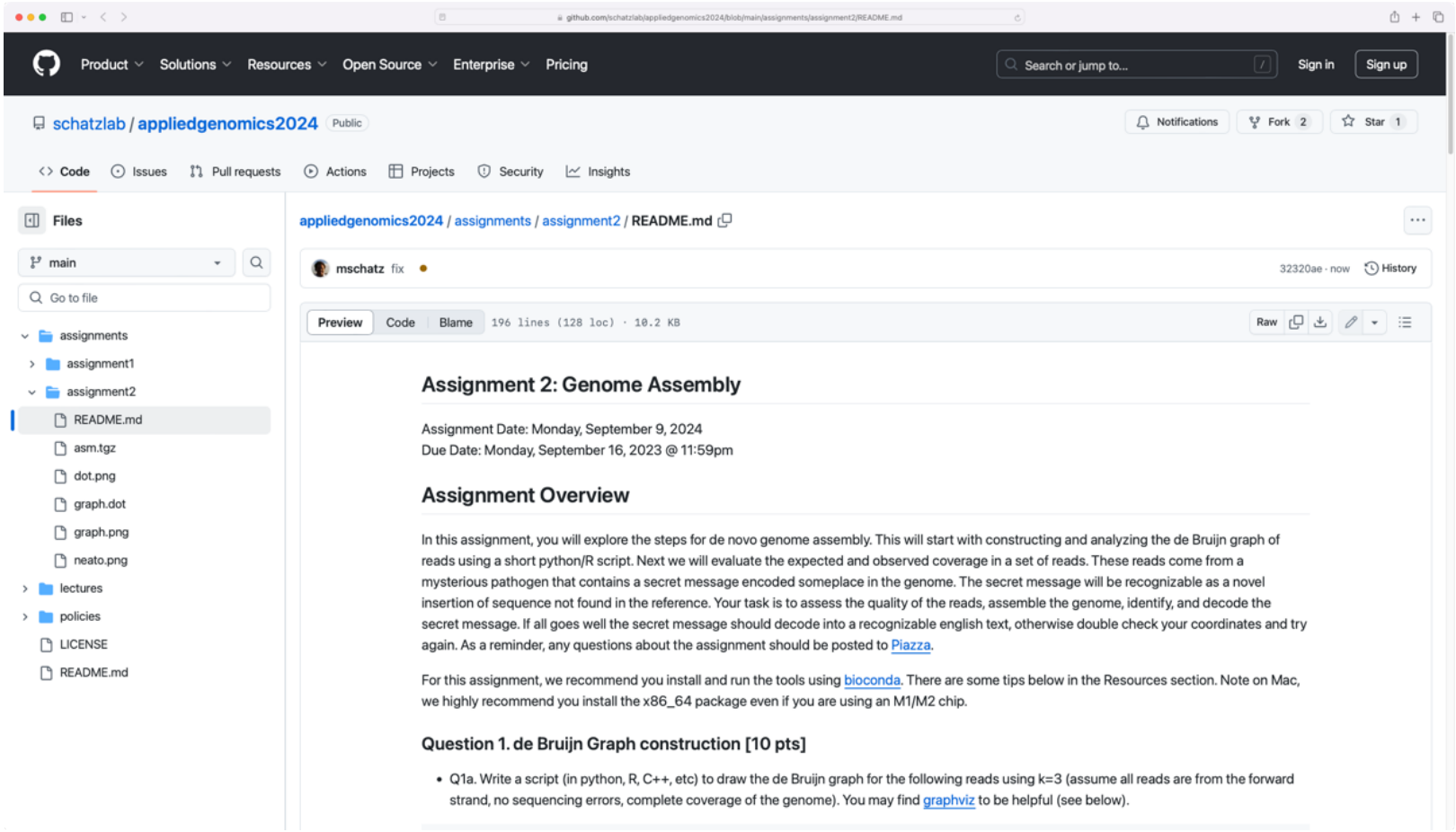
Read Mapping

Michael Schatz

Sept 16, 2024

Lecture 6: Applied Comparative Genomics

Assignment 2: Genome Assembly
Due Monday Sept 16 by 11:59pm



https://github.com/schatzlab/appliedgenomics2024/tree/main/assignments/assignment2

Check Piazza for questions!

The Sequence of the Human Genome
Venter et al.
Science 291. pp 1304-1351 (2001)

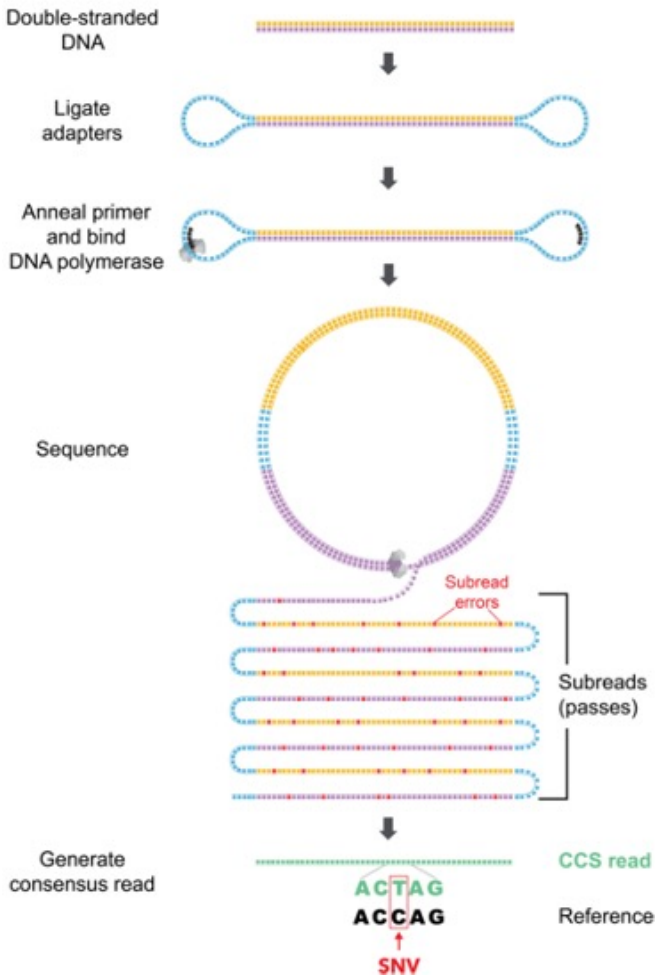
Initial sequencing and analysis of the human genome
International Human Genome Sequencing Consortium
Nature 409, pp 860–921 (2001)

“HiFi” Circular Consensus Reads

High-quality reads produced by sequencing the same molecule multiple times

Higher accuracy for low-coverage sequences like somatic variants or lowly expressed transcripts in RNA-seq, more interpretable alignments, better & faster assembly

