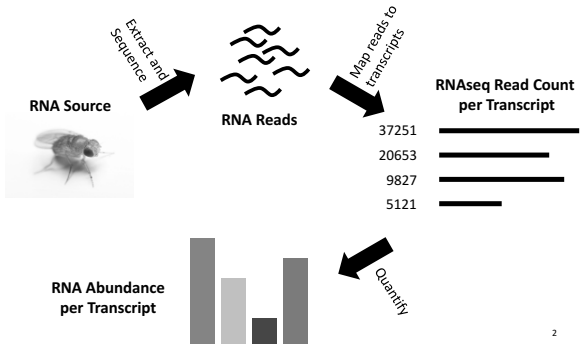
RNAseq: a Closer Look at Read Mapping and Quantitation

Jeremy Buhler
for GEP Alumni Workshop

1

RNAseq Pipeline for Expression Analysis



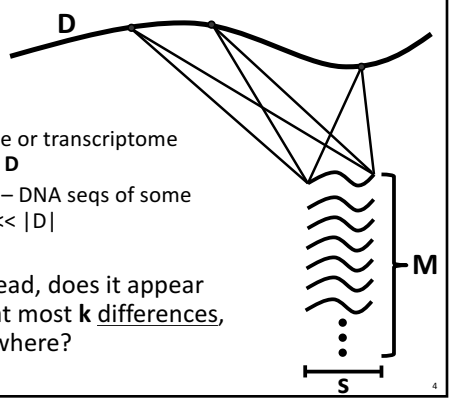
2

Topics for Today

- **Read mapping** and RNA **quantitation** from read counts are two key computational steps in RNAseq expression analysis.
- **Mapping**: how do we do it **fast**?
- **Quantitation**: how do we get from mapped reads to transcript abundance?

3

Problem: Short-Read Mapping



- Given
  - a genome or transcriptome database **D**
  - **M** reads – DNA seqs of some length **s** << |D|
- For each read, does it appear in D with at most **k** differences, and if so, where?

4

Typical Parameters

- Database **D** has size 10^7 – 10^{10} bases
- Number of reads **M** is 10^6 – 10^8
- Length **s** is 75-150 (may vary among reads)
- # differences **k** is 0-3 (added, deleted, changed) to account for sequencing errors, polymorphisms

5

Design Pressures

- # reads M is huge! → minimal time per read
- Cannot afford to spend O(|D|) time per read as in BLAST → need to **index database** in order to spend less time matching each read
- Index should be small enough to reside in fast memory, so as to maximize mapping speed

6

Basic Idea: Suffix Tree Index

- First, sort all suffixes of database string D to create sorted **suffix array** A.
- Second, merge all common prefixes of adjacent strings in A to create a **suffix tree** T.
- Leaves of tree correspond to suffixes of D.

7

Construct a Suffix Array (A)

D = a c a g a c c a g a $ (0 1 2 3 4 5 6 7 8 9 10)

Position	Suffix
10	$
9	a$
8	ga$
7	aga$
6	caga$
5	ccaga$
4	accaga$
3	gaccaga$
2	agaccaga$
1	cagaccaga$
0	acagaccaga$

Sort by suffix

A	Suffix
10	$
9	a$
0	acagaccaga$
4	accaga$
7	aga$
2	agaccaga$
6	caga$
1	cagaccaga$
5	ccaga$
8	ga$
3	gaccaga$

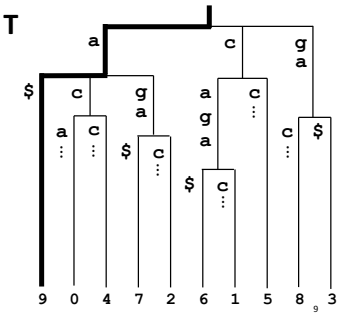
8

Suffix Tree Example

D = a c a g a c c a g a $ (0 1 2 3 4 5 6 7 8 9 10)

A	Suffix
10	$
9	a$
0	aca…
4	acc…
7	aga$
2	agac…
6	caga$
1	cagac…
5	cca…
8	ga$
3	gac…

T

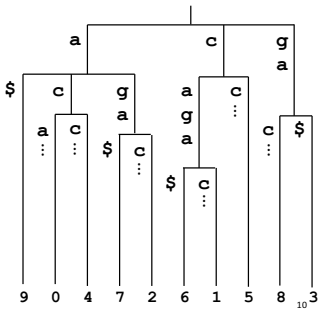


9

Rapid Matching vs Suffix Tree

- Can find all matches to a read in D using time proportional to read length s.

Where is **cag**?

