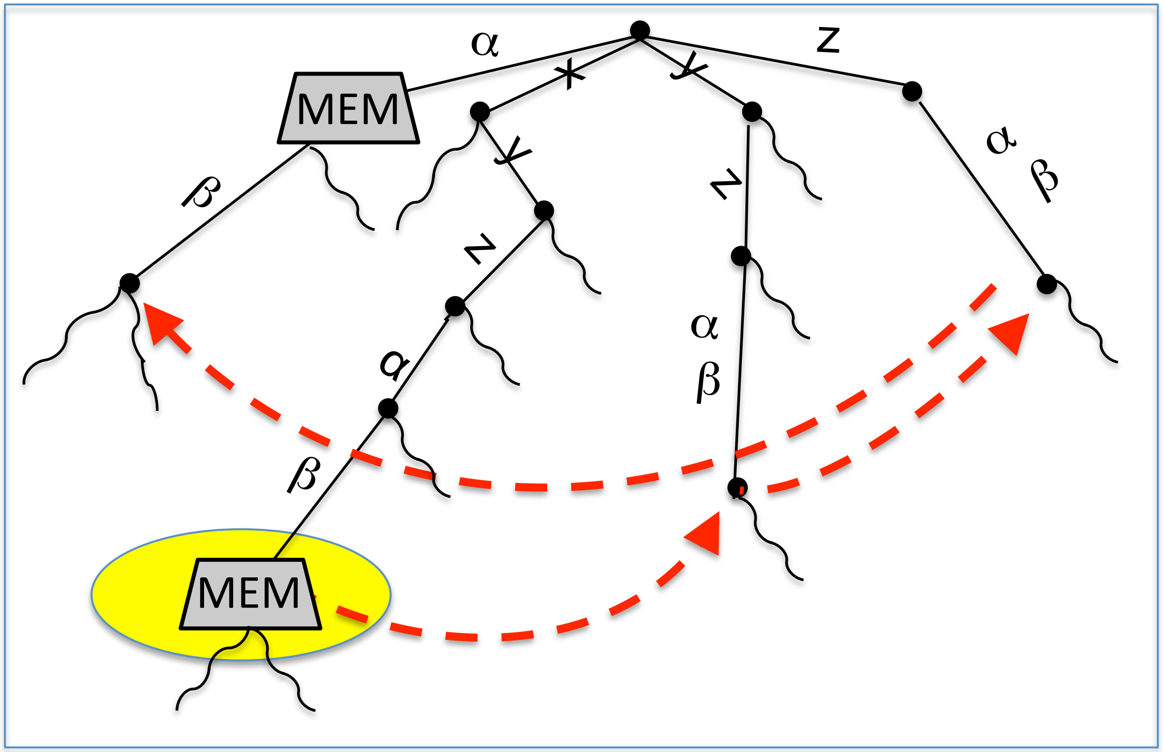
TGCCATCGCCAACCATG

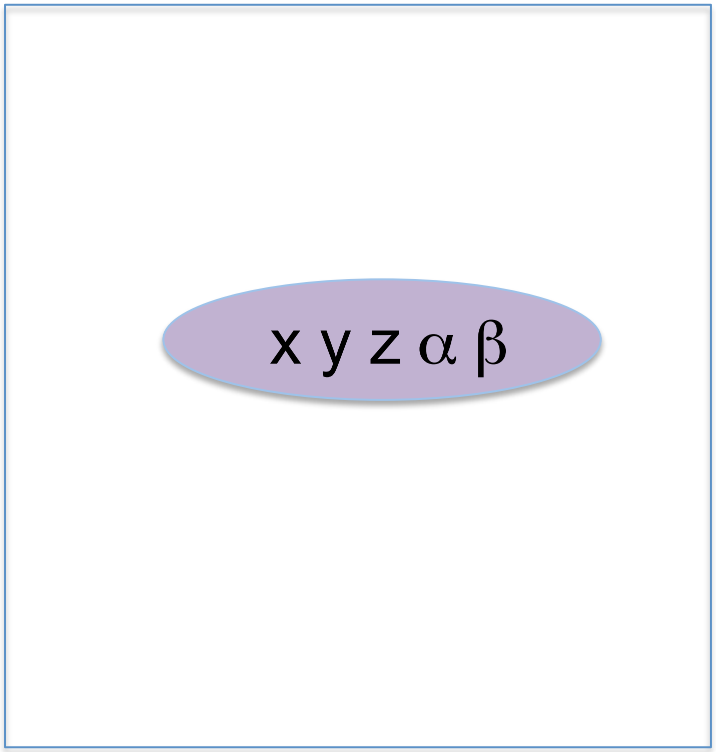
SplitMEM Sketch

1. Find nodes representing repeated sequences
   1. Build suffix tree of genome
   2. Mark internal nodes that are MEMs, length ≥ k
   3. Preprocess suffix tree for LMA queries
   4. Determine repeat-nodes of compressed de Bruijn graph by decomposing MEMs and extracting overlapping components, length ≥ k
2. Finalize graph with nodes and edges of unique sequences