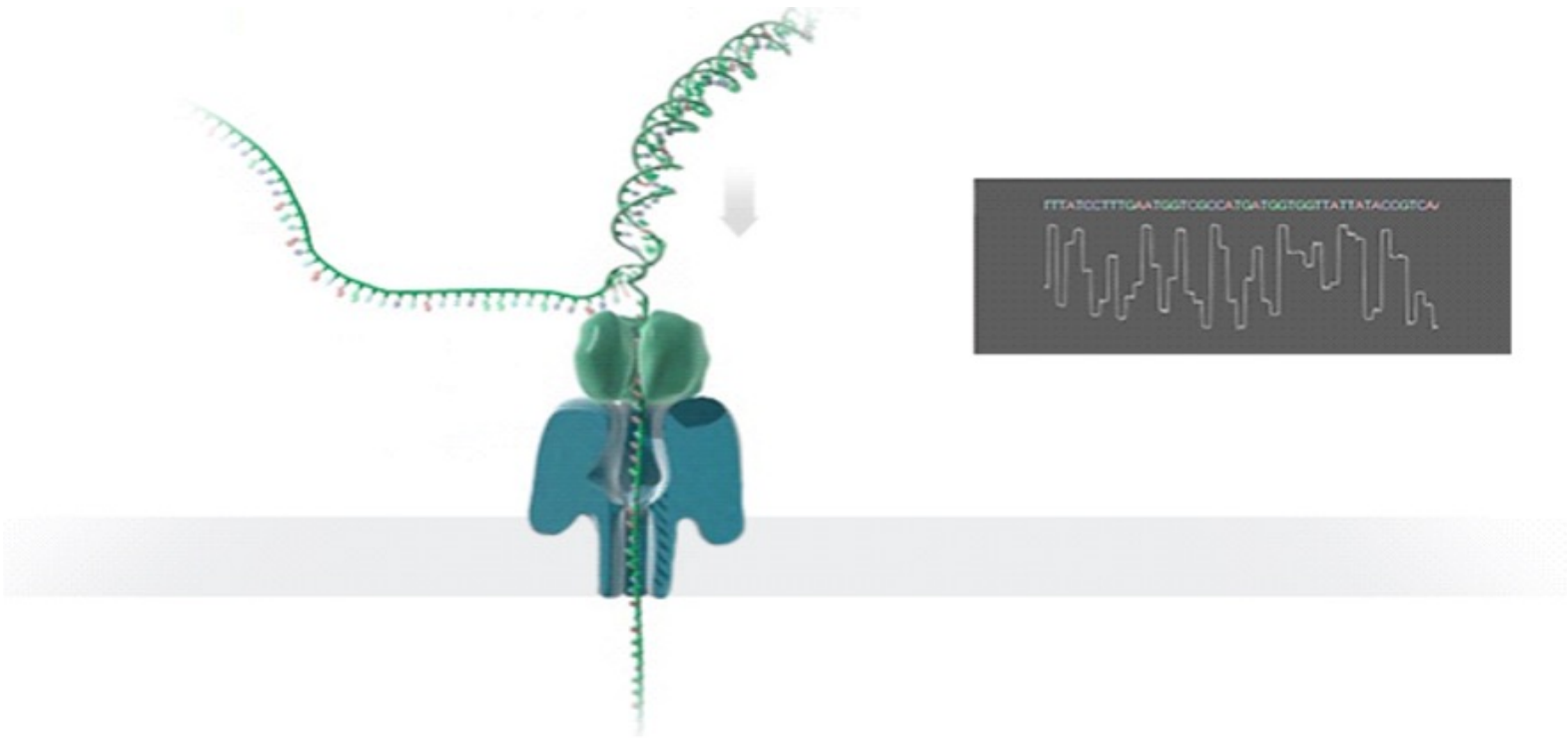
Limits read length, used to be very expensive but more manageable now



Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome
Wenger et al (2019) Nature Biotechnology doi:10.1038/s41587-019-0217-9

Nanopore Sequencing

Sequences DNA/RNA by measuring changes in ionic current as nucleotide strand passes through a pore

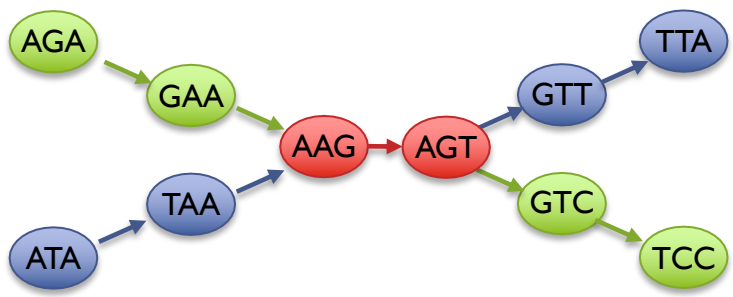


https://www.youtube.com/watch?v=RcP85JHLmnI

nanoporetech.com/applications/dna-nanopore-sequencing

Two Paradigms for Assembly

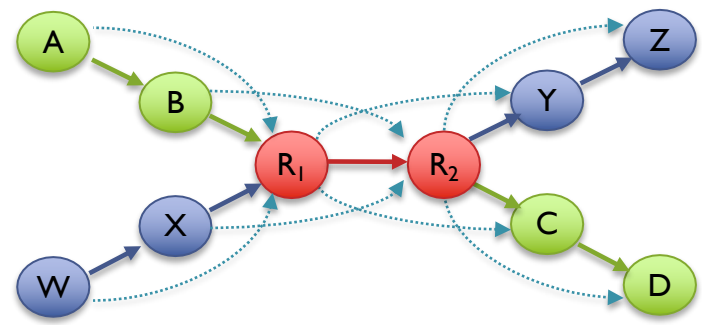
de Bruijn Graph



Short read assemblers

- Repeats depends on word length
- Read coherency, placements lost
- Robust to high coverage

