Read Mapping

Slides by Carl Kingsford

Bowtie

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

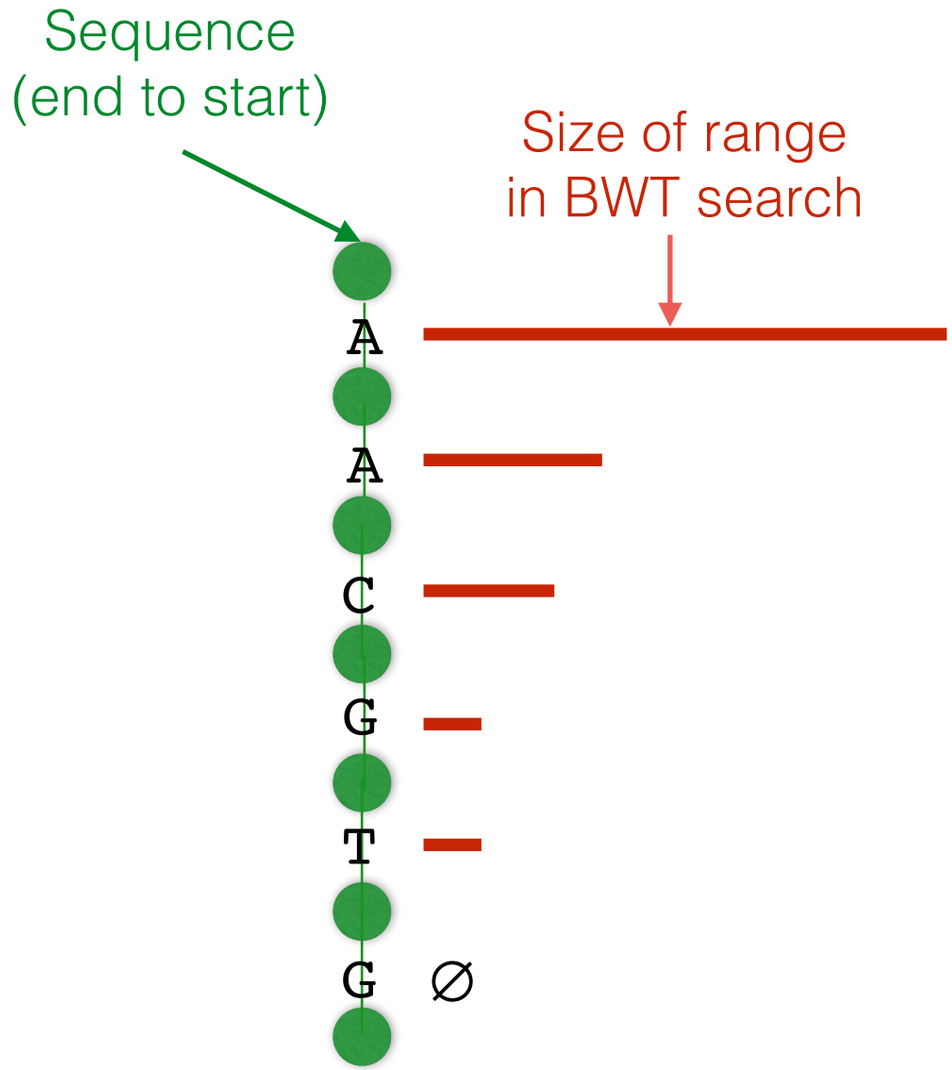
Ben Langmead, Cole Trapnell, Mihai Pop and Steven L Salzberg, *Genome Biology* 2009, 10:R25

Bowtie Features

- Extends basic BWT search to handle mismatches
  - Will find an exact match if it exists.
  - Might not find the best inexact match
- Two innovations:
  - quality-aware backtracking
  - double indexing

SOAP-like Policy (-v N)

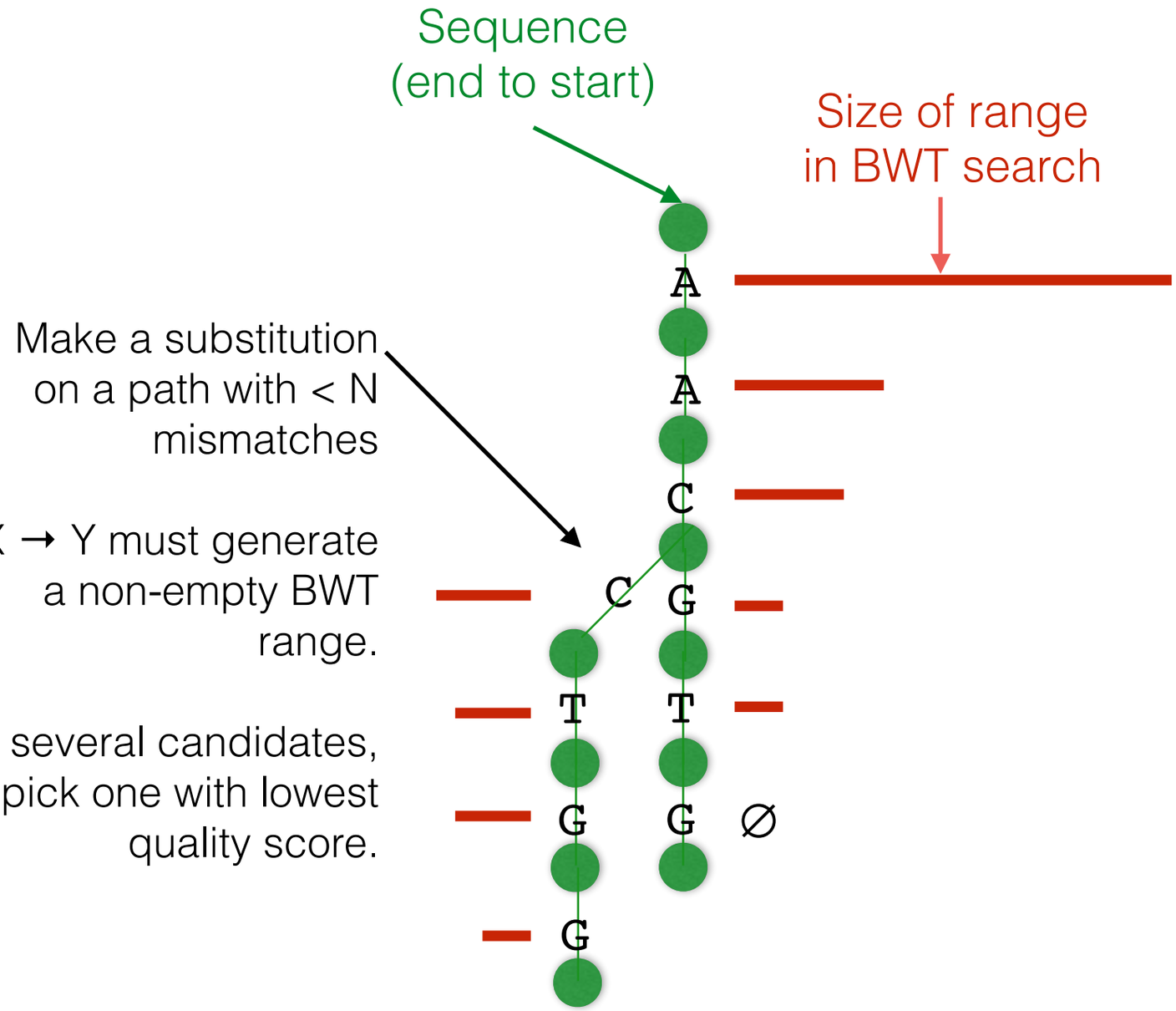
Don't allow more than N mismatches. $N \in \{0,1,2,3\}$