Split MEMs to de Bruijn Graph

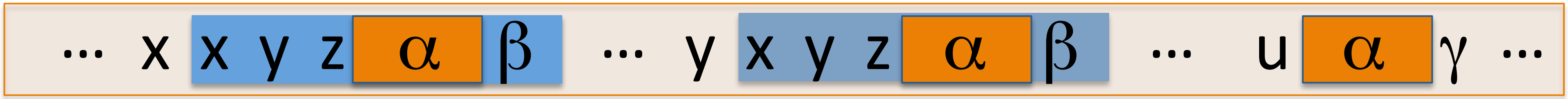
… x x y z α β … y x y z α β … u α γ …



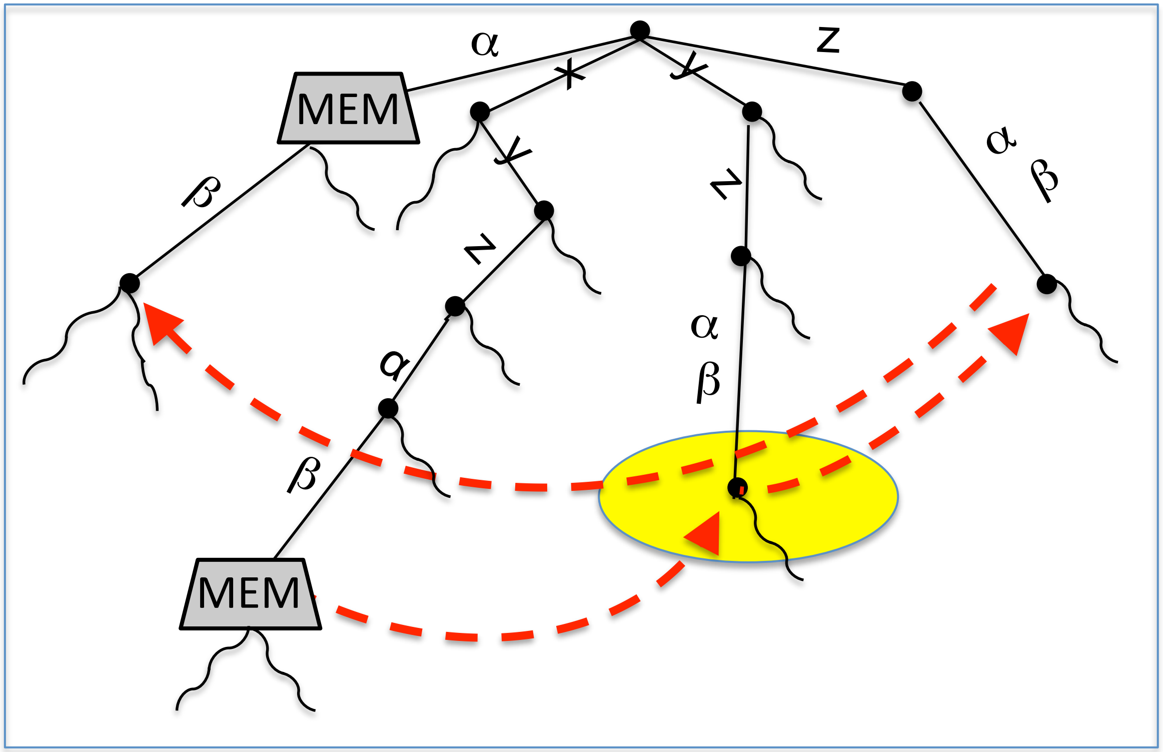


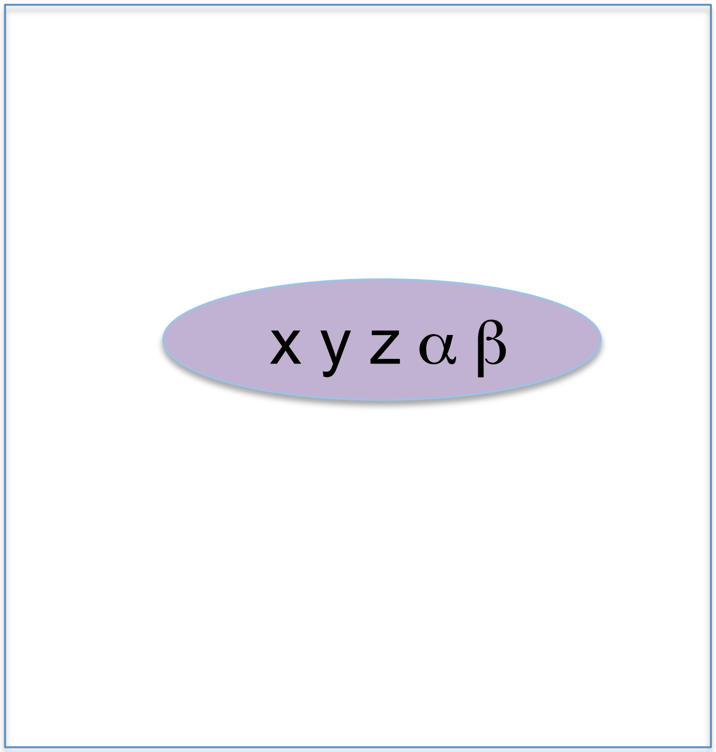
Find deepest MEM in suffix tree.