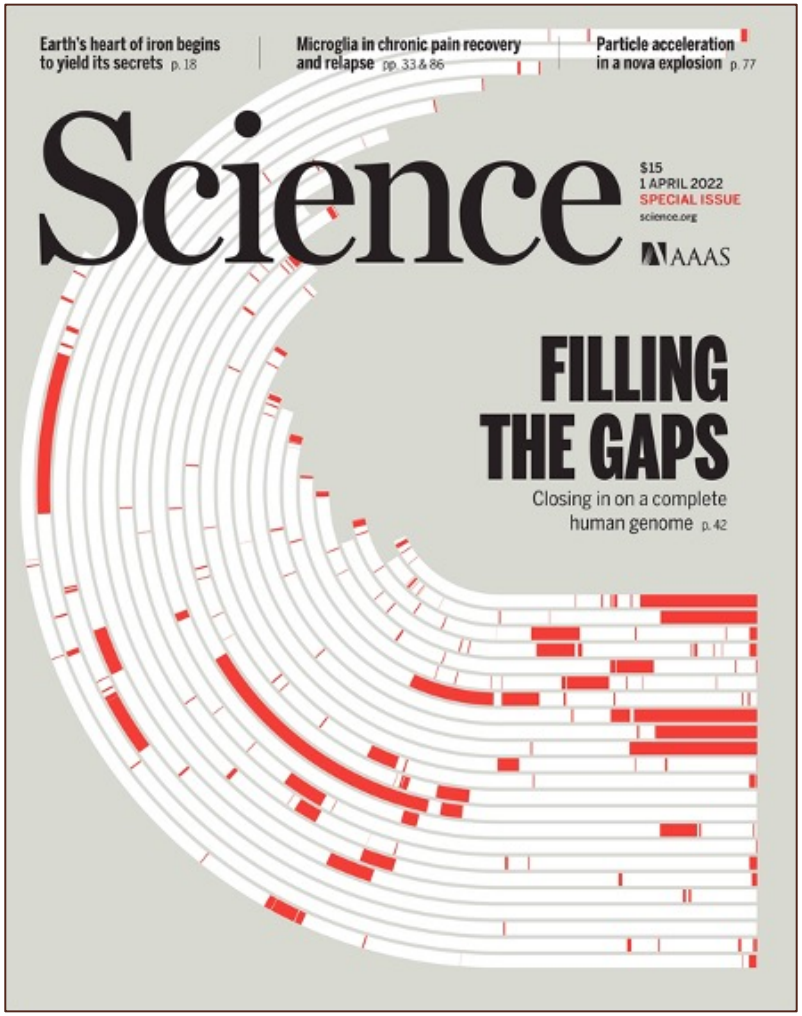
Overlap Graph

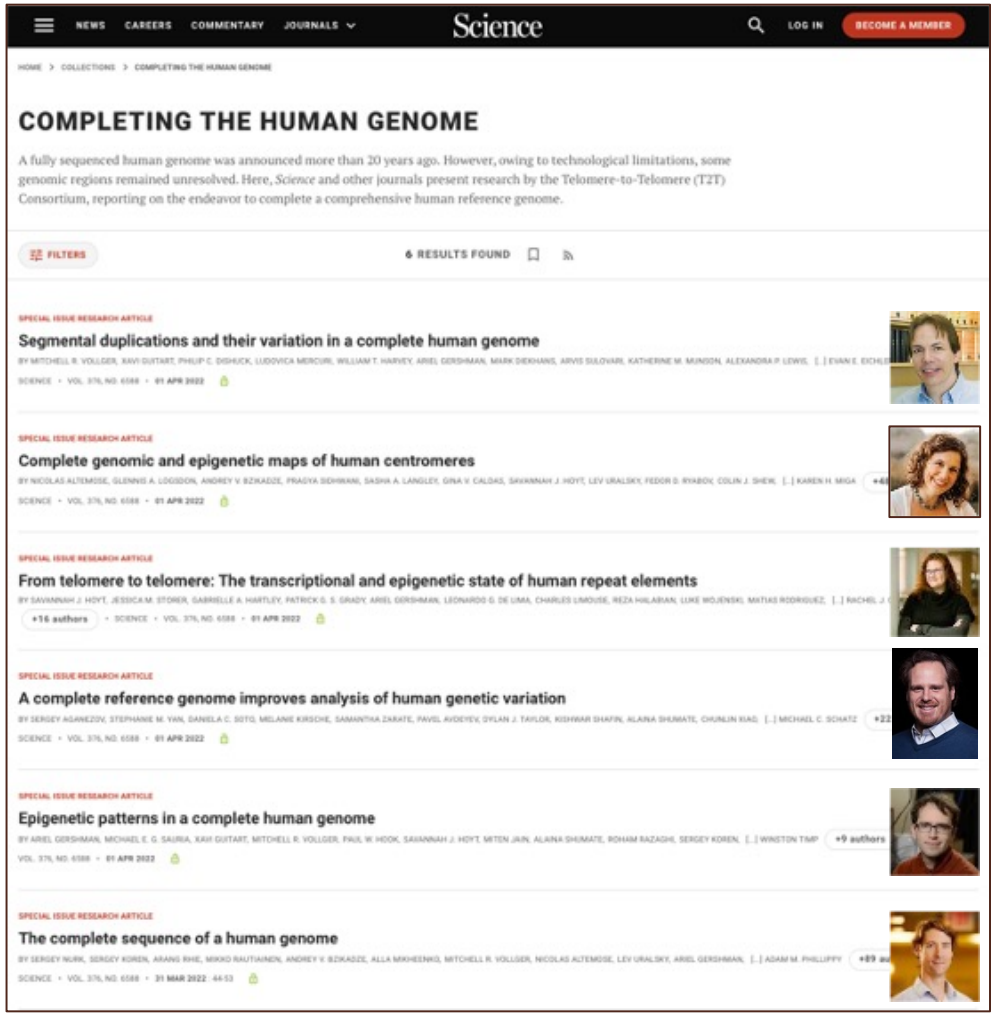


Long read assemblers

- Repeats depends on read length
- Read coherency, placements kept
- Tangled by high coverage

Assembly of Large Genomes using Second Generation Sequencing
Schatz MC, Delcher AL, Salzberg SL (2010) *Genome Research*. 20:1165-1173.


Earth's heart of iron begins to yield its secrets p. 18
Microglia in chronic pain recovery and relapse pp. 33 & 86
Particle acceleration in a nova explosion p. 77
Science
$15
1 APRIL 2022
SPECIAL ISSUE
science.org
AAAS
FILLING THE GAPS
Closing in on a complete human genome p. 42


NEWS
CAREERS
COMMENTARY
JOURNALS
Science
LOG IN
BECOME A MEMBER
HOME > COLLECTIONS > COMPLETING THE HUMAN GENOME
COMPLETING THE HUMAN GENOME
A fully sequenced human genome was announced more than 20 years ago. However, owing to technological limitations, some genomic regions remained unresolved. Here, Science and other journals present research by the Telomere-to-Telomere (T2T) Consortium, reporting on the endeavor to complete a comprehensive human reference genome.
FILTERS
6 RESULTS FOUND
SPECIAL ISSUE RESEARCH ARTICLE
Segmental duplications and their variation in a complete human genome
SPECIAL ISSUE RESEARCH ARTICLE
Complete genomic and epigenetic maps of human centromeres
SPECIAL ISSUE RESEARCH ARTICLE
From telomere to telomere: The transcriptional and epigenetic state of human repeat elements
+16 authors
SPECIAL ISSUE RESEARCH ARTICLE
A complete reference genome improves analysis of human genetic variation
SPECIAL ISSUE RESEARCH ARTICLE
Epigenetic patterns in a complete human genome
+9 authors
SPECIAL ISSUE RESEARCH ARTICLE
The complete sequence of a human genome

BGI CycloneSeq

Posted August 20, 2024.

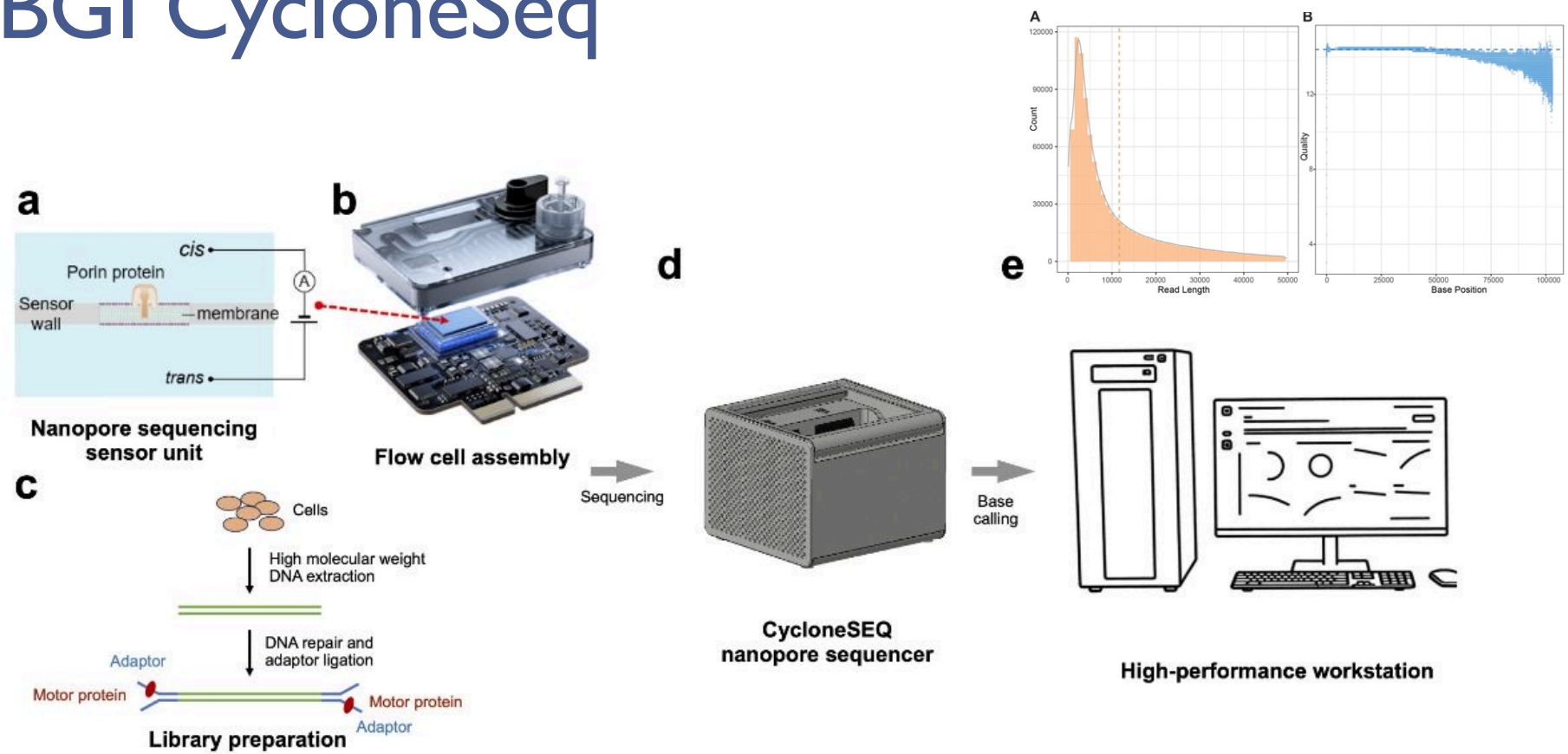


Figure 4: A single-molecule nanopore sequencing platform.
(a) Schematic representation of a sensor unit constructed with an insulating membrane containing an inserted nanopore, along with *cis* and *trans* chambers and corresponding electrodes. **(b)** Exploded view diagram of a flow cell. **(c)** Library preparation process for nanopore sequencing. **(d)** CycloneSEQ nanopore sequencer. **(e)** High-performance workstation and operating software for the CycloneSEQ nanopore sequencing platform.

A single-molecule nanopore sequencing platform
Zhang *et al* (2024) bioRxiv. doi: 10.1101/2024.08.19.608720

Oxford Nanopore Technologies to Sue BGI Group Affiliates for Contract Breach

Sep 10, 2024 | staff reporter

Save for later

NEW YORK – Oxford Nanopore Technologies disclosed Tuesday that it has filed an ex parte application in the US District Court for the Northern District of California to serve subpoenas on Complete Genomics, Innomics (formerly known as BGI Americas), MGI Americas, and other entities to support a lawsuit that it intends to file in the courts of England and Wales.

