



Introduction to Read-Based Alignment

Michael C. Zody, Ph.D.
Workshop on Genomics
Cesky Krumlov
January 11, 2018

Aligning to a Reference

- Aligning sequences is a classic problem
 - Early bioinformatic problem
 - Very similar to older text matching problems
- Several algorithms exist
 - Tradeoffs of speed versus accuracy, sensitivity
- Sequencing throughput creates new problems
 - Short reads have less information than long seqs
 - Data volume requires faster processing per read

Example of alignment

Read:

TCAACTCTGCCAACACCTTCCTCCTCCAGGAAGCACTCCTGGATTTCCTCTTGCCAACAAGATTCTGGGAGGGCA

Genome:

ATAAAATGGCCAAAATTAAGTAGAAGGTGAGTAGAACTTAAATAAACTAATTACCATTGATGAGAAAAAAATC
TGCCACTGAAAAAGGCACCCGGTCCAGAGGGTTTCATGAGCGGGAAGTGTAGAAACCTTTCGAATTCAACTCTGC
CAACACCTTCCTCCTCCAGGAAGCACTCCTGGATTTCCTCTTGCCAACAAGATTCTGGGAGGGCAGCTCCTCCA
ACATGCCCCCAACAGCTCTCTGCAGACATATCATATCATATCATATCTTCCATACCATAACTGCCATGCCATACA

Example of alignment

Read:

TCAACTCTGCCAACACCTTCCTCCTCCAGGAAGCACTCCTGGATTTCCTCTTGCCAACAAGATTCTGGGAGGGCA

Genome:

ATAAAATGGCCAAAATTAAGTAGAAGGTGAGTAGAACTTAAATAAACTAATTACCATTGATGAGAAAAAAATC
TGCCACTGAAAAAGGCACCCGGTCCAGAGGGTTTCATGAGCGGGAAGTGTAGAAACCTTTCGAATTCAACTCTGC
CAACACCTTCCTCCTCCAGGAAGCACTCCTGGATTTCCTCTTGCCAACAAGATTCTGGGAGGGCAGCTCCTCCA
ACATGCCCCAACAGCTCTCTGCAGACATATCATATCATATCATATCTTCCATACCATAACTGCCATGCCATACA

How Would You Find That?

- Brute force comparison
- Smith-Waterman
- Suffix Tree
- Burrows-Wheeler Transform

Brute Force Method

TCGATCC



GACCTCATCGATCCACTG

Brute Force Method

TCGATCC
X
GACCTCATCGATCCACTG

Brute Force Method

TCGATCC
X
GACCTCATCGATCCACTG

Brute Force Method

TCGATCC
GACCTCATCGATCCACTG



Brute Force Method

TCGATCC
GACCTCATCGATCCCACTG

The diagram illustrates a sequence alignment. The top sequence, 'TCGATCC', is written in red. The bottom sequence, 'GACCTCATCGATCCCACTG', is written in black. Vertical blue lines connect the bases of the top sequence to the corresponding bases in the bottom sequence, specifically highlighting the match between 'TCGATCC' and the 7th through 13th bases of the bottom sequence.

Smith-Waterman

Simplistic Scoring Scheme:

+1 match

if moving diagonally

-1 mismatch

if moving diagonally

-1 gap

if moving hor. or vert.