Split MEMs to de Bruijn Graph



… x x y z α β … y x y z α β … u α γ …



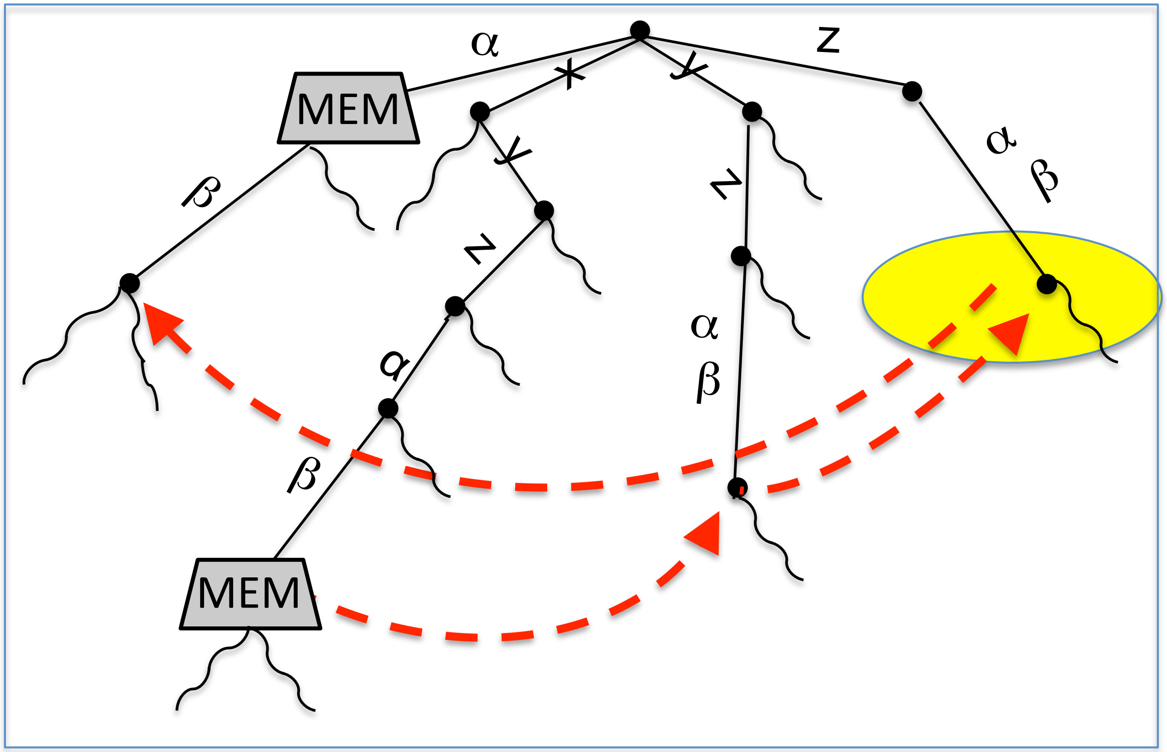


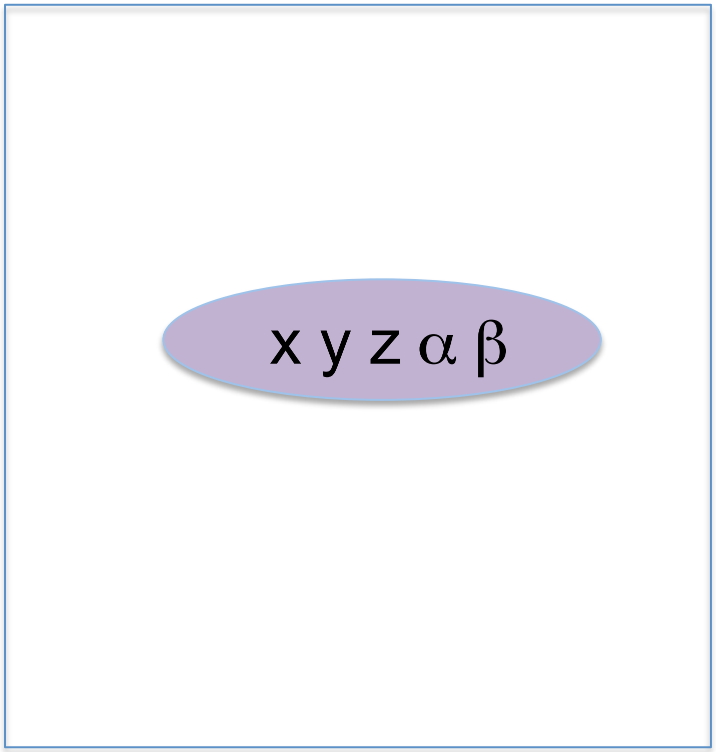
Traverse suffix link.

Look for MEM as ancestor.