Repeat this until you hit 175 backtracks, or you find K matches (K is a parameter).

SOAP-like Policy (-v N)

Don't allow more than N mismatches. $N \in \{0,1,2,3\}$

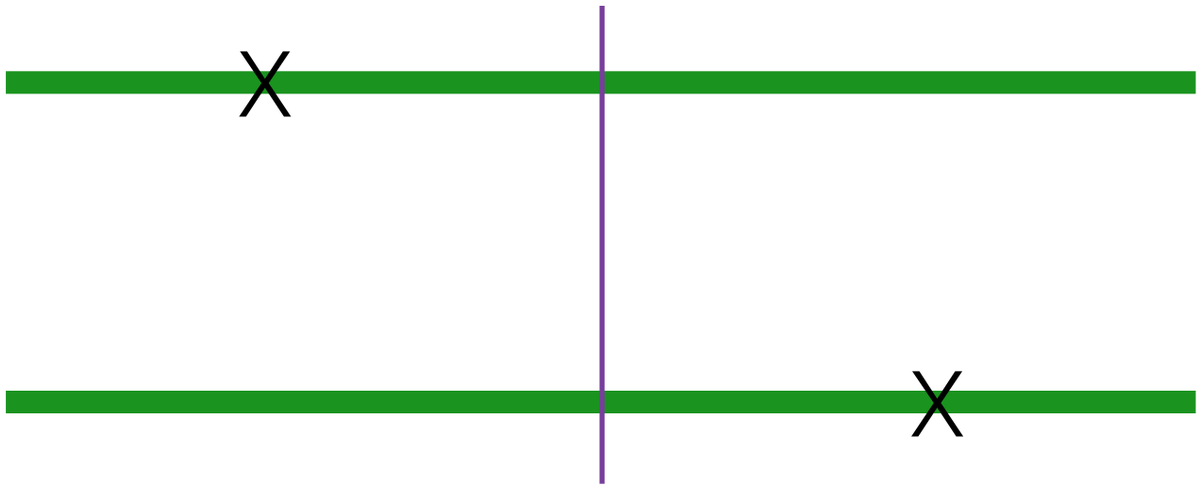


Repeat this until you hit 175 backtracks, or you find K matches (K is a parameter).

Using Forward and Mirror Index

Use a BWT of both forward and reverse of string:

If mutation in first half of read, then the forward index can walk at least |read| / 2 before backtracking.

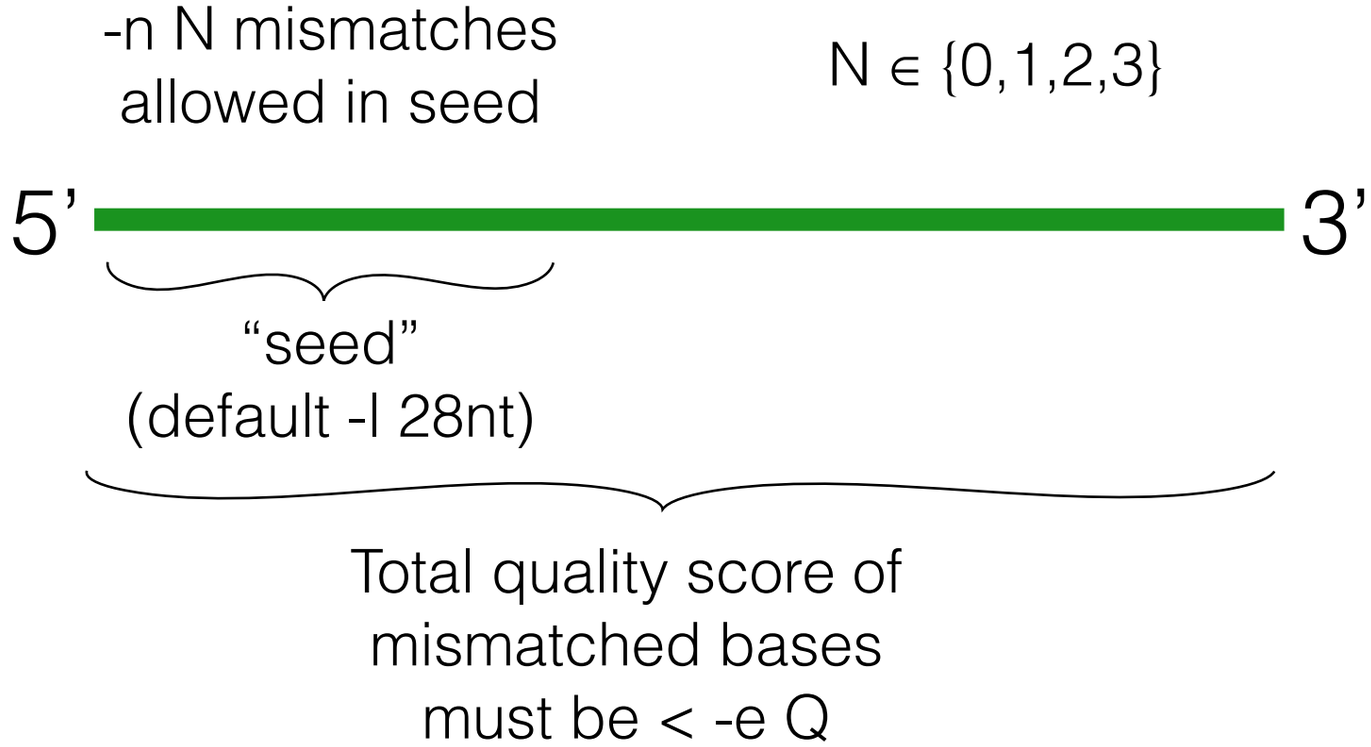


If mutation in second half of read, then the second index can walk at least |read| / 2 before backtracking.