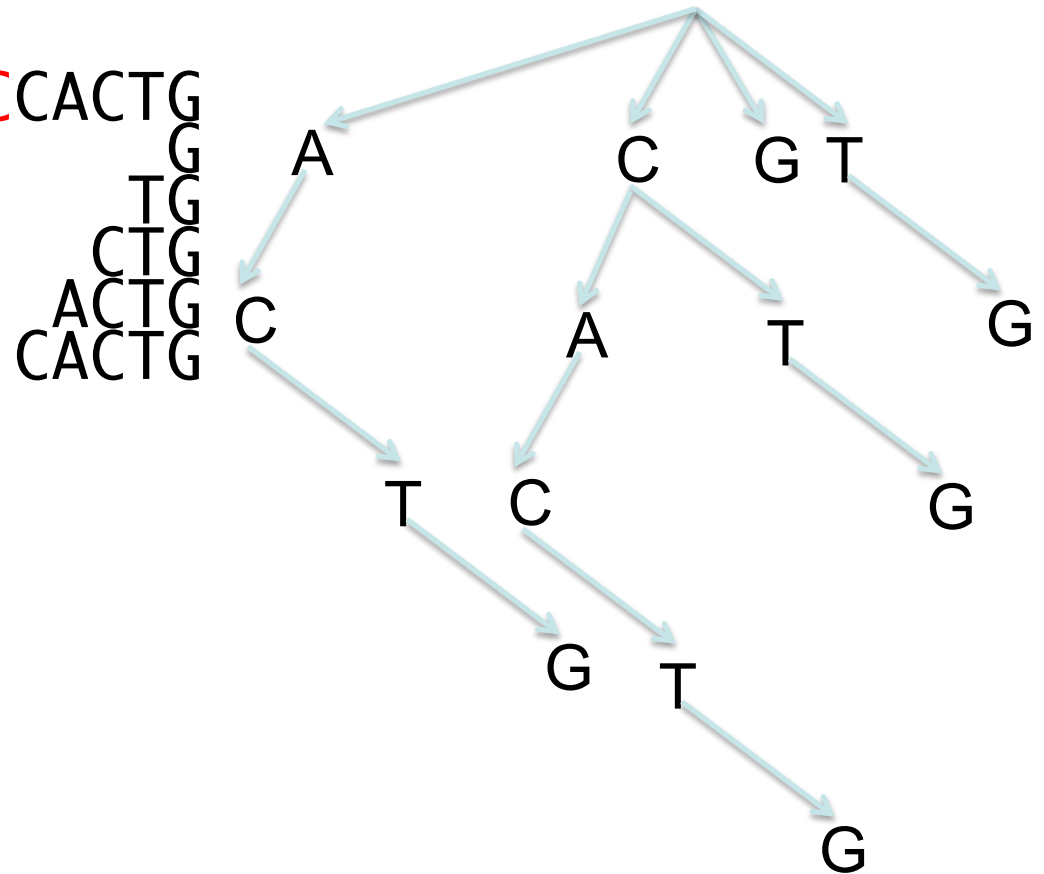
take the
highest of
these

(no penalty for terminal gaps)

^ T C G A T C C	0	-1	-2	0	2	1	1	0	1	3	3	2	3	5	7	6	5	4	3	2
	0	-1	-1	1	1	0	1	0	2	4	3	2	4	6	5	4	3	2	1	0
	0	-1	0	0	-1	0	-1	-1	3	2	1	1	5	4	3	2	1	0	1	0
	0	0	1	0	-1	-2	0	2	1	0	2	4	3	2	1	0	1	0	-1	-2
	0	1	0	-1	-2	-1	1	1	0	1	3	2	1	1	1	0	-1	-2	-1	0
	0	-1	0	-1	0	0	2	1	0	2	1	0	0	2	1	0	-1	0	0	0
	0	-1	-1	-1	-1	1	0	-1	1	0	-1	-1	1	0	-1	-1	-1	-1	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	^	G	A	C	C	T	C	A	T	C	G	A	T	C	C	C	A	C	T	G