According to a regulatory filing, Oxford Nanopore said it plans to sue BGI Tech Solutions, BGI Group, BGI, Beijing Genomics Institute at Shenzhen, MGI Holdings, and MGI Tech in England for breach of contractual obligations, common law obligations of confidence, and duties under the Trade Secrets Regulations 2018. Oxford Nanopore also said it will file an entitlement claim to a license of certain patents.

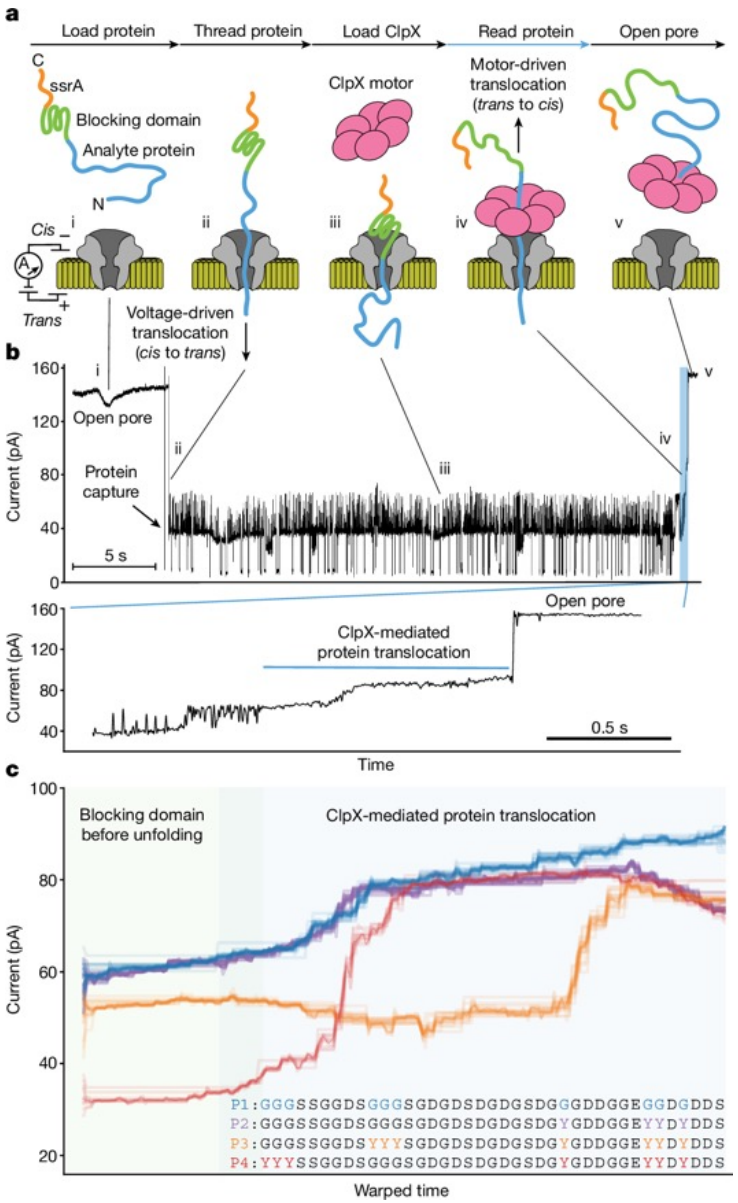
Oxford Nanopore said it "does not believe BGI's nanopore-based sequencing technology is able to be used in commercial products around the world without infringing or misappropriating the group's substantial portfolio of proprietary rights."

Named CycloneSeq, BGI's nanopore sequencing technology was developed and recently unveiled by BGI Research, a unit of BGI Group.

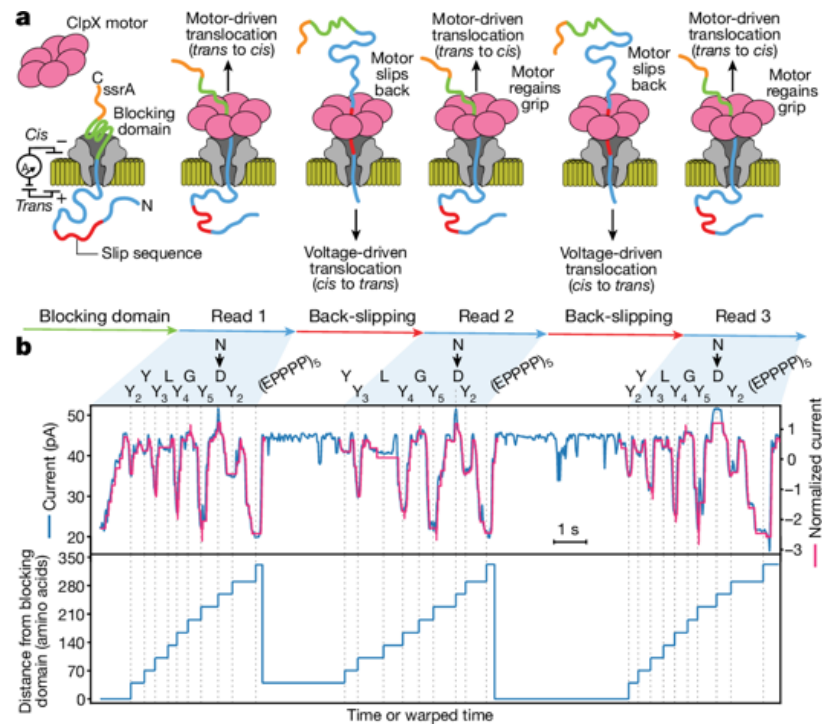
As previously reported, BGI Group, formally named BGI Shenzhen, has established a new subsidiary called Hangzhou BGI Xufeng Technology that holds the IP related to CycloneSeq. Earlier this year, BGI Xufeng granted the exclusive rights to commercialize and distribute CycloneSeq globally to BGI

Breaking News
- UK Report Provides Guidance on Use of AI in Genomic Health Prediction
- GT Molecular Awarded CDC Contract to Develop dPCR Wastewater Assays
- New Products Posted to GenomeWeb: Illumina, Applied DNA Sciences, Seegene, More
- People in the News at CareDx, Parse Biosciences, LetsGetChecked, Sphere Fluidics, MedGenome, More
- Population Testing for BRCA1/2 Appears Cost-Effective in Younger Canadian Women

What's Popular?
1 Singular Genomics Shares Soar on Deerfield Management Takeover Proposal
2 Cancer Centers Look to New NGS-Based 'Frankenpanel' to Get Fast Info on Clinically Actionable Genes
3 In Brief This Week: Illumina, Sophia Genetics, Thermo Fisher, Oxford Nanopore, Agilent, More



Rereading single protein molecules multiple times with an unfoldase slip sequence.



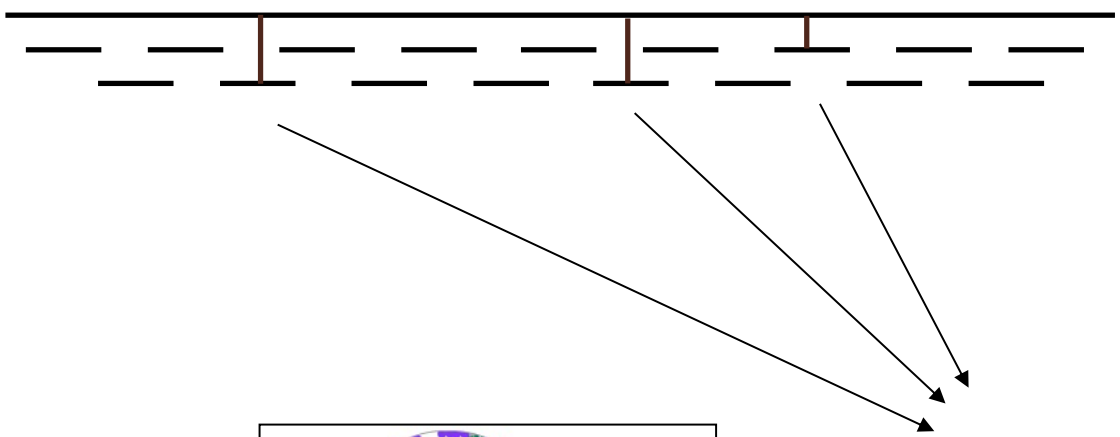
Multi-pass, single-molecule nanopore reading of long protein strands

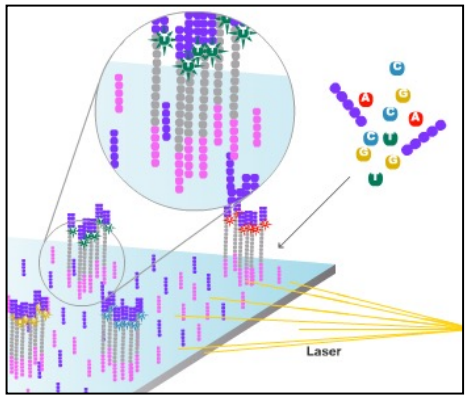
Motone *et al* (2024) Natue. https://doi.org/10.1038/s41586-024-07935-7

Read Mapping

Personal Genomics

How does your genome compare to the reference?