Split MEMs to de Bruijn Graph

… x x y z α β … y x y z α β … u α γ …

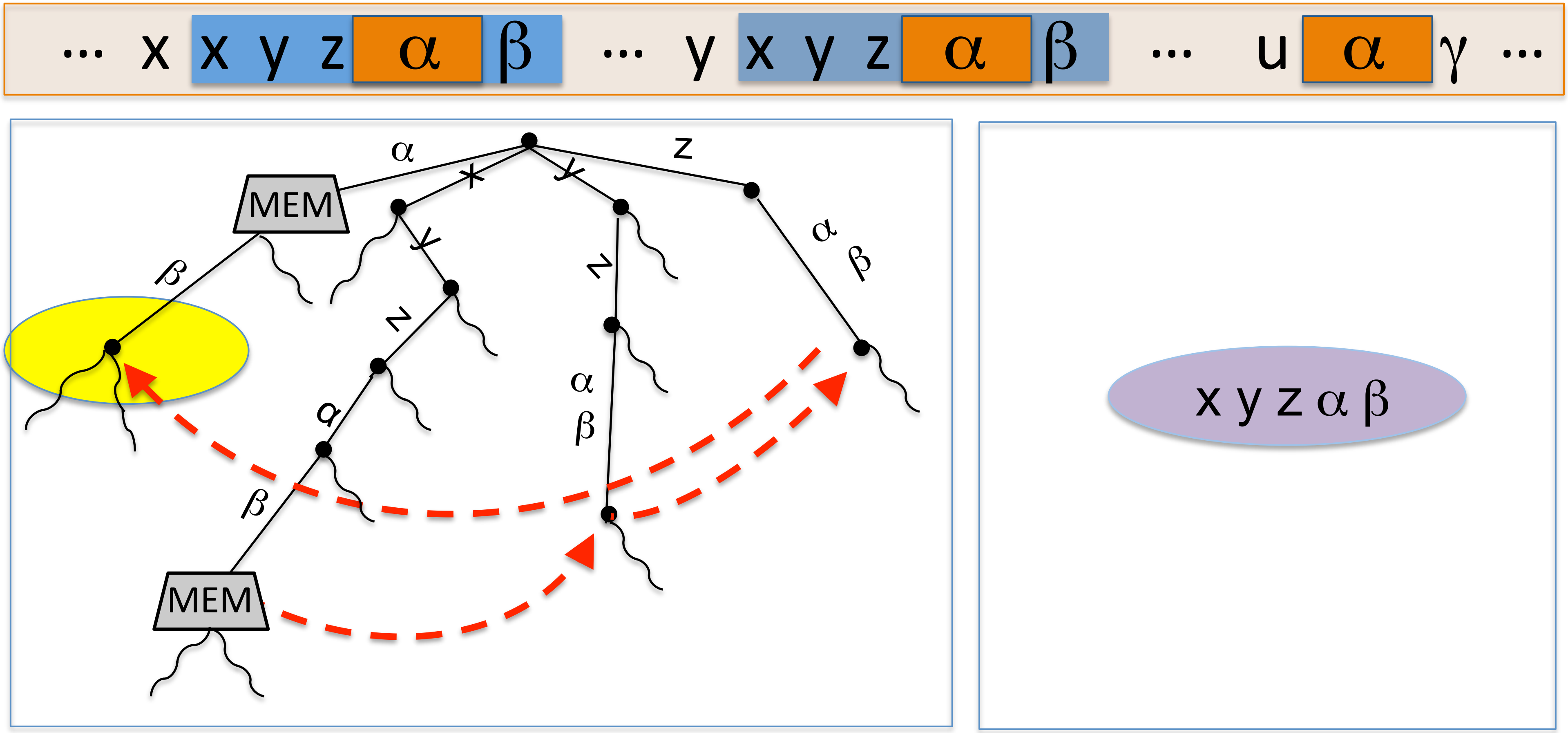




Traverse suffix link.

Look for MEM as ancestor.