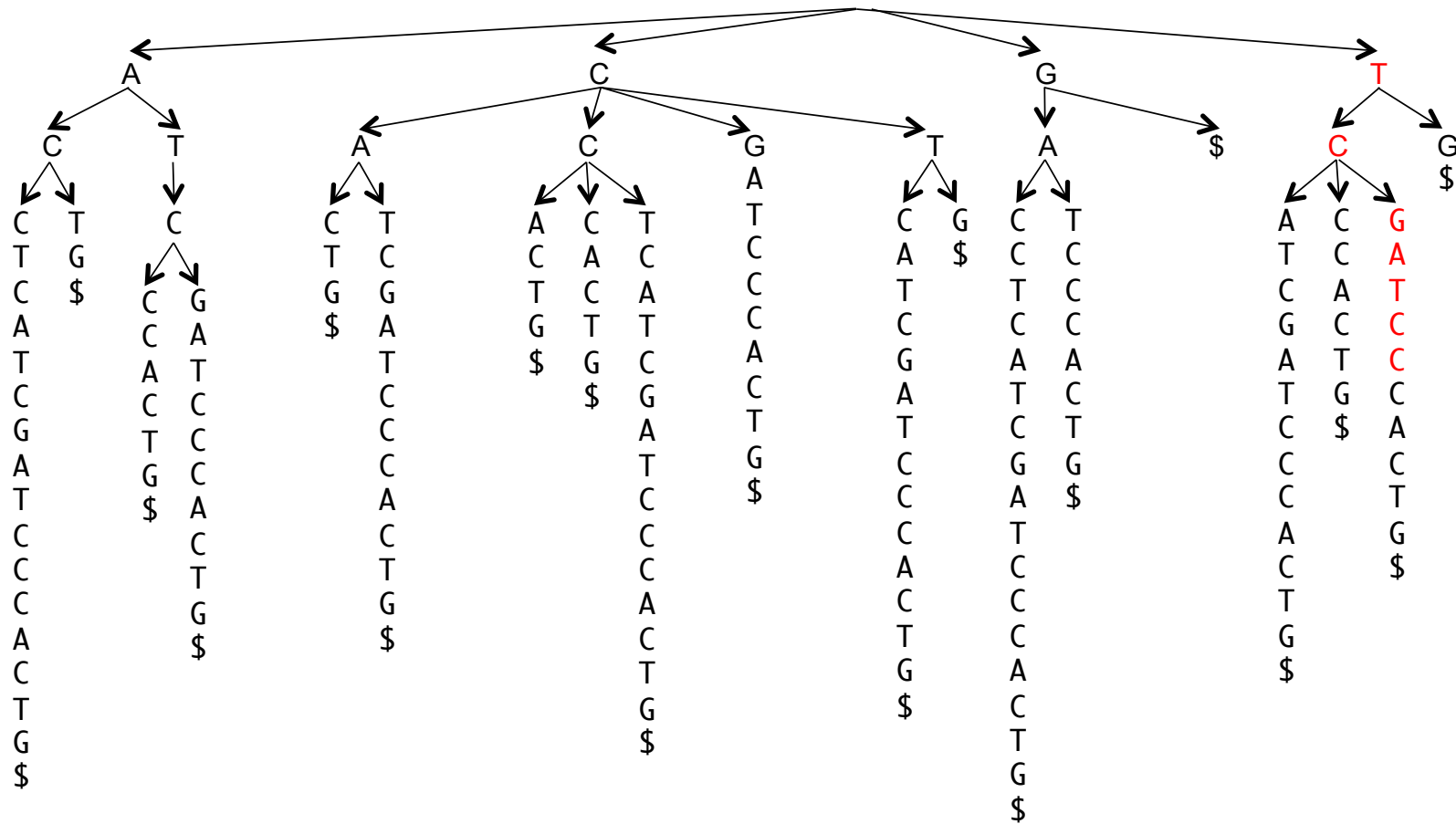
Suffix Tree

GACCTCATCGATCCACTG



Suffix Tree

GACCTCATCGATCCCACTG



Burrows-Wheeler Transform



GACCTCATCGATCCCACTG\$
ACCTCATCGATCCCACTG\$G
CCTCATCGATCCCACTG\$GA
CTCATCGATCCCACTG\$GAC
TCATCGATCCCACTG\$GACC
CATCGATCCCACTG\$GACCT
ATCGATCCCACTG\$GACCTC
TCGATCCCACTG\$GACCTCA
CGATCCCACTG\$GACCTCAT
GATCCCACTG\$GACCTCATC
ATCCCACTG\$GACCTCATCG
TCCCACTG\$GACCTCATCGA
CCCACTG\$GACCTCATCGAT
CCACTG\$GACCTCATCGATC
CACTG\$GACCTCATCGATCC
ACTG\$GACCTCATCGATCCC
CTG\$GACCTCATCGATCCCA
TG\$GACCTCATCGATCCCAC
G\$GACCTCATCGATCCCACT
\$GACCTCATCGATCCCACTG

ACCTCATCGATCCCACTG\$G
ACTG\$GACCTCATCGATCCC
ATCCCACTG\$GACCTCATCG
ATCGATCCCACTG\$GACCTC
CACTG\$GACCTCATCGATCC
CATCGATCCCACTG\$GACCT
CCACTG\$GACCTCATCGATC
CCCACTG\$GACCTCATCGAT
CCTCATCGATCCCACTG\$GA
CGATCCCACTG\$GACCTCAT
CTCATCGATCCCACTG\$GAC
CTG\$GACCTCATCGATCCCA
GACCTCATCGATCCCACTG\$
GATCCCACTG\$GACCTCATC
G\$GACCTCATCGATCCCACT
TCATCGATCCCACTG\$GACC
TCCCACTG\$GACCTCATCGA
TCGATCCCACTG\$GACCTCA
TG\$GACCTCATCGATCCCAC
\$GACCTCATCGATCCCACTG

How Do We Use This To Align?