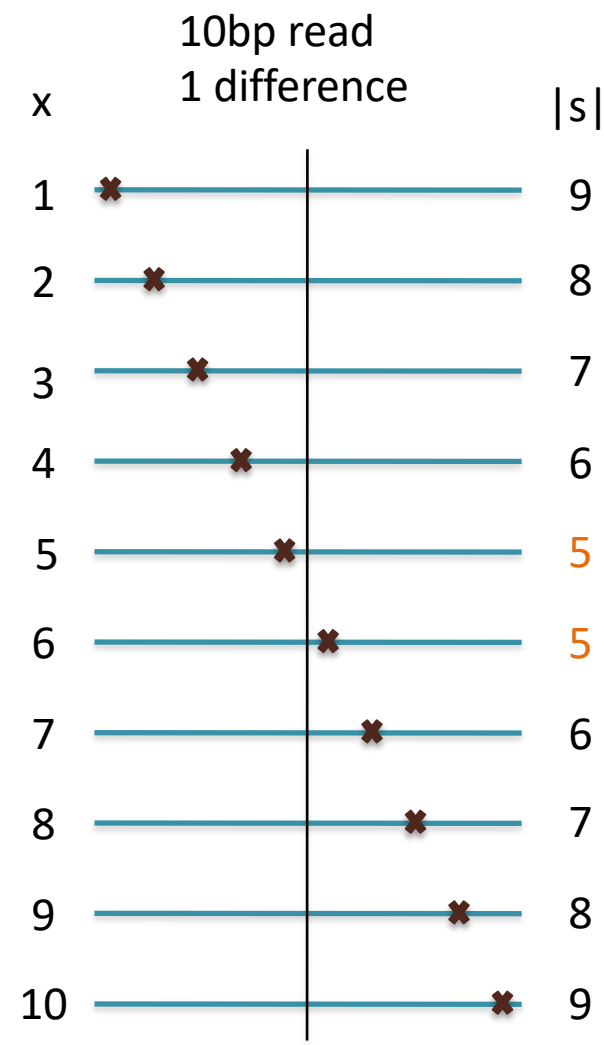
Heart Disease

Cancer

Presidential smile

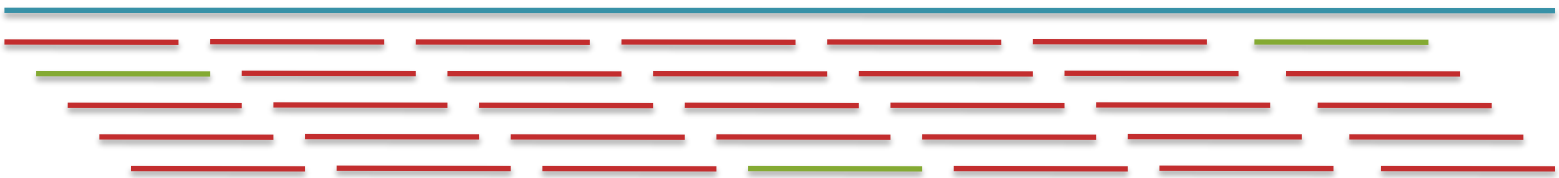
Seed-and-Extend Alignment

Theorem: An alignment of a sequence of length *m* with at most *k* differences ***must*** contain an exact match at least $s=m/(k+1)$ bp long
(*Baeza-Yates* and Perleberg, 1996)



- Proof: Pigeonhole principle
  - 1 pigeon can't fill 2 holes
- Seed-and-extend search
  - Use an index to rapidly find short exact alignments to seed longer in-exact alignments
    - BLAST, MUMmer, Bowtie, BWA, SOAP, …
  - Specificity of the depends on seed length
    - Guaranteed sensitivity for k differences
    - Also finds some (but not all) lower quality alignments <- heuristic

Brute Force Analysis



- Brute Force:
  - At every possible offset in the genome:
    - Do all of the characters of the query match?

- Analysis
  - Simple, easy to understand
  - Genome length = n [3B]
  - Query length = m [7]
  - Comparisons: (n-m+1) * m [21B]

- Overall runtime: O(nm)

[How long would it take if we double the genome size, read length?]

[How long would it take if we double both?]

Brute Force Reflections

Why check every position?

– GATTACA can't possibly start at position 15 [WHY?]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

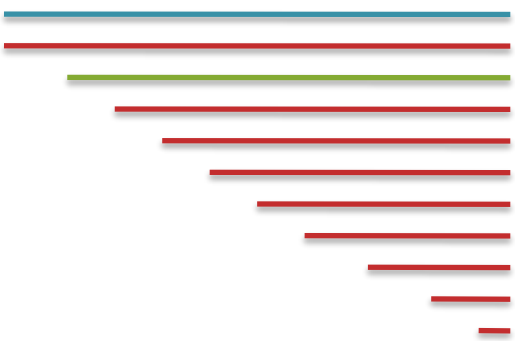
– Improve runtime to O(n + m) [3B + 7]
  - If we double both, it just takes twice as long
  - Knuth-Morris-Pratt, 1977
  - Boyer-Moyer, 1977, 1991

– For one-off scans, this is the best we can do (optimal performance)
  - We have to read every character of the genome, and every character of the query
  - For short queries, runtime is dominated by the length of the genome

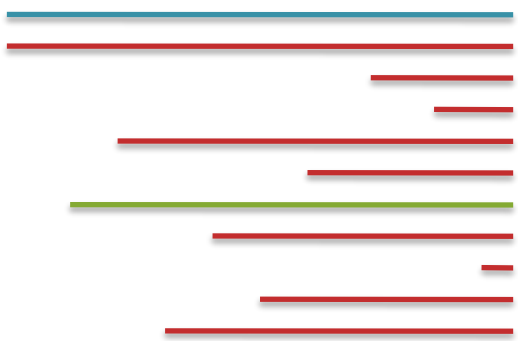
Suffix Arrays: Searching the Phone Book

- What if we need to check many queries?
  - We don't need to check every page of the phone book to find 'Schatz'
  - Sorting alphabetically lets us immediately skip 96% (25/26) of the book *without any loss in accuracy*

- Sorting the genome: Suffix Array (Manber & Myers, 1991)
  - Sort every suffix of the genome



Split into n suffixes



Sort suffixes alphabetically

[Challenge Question: How else could we split the genome?]

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15;

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15;

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
Hi →	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
Hi →	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11;

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
Hi →	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo →	8	CC…	14
	9	GATTACAGATTACC…	2
	10	GATTACC…	9
Hi →	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9;

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo	8	CC…	14
Hi	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9; Mid = (9+9)/2 = 9
  - Middle = Suffix[9] = GATTACA…
    => Match at position 2!

	#	Sequence	Pos
	1	ACAGATTACC…	6
	2	ACC…	13
	3	AGATTACC…	8
	4	ATTACAGATTACC…	3
	5	ATTACC…	10
	6	C…	15
	7	CAGATTACC…	7
Lo	8	CC…	14
Hi	9	GATTACAGATTACC…	2
	10	GATTACC…	9
	11	TACAGATTACC…	5
	12	TACC…	12
	13	TGATTACAGATTACC…	1
	14	TTACAGATTACC…	4
	15	TTACC…	11

Binary Search Analysis

- Binary Search

  Initialize search range to entire list

  mid = (hi+lo)/2; middle = suffix[mid]

  if query matches middle: done

  else if query < middle: pick low range

  else if query > middle: pick hi range

  Repeat until done or empty range [WHEN?]