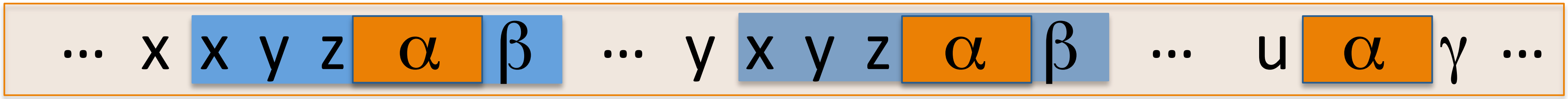
Split MEMs to de Bruijn Graph

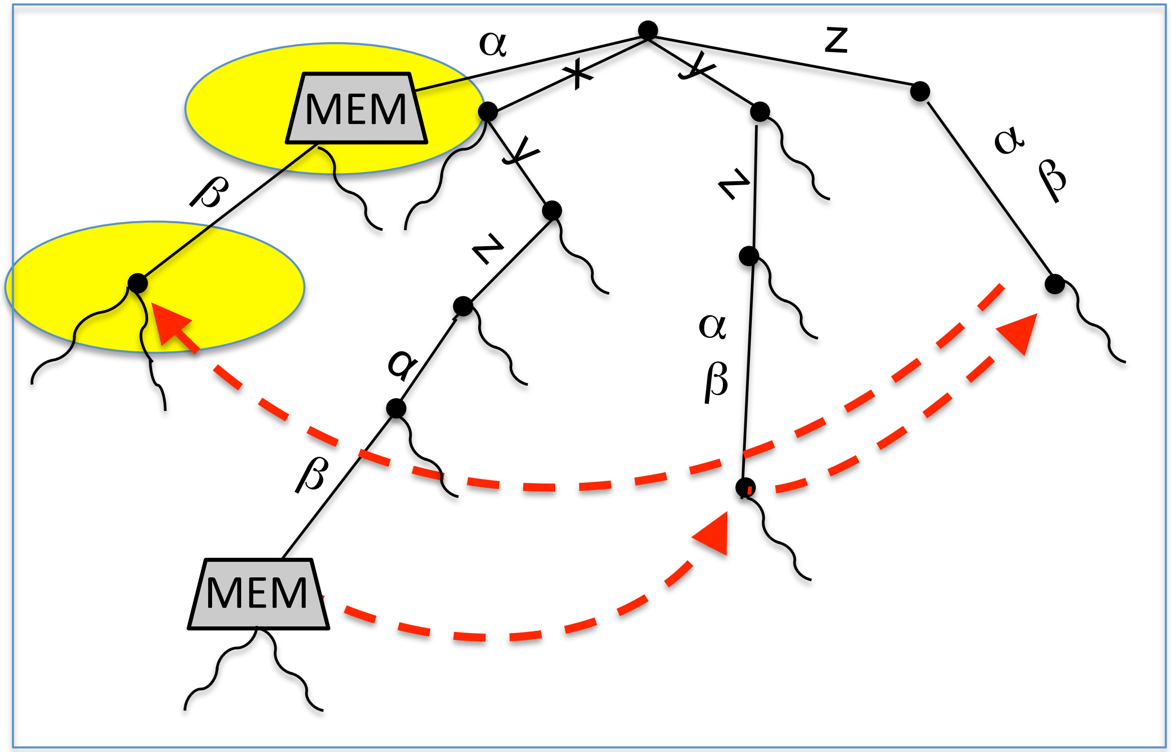


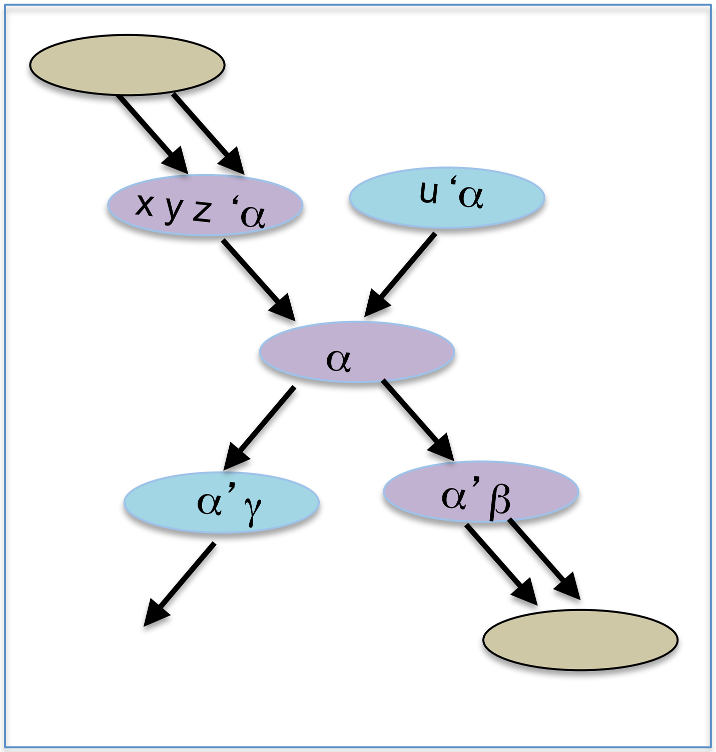
Traverse suffix link.

Look for MEM as ancestor.