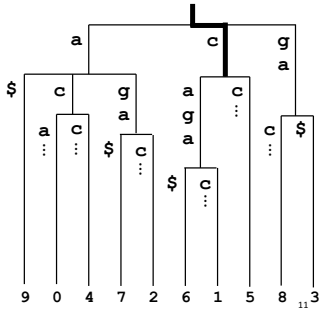


10

Rapid Matching vs Suffix Tree

- Can find all matches to a read in D using time proportional to read length s.

Where is **cag**?

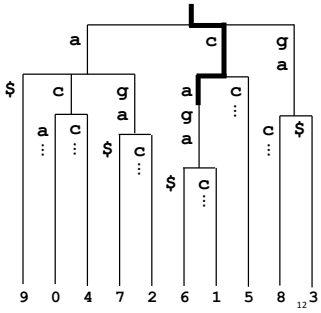


11

Rapid Matching vs Suffix Tree

- Can find all matches to a read in D using time proportional to read length s.

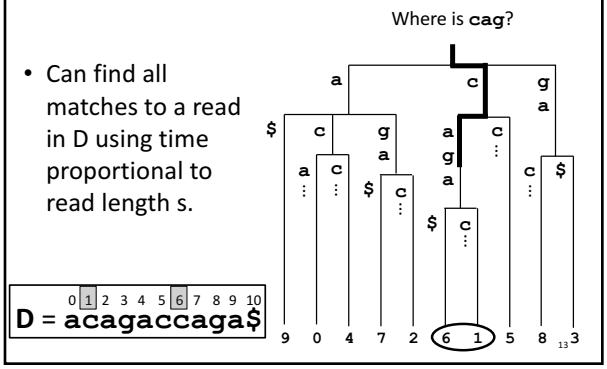
Where is **cag**?



12

Rapid Matching vs Suffix Tree

- Can find all matches to a read in D using time proportional to read length s.



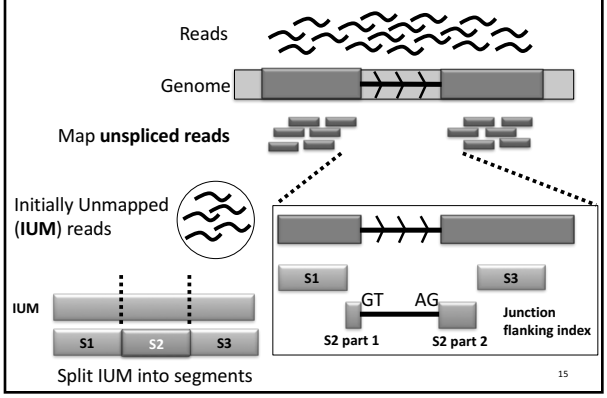
13

Extension to Inexact Matching

- To permit matches with k **substitutions**, try multiple paths, but charge for each mismatch. (Try searching for "aca" with one mismatch.)
- To permit matches with k **differences**, can do dynamic programming or heuristic search to compute edit distance of read against many paths in tree.
- Descent stops for a read when we hit bottom of tree or find that path requires > k differences.

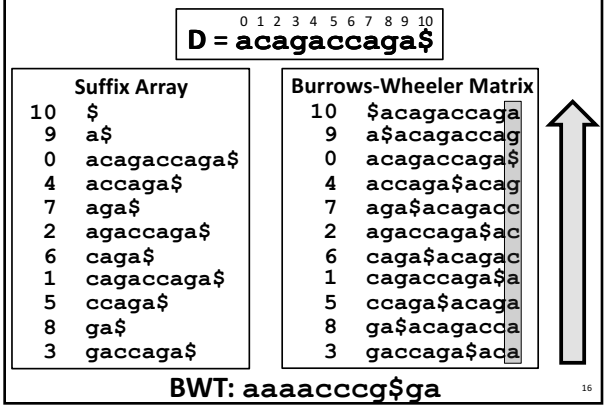
14

Spliced read mapping (TopHat)



15

Burrows-Wheeler Transform (BWT)



16

"Virtual Tree": FM-Index

- Suffix trees need 10-100 bytes memory/base indexed
- **Theorem** [Ferragina & Manzini, 2000]: we can build an *O(1)-time oracle* that, given "suffix tree posn" corresponding to a string z, returns posn corresponding to one-char extension z.c
- Hence, we can simulate suffix tree matching without explicitly storing the tree!
- Space cost: ~1 byte/base

17

Commonly Used Mapping Software

- **Bowtie** [Langmead *et al.* 2009, 2012]
- **BWA** [Li & Durbin 2009, 2010]
- **SOAP2** [Li *et al.* 2009]

All these tools use variants of FM-index approach.

18