- Analysis
  - More complicated method
  - How many times do we repeat?
    - How many times can it cut the range in half?
    - Find smallest x such that: $n/(2^x) \leq 1$; $x = \lg_2(n)$ [32]

- Total Runtime: O(m lg n)
  - More complicated, but much faster!
  - Looking up a query loops 32 times instead of 3B

[How long does it take to search 6B or 24B nucleotides?]

Binary Search Analysis

- Binary Search

  Initialize search range to entire list

  mid = (hi+lo)/2; middle = suffix[mid]

  if query matches middle: done

  else if query < middle: pick low range

  else if query > middle: pick hi range

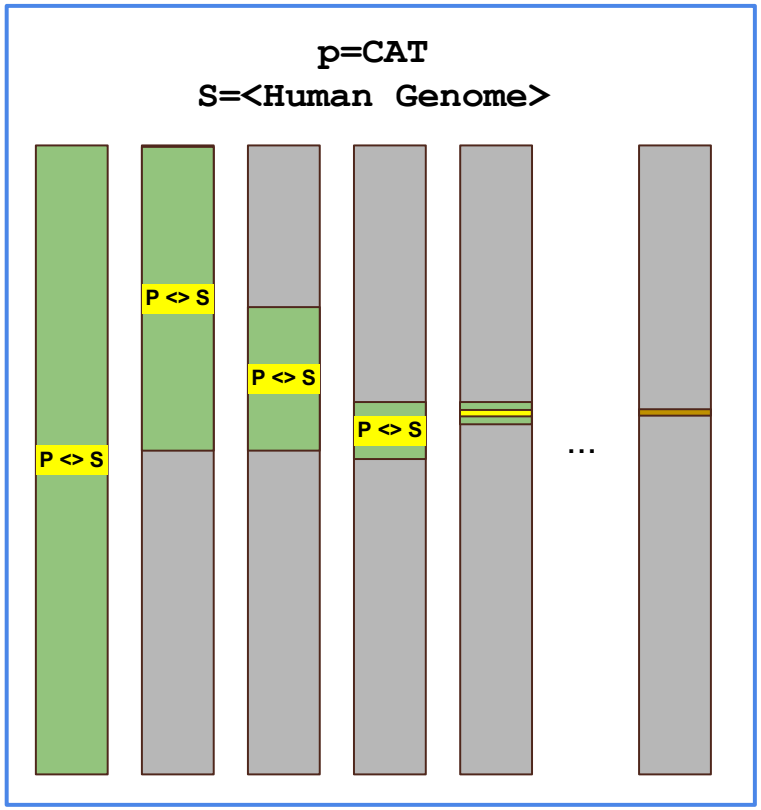
  Repeat until done or empty range [WHEN?]

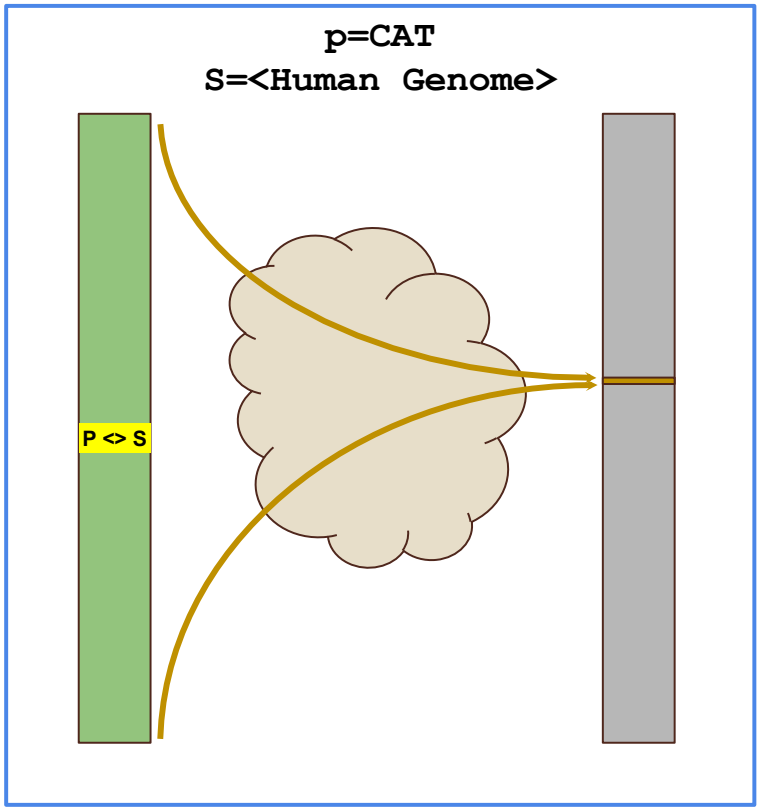
- Analysis
  - More complicated method
  - How many times do we repeat?
    - How many times can it cut the range in half?
    - Find smallest x such that: $n/(2^x) \leq 1$; $x = \lg_2(n)$ [32]

- Total Runtime: O(m lg n)
  - More complicated, but much faster!
  - Looking up a query loops 32 times instead of 3B

Can be reduced to O(m + lg n)
using an auxiliary data structure called the LCP array

Sapling: Accelerating Suffix Array Queries with Learned Data Models





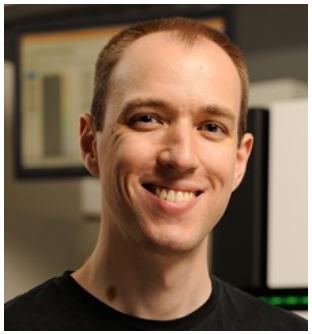
What if instead of a slow algorithmic approach to find the correct rows, we could somehow quickly guess/predict the correct rows?

Kirsche, M, Das, A, Schatz, MC (2020) Bioinformatics
doi: https://doi.org/10.1093/bioinformatics/btaa911

Algorithmic challenge

How can we combine the speed of a suffix array O(m + lg(n)) (or even O(m)) with the size of a brute force analysis (n bytes)?

What would such an index look like?



Bowtie: Ultrafast and memory efficient alignment of short DNA sequences to the human genome

Slides Courtesy of Ben Langmead

Algorithm Overview

1. Split read into segments

Read

CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA

Read (reverse complement)

TACAGGCCTGGGTAAAATAAGGCTGAGAGCTACTGG

Policy: extract 16 nt seed every 10 nt

Seeds

+, 0: CCAGTAGCTCTCAGCC
+, 10: TCAGCCTTATTTTACC
+, 20: TTTACCCAGGCCTGTA

-, 0: TACAGGCCTGGGTAAA
-, 10: GGTAAAATAAGGCTGA
-, 20: GGCTGAGAGCTACTGG

2. Lookup each segment and prioritize

Seeds

+, 0: CCAGTAGCTCTCAGCC
+, 10: TCAGCCTTATTTTACC
+, 20: TTTACCCAGGCCTGTA
-, 0: TACAGGCCTGGGTAAA
-, 10: GGTAAAATAAGGCTGA
-, 20: GGCTGAGAGCTACTGG

→ Ungapped alignment with FM Index

→ Seed alignments (as B ranges)

{ [211, 212], [212, 214] }
{ [653, 654], [651, 653] }
{ [684, 685] }
{ }
{ }
{ [624, 625] }

3. Evaluate end-to-end match

Extension candidates

SA:684, chr12:1955
SA:624, chr2:462
SA:211: chr4:762
SA:213: chr12:1935
SA:652: chr12:1945