Split MEMs to de Bruijn Graph





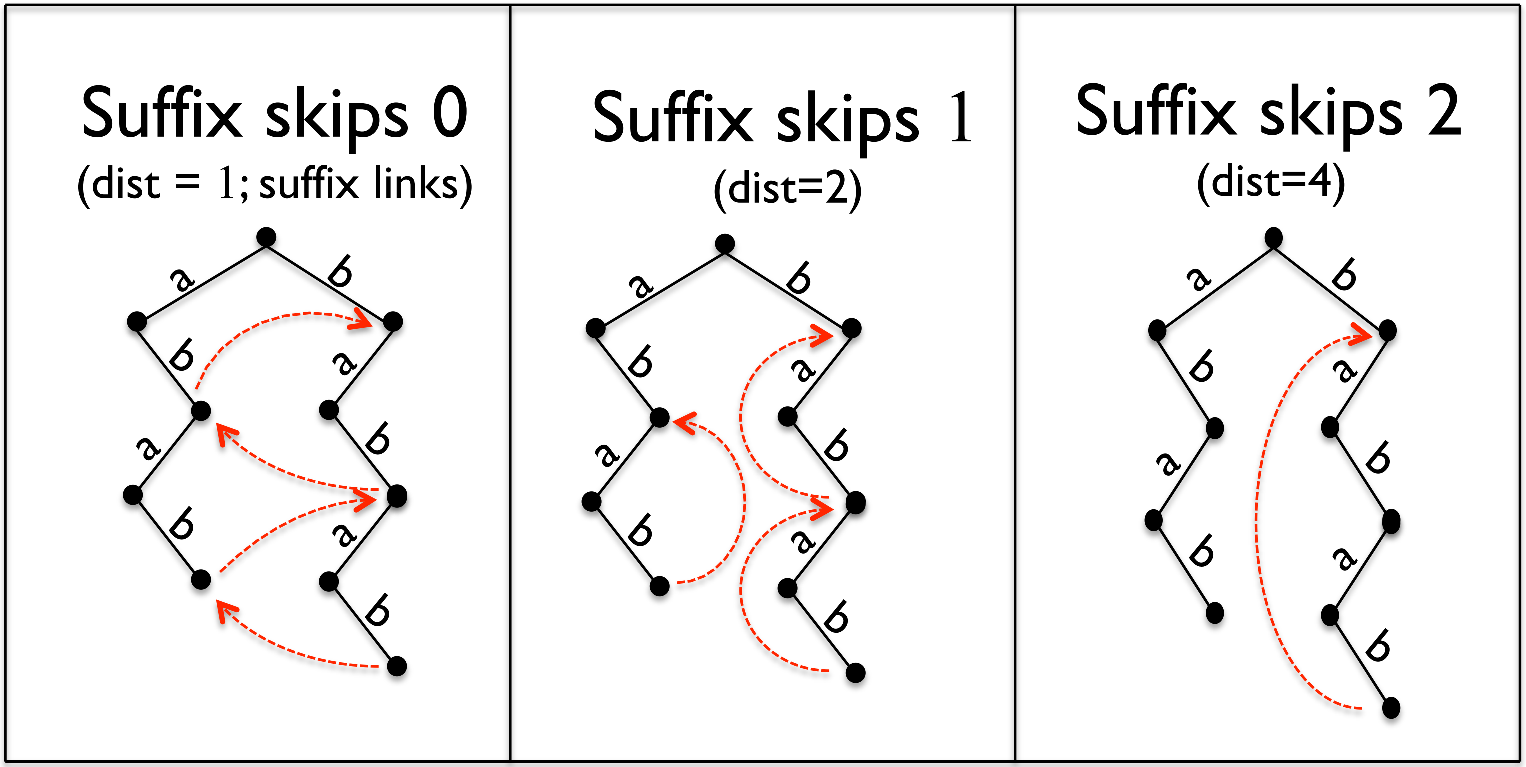


Found MEM as ancestor. Decompose.

Remove embedded MEM (suffix links). Find next embedded MEM.

Suffix Skips

Genome: babab



Suffix skips 0
(dist = 1; suffix links)

Suffix skips 1
(dist=2)

Suffix skips 2
(dist=4)

Skip c characters in log(c) steps instead of c suffix links

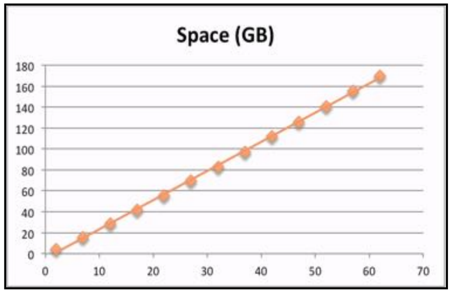
- Pointer jumping technique: n->ss[i] = n->ss[i-1]->ss[i-1]

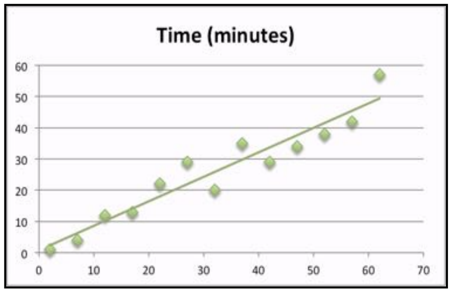
Microbial Pan-Genomes

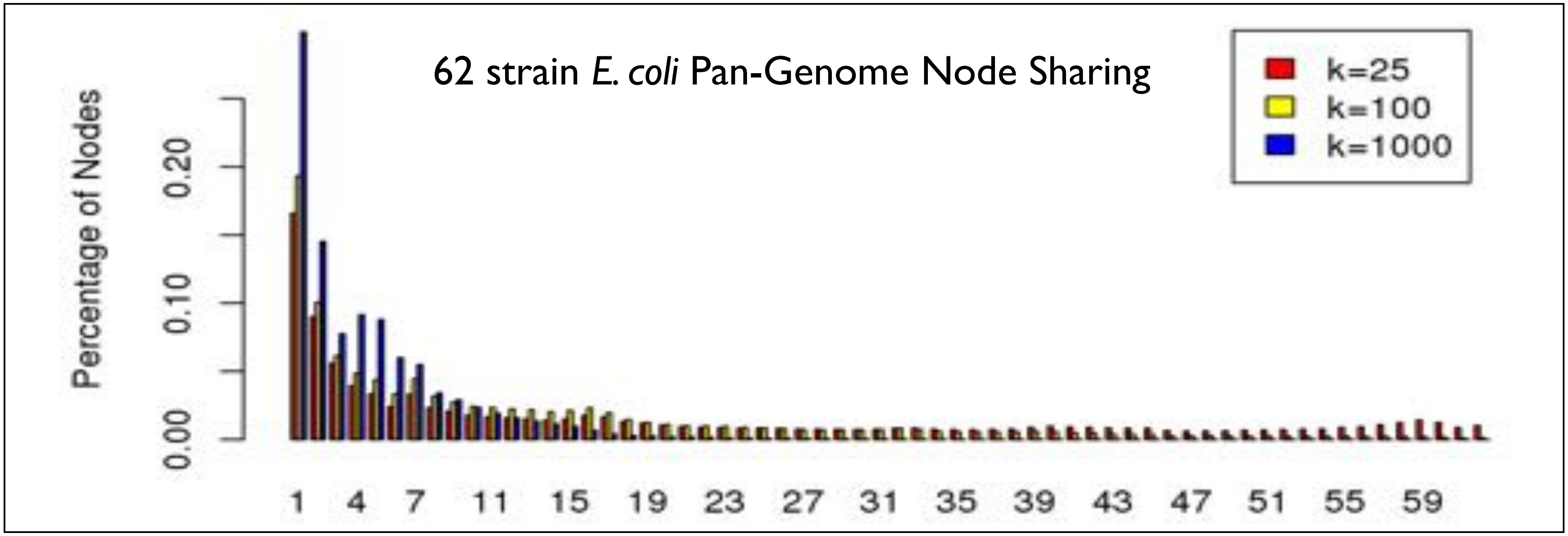
E. coli (62) and B. anthracis (9) pan-genome analysis