Try both the forward and reverse indexes; will avoid a lot of backtracking because you will have narrowed the BWT range a lot by the time you start backtracking.

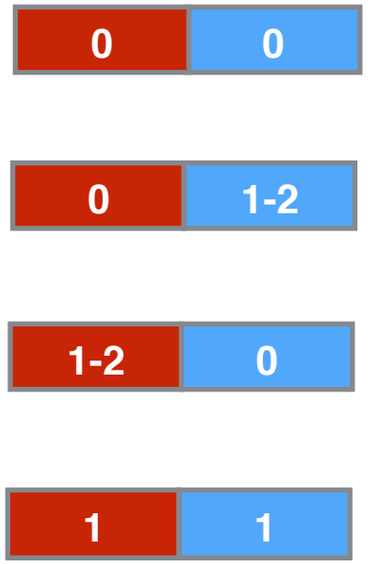
Maq-like (default) Approach

Up to N mismatches allowed in “seed”, which is prefix of the read

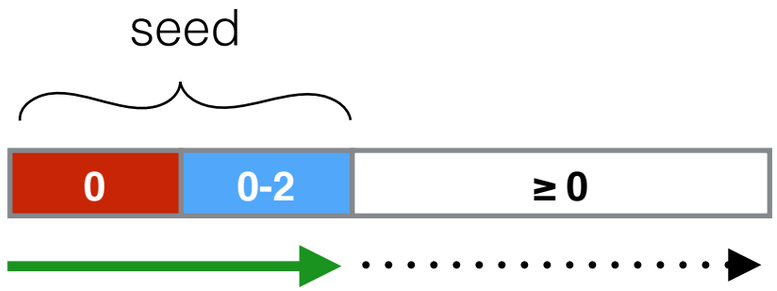


3 Phases

4 possible cases for distribution of 0-2 mutations within the seed:

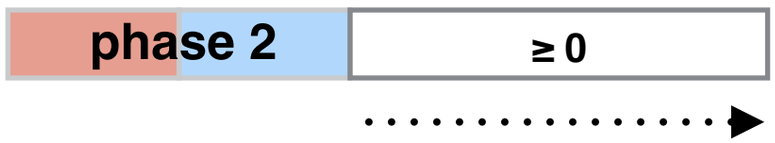


Same backtracking scheme used to handle mismatches

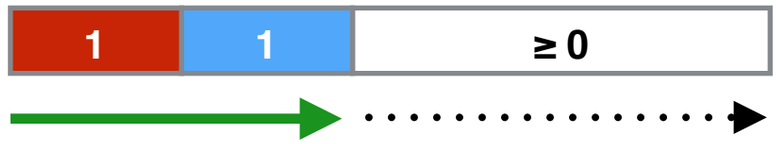


phase 1 (using mirror index)

phase 2 (using forward index)



phase 3a (using mirror index), extending phase 2 alignments outside of seed



phase 3b (using mirror index)

Bowtie Performance

	Platform	CPU time	Wall clock time	Reads mapped per hour (millions)	Peak virtual memory footprint (megabytes)	Bowtie speed-up	Reads aligned (%)
Bowtie -v 2	Server	15 m 7 s	15 m 41 s	33.8	1,149	-	67.4
SOAP		91 h 57 m 35 s	91 h 47 m 46 s	0.10	13,619	351×	67.3
Bowtie	PC	16 m 41 s	17 m 57 s	29.5	1,353	-	71.9
Maq		17 h 46 m 35 s	17 h 53 m 7 s	0.49	804	59.8×	74.7
Bowtie	Server	17 m 58 s	18 m 26 s	28.8	1,353	-	71.9
Maq		32 h 56 m 53 s	32 h 58 m 39 s	0.27	804	107×	74.7

Bowtie 2

Features

- Supports gapped alignment
- Uses bi-directional BWT instead of two separate BWTs
- Supports paired-end alignment
  - Align one end as normal
  - Find the window where the other end could go
  - Do dynamic programming alignment step to align the other end of the pair within this window

1. Extract seeds

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. Align with FM Index

Seeds

+, 0: CCAGTAGCTCTCAGCC

+, 10: TCAGCCTTATTTTACC

+, 20: TTTACCCAGGCCTGTA

-, 0: TACAGGCCTGGGTAAA

-, 10: GGTAAAATAAGGCTGA

-, 20: GGCTGAGAGCTACTGG

→

Ungapped alignment with FM Index

→

Seed alignments (as Burrows-Wheeler ranges)