→ SIMD dynamic programming aligner

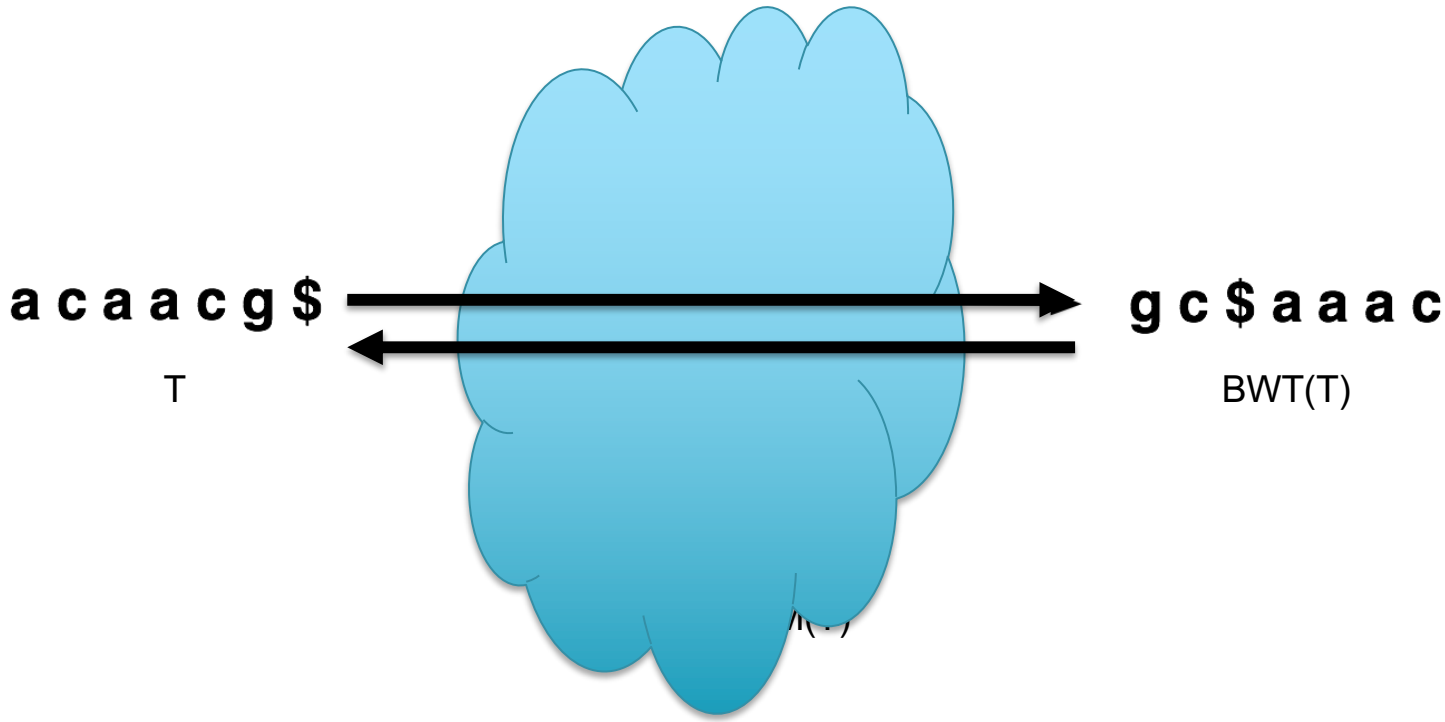
→ SAM alignments

```
r1   0    chr12    1936     0
     36M  *    0    0
     CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA
     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
     AS:i:0    XS:i:-2   XN:i:0
     XM:i:0    XO:i:0    XG:i:0
     NM:i:0    MD:Z:36   YT:Z:UU
     YM:i:0
...
```

(Langmead & Salzberg, 2012)

Burrows-Wheeler Transform

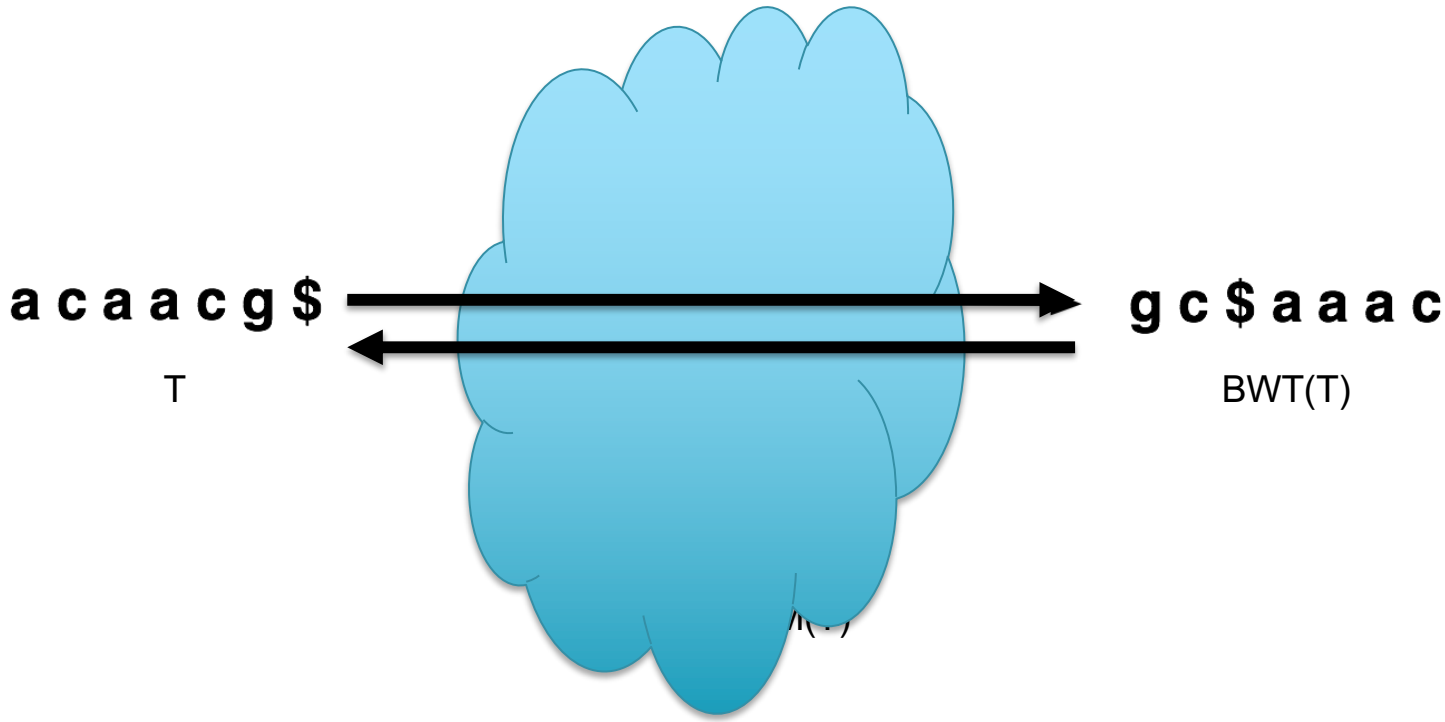
- Reversible permutation of the characters in a text



A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Reversible permutation of the characters in a text



A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g $ ⟶

T

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g $ →

T

a c a a c g $
c a a c g $ a
a a c g $ a c
a c g $ a c a
c g $ a c a a
g $ a c a a c
$ a c a a c g

All cyclic permutations

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g $ →

T

$ a c a a c g
a a c g $ a c
a c a a c g $
a c g $ a c a
c a a c g $ a
c g $ a c a a
g $ a c a a c

Sorted cyclic permutations
AKA Burrows Wheeler Matrix

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Permutation of the characters in a text

a c a a c g $ →

T

$	a	c	a	a	c	g
a	a	c	g	$	a	c
a	c	a	a	c	g	$
a	c	g	$	a	c	a
c	a	a	c	g	$	a
c	g	$	a	c	a	a
g	$	a	c	a	a	c