- Analyzed all available strains in Genbank
- Space is linear in the number of genomes
- Time is O(n log g) where g is the length of the longest genome
  - Linear time for most practical applications

- Many possible applications:
  - Identifying "core" genes present in all strains
  - Characterizing highly variable regions
  - Cataloging sequences shared by pathogenic varieties







The Rise of Pan-Genomics

Human Pan-Genomics

- We now have the capacity to consider the pan-genome structure of the human population and other high value species

- Already the current human reference genome has “alternate” sequence paths representing major differences between the different ethnicities (haplotype groups)

- However, virtually none of existing genomics algorithms operate on reference graphs, creating a major opportunity for research:
  - New and interesting CS problems
    - Online graph construction, searching, annotating, visualizing…
  - New and interesting biology
    - Detailed analysis of mutation, disease, and evolution

Extending reference assembly models
Church et al (2015) *Genome Biology.* 16:13 doi:10.1186/s13059-015-0587-3

Interfacing CS & Biology

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Theory & Programming Languages

- *How can we efficiently search & analyze genomic data?*
- *How do natural systems use abstraction or recursive processing?*

Systems

- *How do we scale to exascale or zettascale genomic data?*

Information Security

- *How do we balance the benefits of sharing genomic data with potential privacy abuses?*

Machine Learning & Data Intensive Computing

- *How do we learn from high dimensional biological data?*

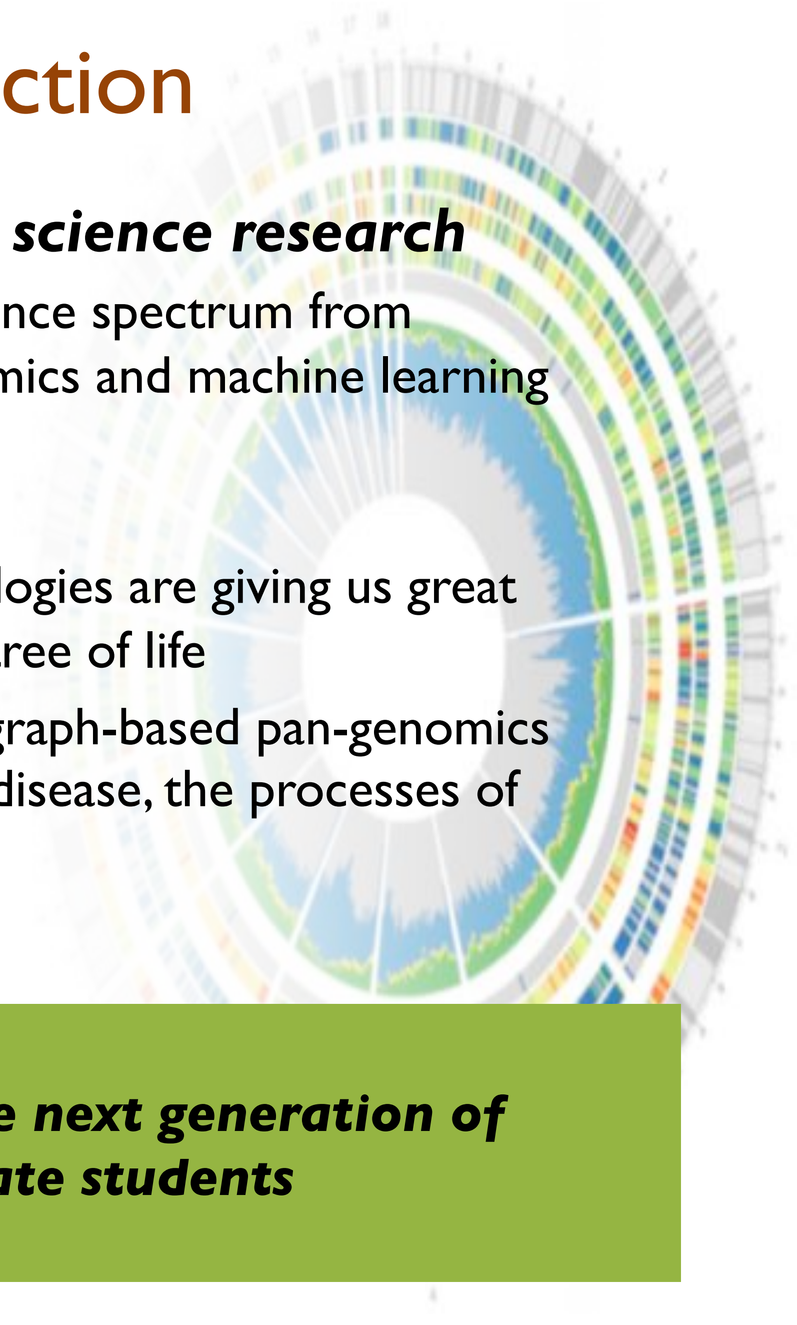
Language & Speech Processing

- *How do we recognize important features of sequences and other bio-molecular data?*