{ [211, 212], [212, 214] }

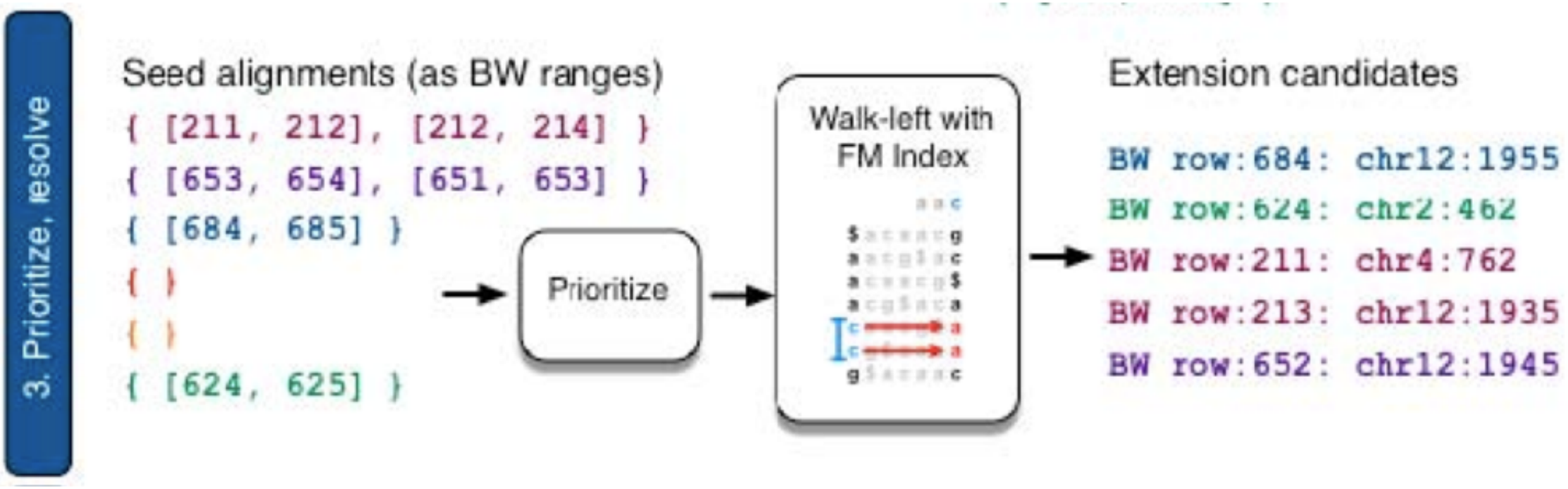
{ [653, 654], [651, 653] }

{ [684, 635] }

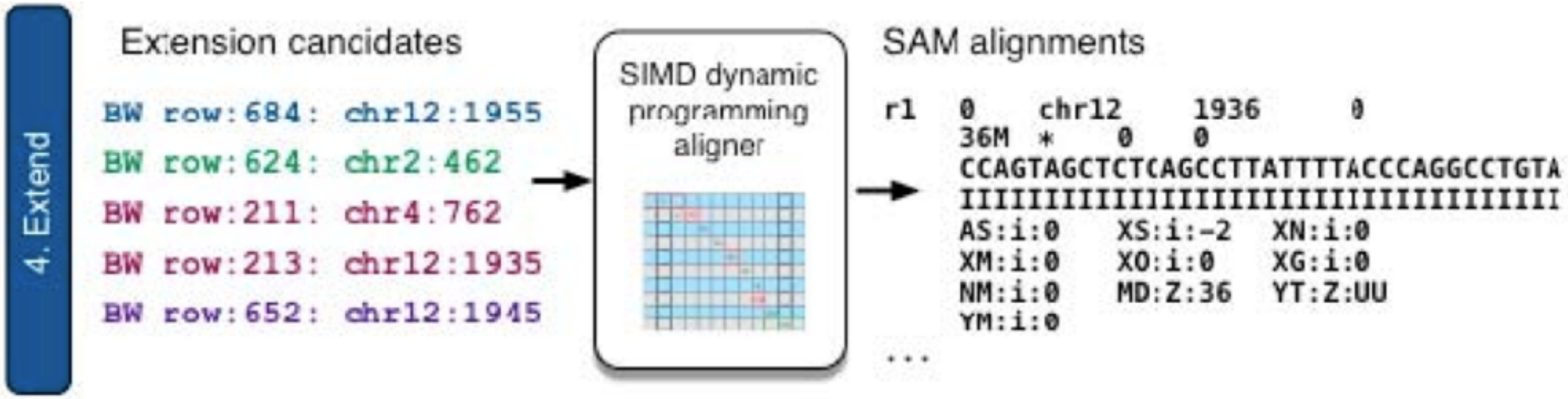
{ }

{ }

{ [624, 625] }

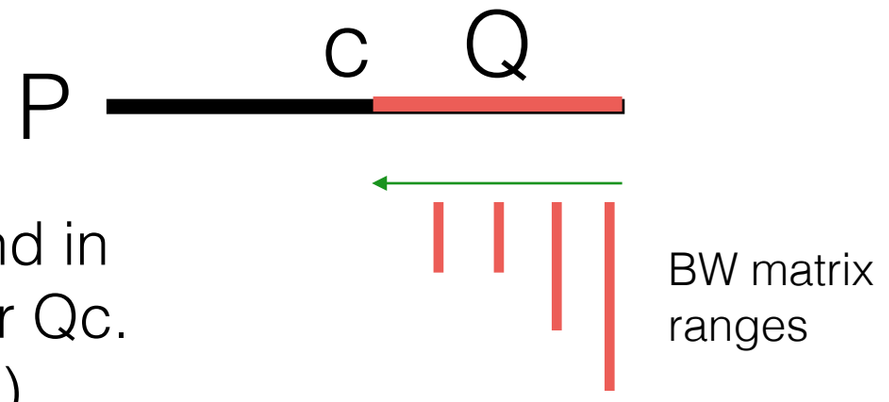


priority of row $i = 1/ r_i^2$, where r_i is the # of sequences in the range containing row i.



Bi-directional BWT

Standard “forward” BWT search:

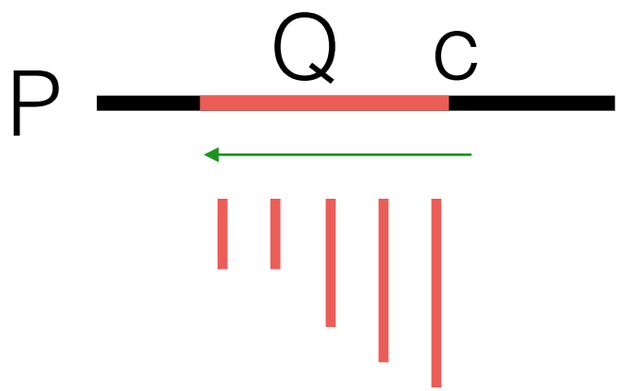


Given range for Q, can find in constant time the range for Qc. (Use LF mapping, etc.)