GAC
CAC
GAT
CAT
CCA
→ TCA
CCC
→ TCC
ACC
→ TCG
CCT
ACT
\$GA
CGA
→ TG\$
CTC
ATC
ATC
CTG
G\$G

- Start with the transform column
- My read starts with a T, so I want rows with Ts in them
- This column gives me all the single nucleotide counts
- Sort the single nucleotide counts to get the alphabetically first column
- Now these two columns give me all the dinucleotide counts
- Sort those to get the alphabetically first two columns
- Now there is only one place my read can match

FM Index

G
C
G
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Start with the transform column
- Count all the characters, sort them, and store the count of lower characters

A	C	G	T	\$
0	4	12	15	19

- This gives the positions of all the bases in the first column (because it's sorted)

FM Index

A
A
A
A
C
C
C
C
C
C
C
C
C
G
G
G
T
T
T
T
T
T
\$

G
C
G
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- Start at the end and use the count table to look up the position of the *last base* in the *first column*

A	C	G	T	\$
0	4	12	15	19

- The last column comes immediately before the first column
- Find all the rows of the last column with the next to last base

FM Index

AC
AC
AT
AT
CA
CA
CC
CC
CC
CG
CT
CT
GA
GA
G\$
TC
TC
TC
TG
\$G

G
C
G
C
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- The order of a given character in the last column matches the order of the same instance of that character in the first column
- The 3rd-5th Cs in the last column precede Cs in the first column, so we now want the 3rd-5th Cs in column 1

A	C	G	T	\$
0	4	12	15	19

- Now we take the next character and look for Ts in the last column (the 2nd T)

FM Index

ACC
ACT
ATC
ATC
CAC
CAT
CCA
CCC
CCT
CGA
CTC
CTG
GAC
GAT
G\$G
TCA
TCC
TCG
TG\$
\$GA

G
C
G
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- The second T is preceded by the 3rd A

A	C	G	T	\$
0	4	12	15	19

FM Index

ACCT
ACTG
ATCC
ATCG
CACT
CATC
CCAC
CCCA
CCTC
CGAT
CTCA
CTG\$
GACC
GATC
G\$GA
TCAT
TCCC
TCGA
TG\$G
\$GAC

G
C
G
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- The third A is preceded by the 2nd G

A	C	G	T	\$
0	4	12	15	19

FM Index

ACCTC
ACTG\$
ATCCC
ATCGA
CACTG
CATCG
CCACT
CCCAC
CCTCA
CGATC
CTCAT
CTG\$G
GACCT
GATCC
G\$GAC
TCATC
TCCCA
TCGAT
TG\$GA
\$GACC

G
C
G
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- The second G is preceded by the 6th C

A	C	G	T	\$
0	4	12	15	19

FM Index

ACCTCA
ACTG\$G
ATCCCA
ATCGAT
CACTG\$
CATCGA
CCACTG
CCCACT
CCTCAT
CGATCC
CTCATC
CTG\$GA
GACCTC
GATCCC
G\$GACC
TCATCG
TCCAC
TCGATC
TG\$GAC
\$GACCT

G
C
G
C
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- The sixth C is preceded by the 3rd T

A	C	G	T	\$
0	4	12	15	19

FM Index

ACCTCAT
ACTG\$GA
ATCCCAC
ATCGATC
CACTG\$G
CATCGAT
CCACTG\$
CCCACTG
CCTCATC
CGATCCC
CTCATCG
CTG\$GAC
GACCTCA
GATCCCA
G\$GACCT
TCATCGA
TCCCACT
TCGATCC
TG\$GACC
\$GACCTC

G
C
G
C
C
T
C
T
A
T
C
A
\$
C
T
C
A
A
C
G

- Take the query sequence **TCGATCC**
- And we're done

A	C	G	T	\$
0	4	12	15	19

- To find the position in the genome, we keep a separate index of positions for a sparse set of rows in the table and then just walk through the transform to the nearest indexed row

Heuristic Improvements

- Seeding
 - Use a hash of exact matches to limit searches
 - Use varying hash types
- Limiting mismatches or indels
 - Constrain poor quality searches
 - Known as ‘banded’ in Smith-Waterman
- Using base quality to guide backtracking

Common Short Read Aligners

- Seed and Smith-Waterman extend
 - Mosaik, BFAST, Novoalign
- BWA align gap-free
 - Bowtie
- BWA align with gaps
 - BWA aln, Bowtie2
- BWA Seed and Smith-Waterman extend
 - BWA mem
- Seed clustering and stitching
 - Blasr, STAR

Blasr

- Designed for long error-prone reads (PacBio)
- Combines multiple methods
- Starts by finding short exact matches using suffix or B-W
- Next locally identifies a linear chain of shorter exact matches
- Performs banded Smith-Waterman constrained by the shorter exact matches