Robotics, Vision & Graphics

- *How do we integrate and model molecular with behavioral data?*

Understanding Genome Structure & Function



Genomics is a rich field for computer science research

- Opportunities across the entire data science spectrum from sensors & data systems, through algorithmics and machine learning

Sequencing Algorithmics

- Long reads and other sequencing technologies are giving us great power to look into genomes across the tree of life
- With these advances, expect the rise of graph-based pan-genomics giving us new insights into the origins of disease, the processes of development, and the forces of evolution

Also very interested in teaching the next generation of undergraduate and graduate students

Acknowledgements

Schatz Lab

Rahul Amin
Eric Biggers
Han Fang
Tyler Gavin
James Gurtowski
Ke Jiang
Hayan Lee
Zak Lemmon
Shoshana Marcus
Giuseppe Narzisi
Maria Nattestad
Aspyn Palatnick
Srividya Ramakrishnan
Rachel Sherman
Greg Vurture
Alejandro Wences

CSHL

Hannon Lab
Gingeras Lab
Jackson Lab
Hicks Lab
Iossifov Lab
Levy Lab
Lippman Lab
Lyon Lab
Martienssen Lab
McCombie Lab
Tuveson Lab
Ware Lab
Wigler Lab

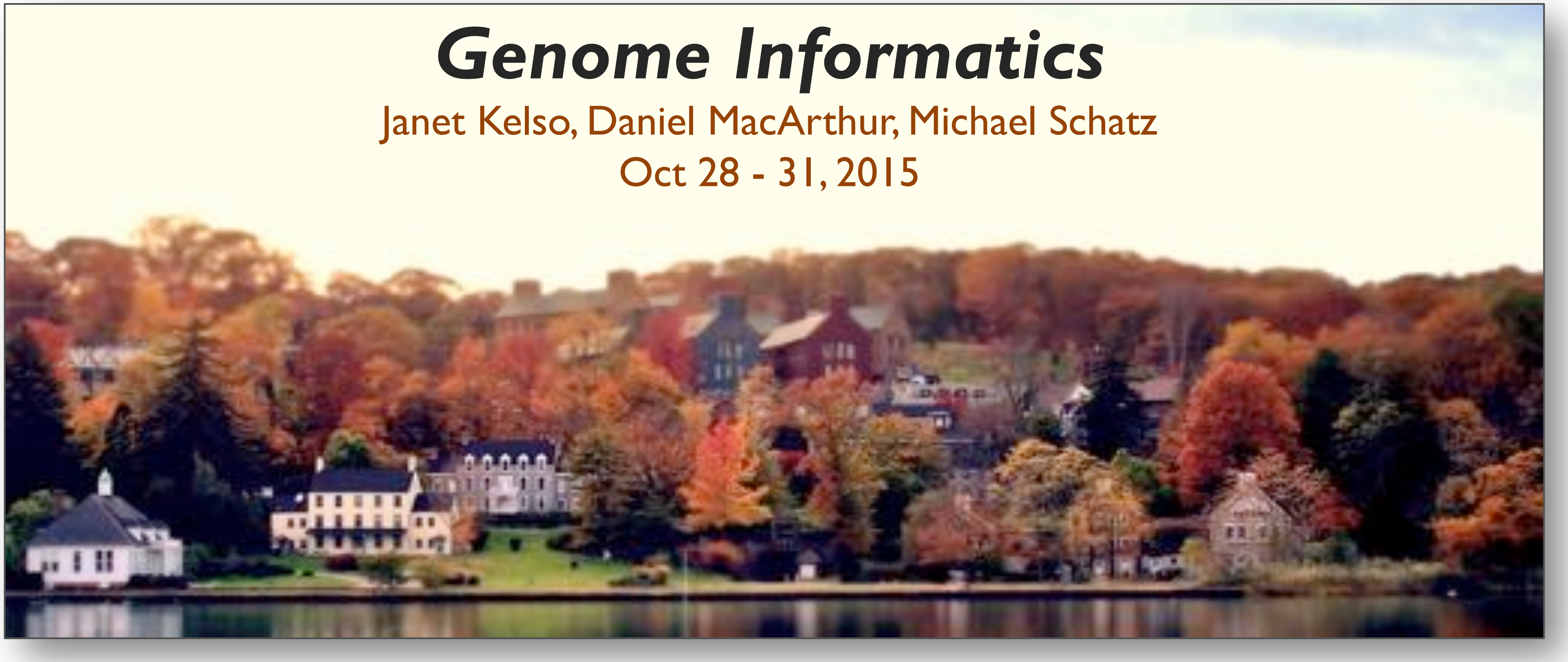
IT & Meetings Depts.
Pacific Biosciences
Oxford Nanopore

NSF

National Human Genome Research Institute

U.S. DEPARTMENT OF ENERGY

SFARI
SIMONS FOUNDATION AUTISM RESEARCH INITIATIVE



Genome Informatics

Janet Kelso, Daniel MacArthur, Michael Schatz

Oct 28 - 31, 2015

Thank you

http://schatzlab.cshl.edu

@mike_schatz