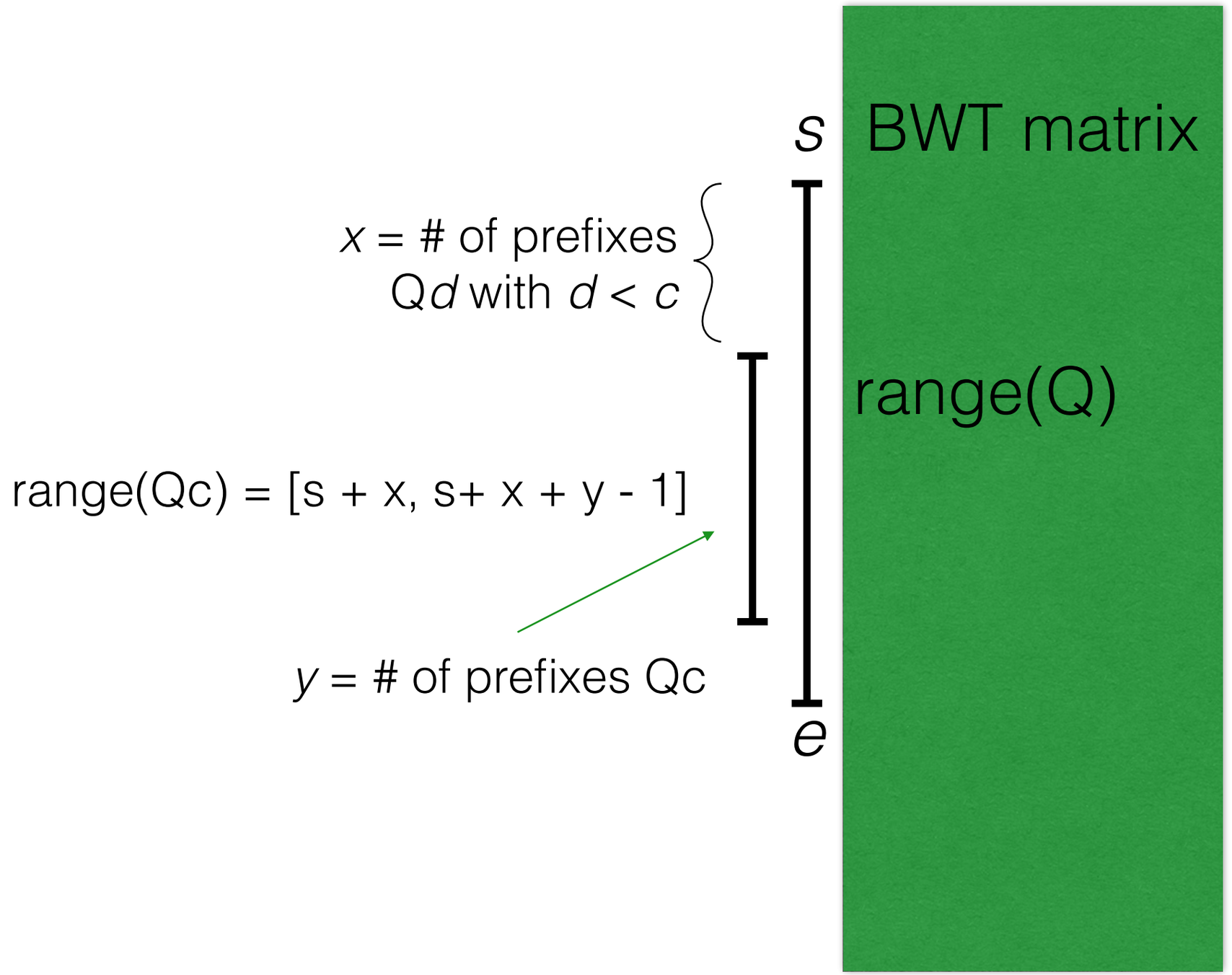
Thm. Given range(Q), reverseRange(Q), can find in O(|∑|)-time:

- range(Qc)
- range(cQ) ⟵ easy (this is the standard BWT search)
- reverseRange(cQ)
- reverseRange(Qc)

easy (this is the standard BWT search applied to the reverse string)



Computing Range(Qc)



Computing Range(Qc)

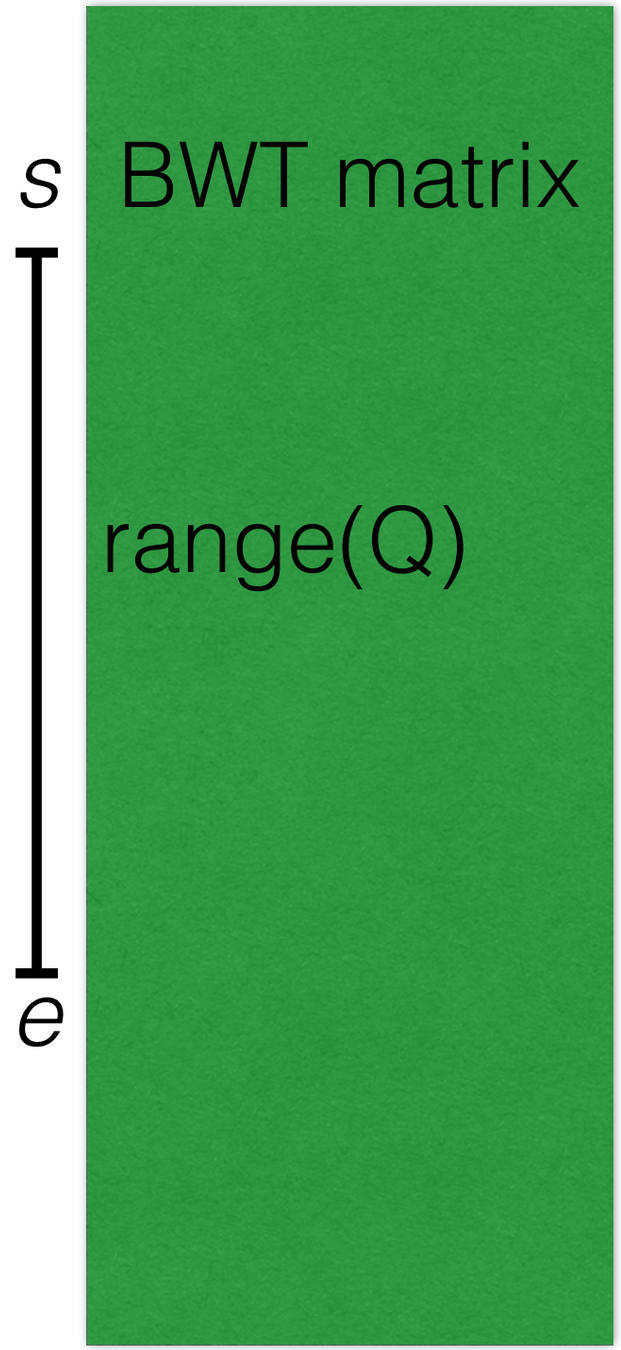
$x = \sum_d \text{reverseRange}((Qd)^R) = \text{reverseRange}(dQ^R)$

because the size of the range of Qc in T = the size of the range of $(Qc)^R$ in T^R

x = # of prefixes Qd with $d < c$

range(Qc) = [s + x, s+ x + y - 1]

y = # of prefixes Qc



Computing Range(Qc)