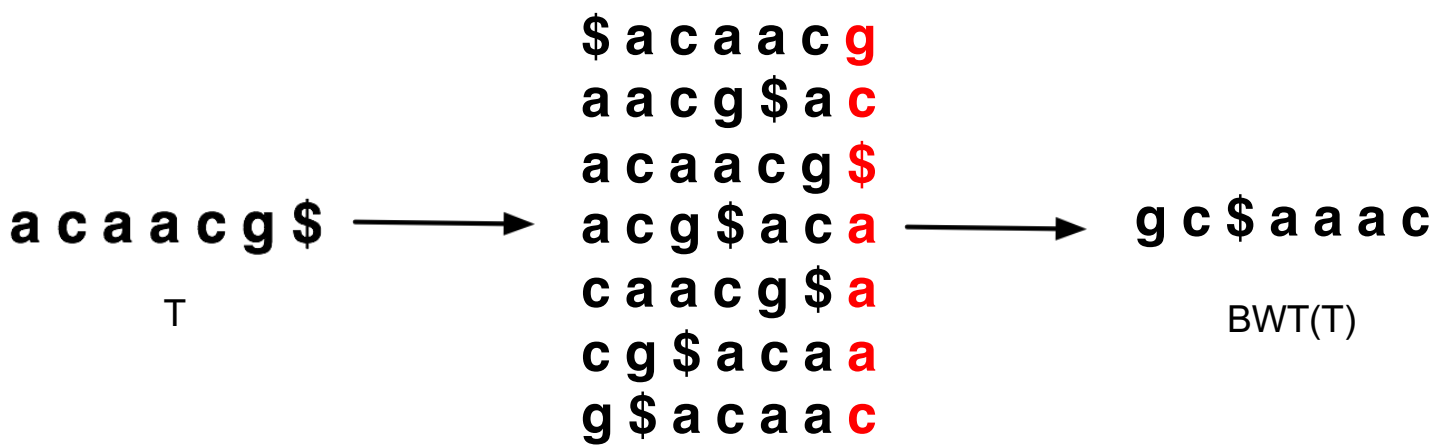
Sorted cyclic permutations
AKA Burrows Wheeler Matrix

Last Column = Burrows Wheeler Transform

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

- Permutation of the characters in a text

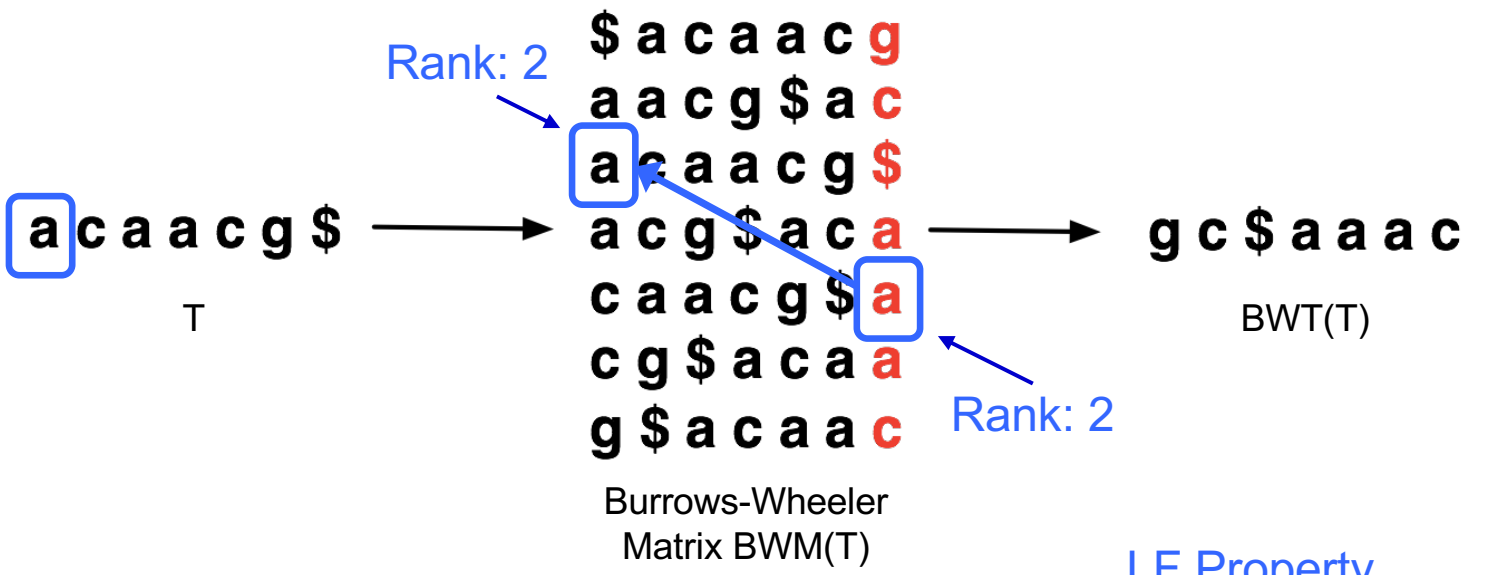


BWT(T) is the index for T

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124

Burrows-Wheeler Transform

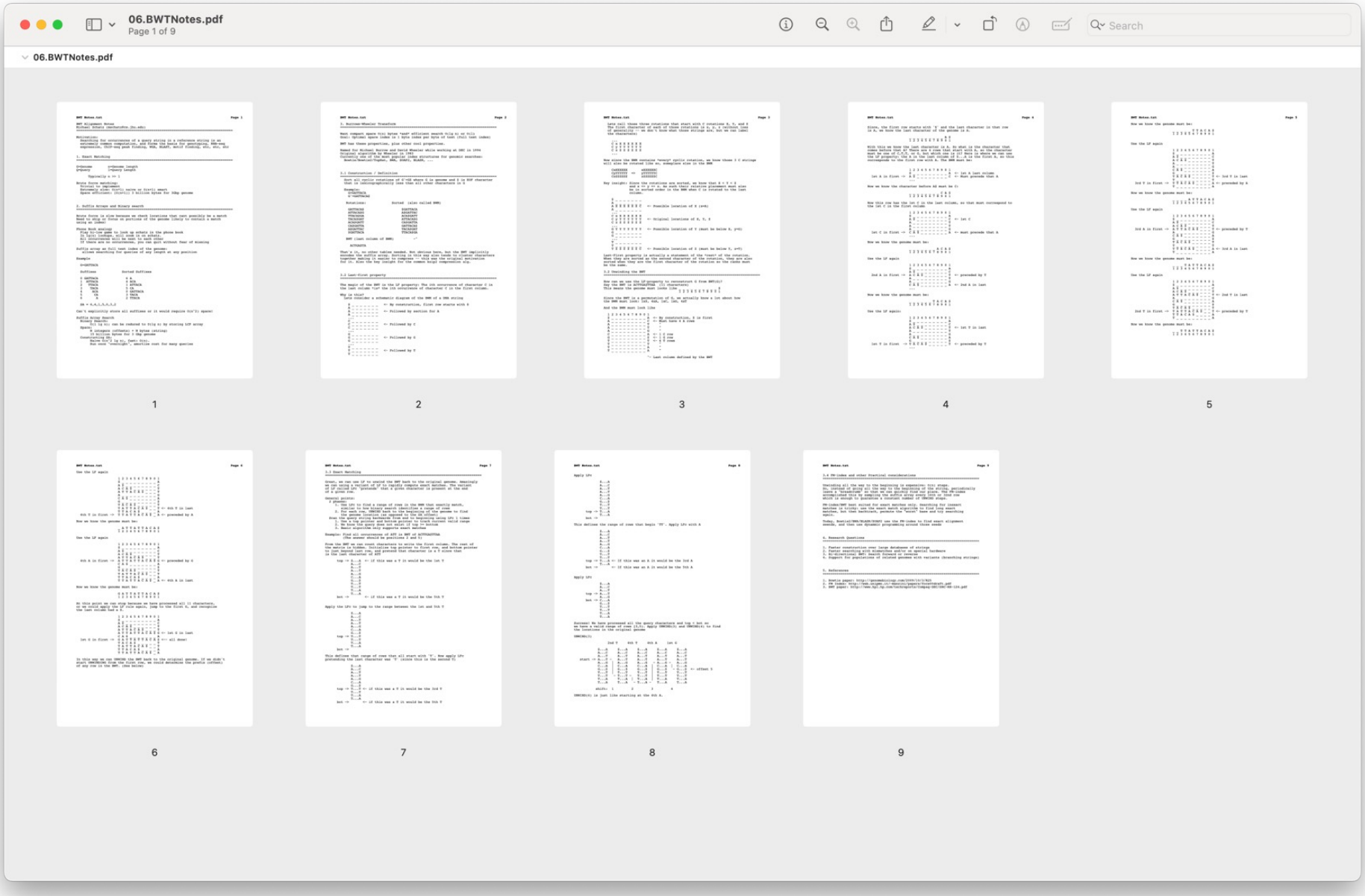
- Reversible permutation of the characters in a text



LF Property
implicitly encodes
Suffix Array

- BWT(T) is the index for T

A block sorting lossless data compression algorithm.
Burrows M, Wheeler DJ (1994) *Digital Equipment Corporation.* Technical Report 124



Recompute the original text when the BWT is ACTTGA$TTAA