$x = \sum_d \text{reverseRange}((Qd)^R) = \text{reverseRange}(dQ^R)$

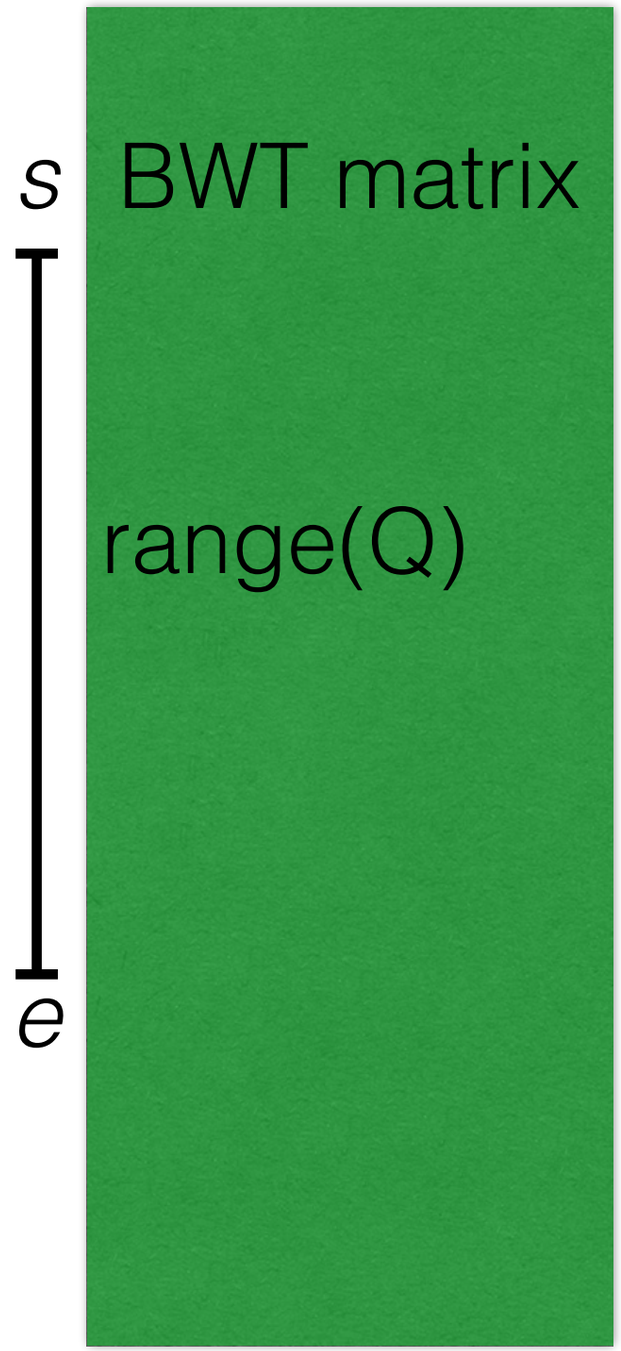
because the size of the range of Qc in T = the size of the range of $(Qc)^R$ in T^R

x = # of prefixes Qd with $d < c$

range(Qc) = [s + x, s+ x + y - 1]

y = # of prefixes Qc

$y = \text{reverseRange}((Qc)^R)$



Computing Range(Qc)

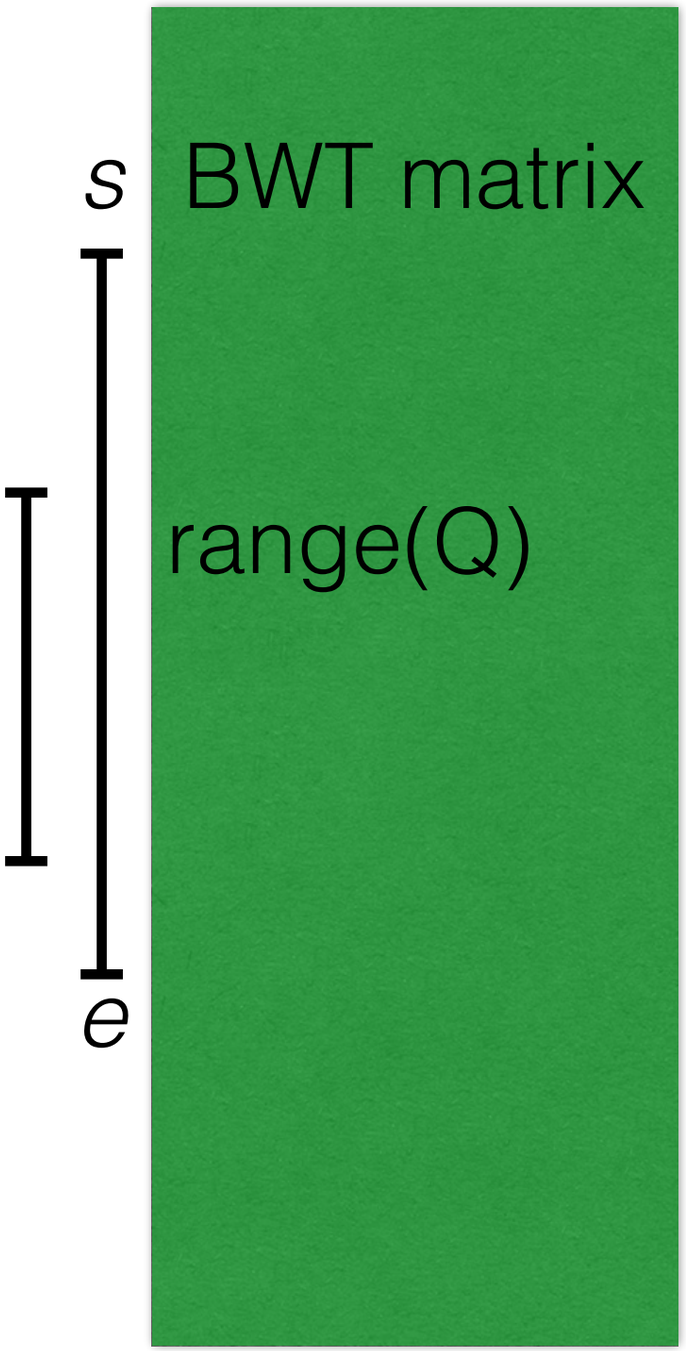
$x = \sum_d \text{reverseRange}((Qd)^R) = \text{reverseRange}(dQ^R)$

because the size of the range of Qc in T = the size of the range of $(Qc)^R$ in T^R

x = # of prefixes Qd with $d < c$

range(Qc) = [s + x, s+ x + y - 1]

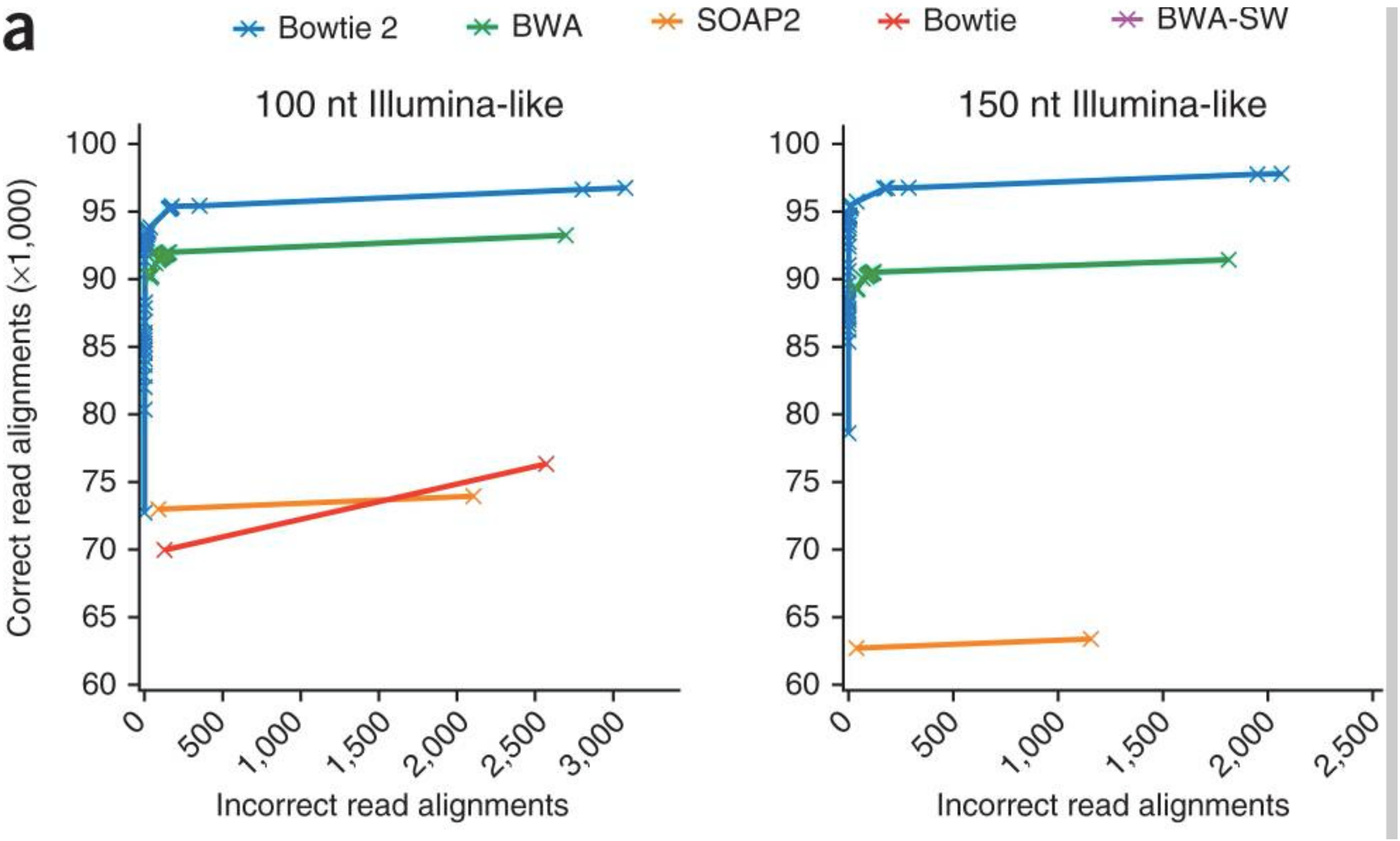
y = # of prefixes Qc



$y = \text{reverseRange}((Qc)^R)$

Can extend reverseRange(Q) to reverseRange(dQ) with 1 BWT LF step

Accuracy Results

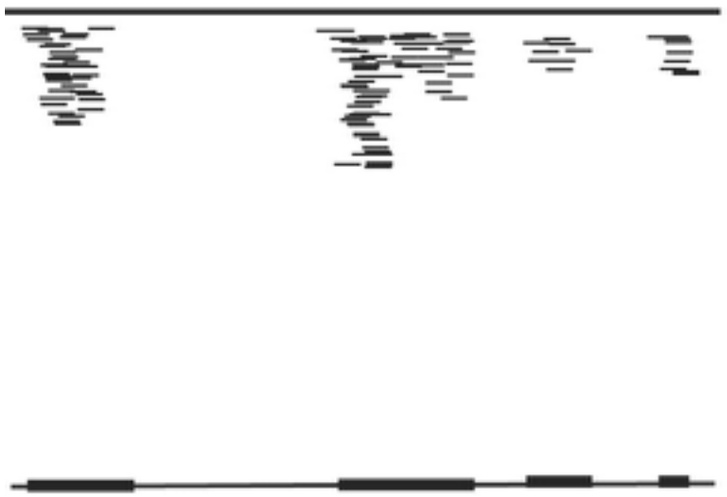


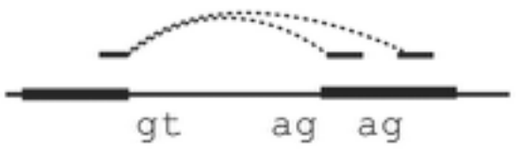
TopHat

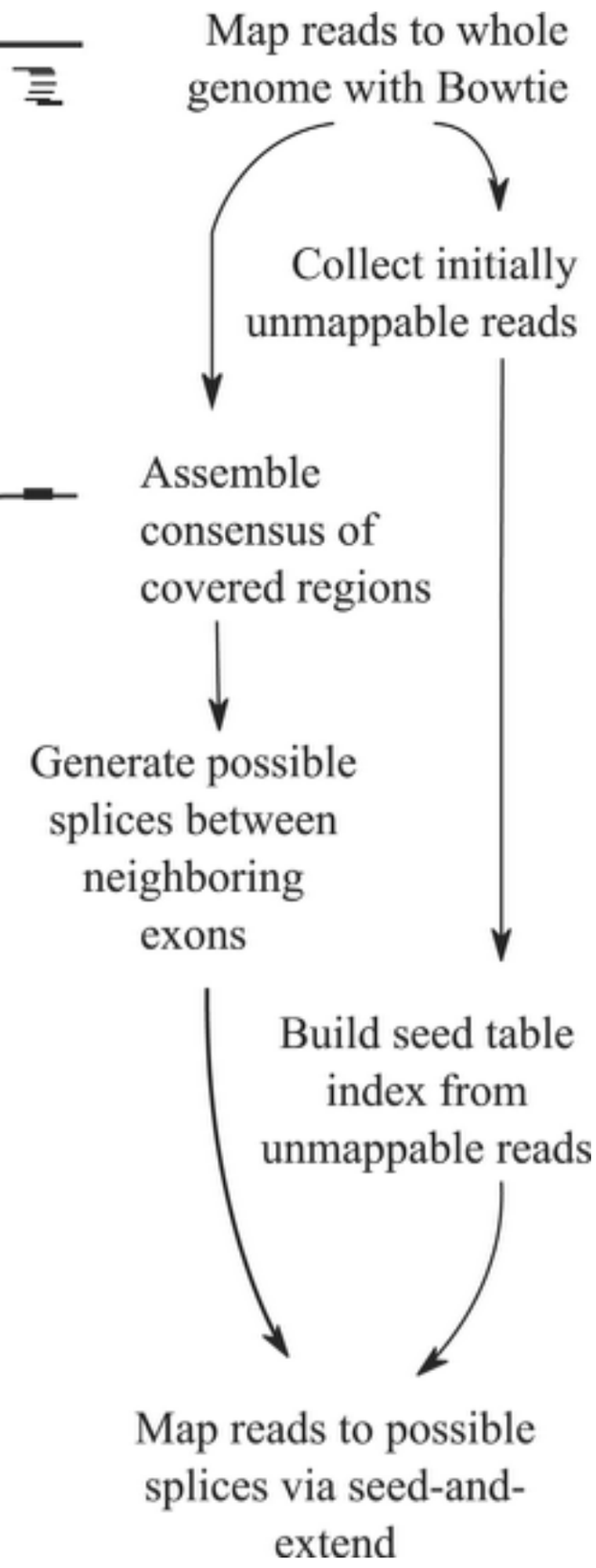
Cole Trapnell, Lior Pachter and Steven L. Salzberg· TopHat: discovering splice junctions with RNA-Seq. Bioinformatics (2009) 25 (9): 1105-1111.

TopHat

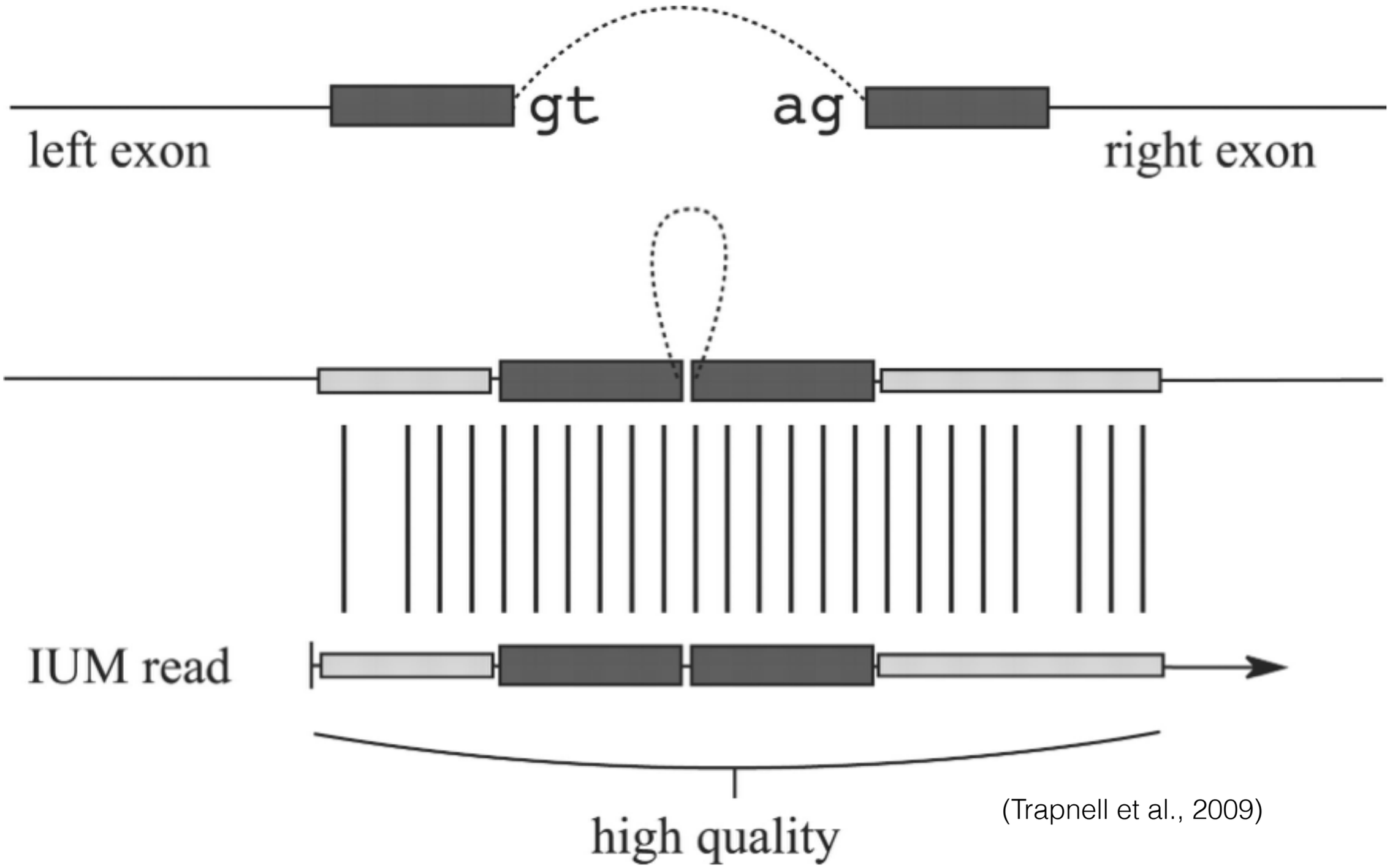
- Find Exons:
  - Map reads
  - Assemble exons (using “Maq”)
  - Extend exons by 45bp
  - Join exons closer than 6bp
- Find splices:
  - Find every pair of donor/acceptor sites that are close enough
  - Look at 2kmers spanning that junction (where k appears on each side; k=5)
  - Use a 2k-mer lookup table to find those reads







TopHat Approach



(Trapnell et al., 2009)