CSC 334: TOPICS IN COMPUTATIONAL BIOLOGY

"Algorithms for Genomic Data"

Fall 2015

Smith College

Instructor: Prof. Sara Sheehan

Outline: 9/25

- Bowtie paper

BWT alignment vocabulary

- **Oligomer**: like a k-mer
- **Re-sequencing**: in this context, sequencing a new individual from the same species
- **Reversible permutation**: rearrangement of the characters of a string such that the original string is recoverable
- **Depth-first search**: (board)

Bowtie description of BWT

The Burrows-Wheeler transformation of a text T, BWT(T), is constructed as follows. The character $ is appended to T, where $ is not in T and is lexicographically less than all characters in T. The Burrows-Wheeler matrix of T is constructed as the matrix whose rows comprise all cyclic rotations of T$. The rows are then sorted lexicographically. BWT(T) is the sequence of characters in the rightmost column of the Burrows-Wheeler matrix (Figure 1a). BWT(T) has the same length as the original text T.

Bowtie: Figure 1

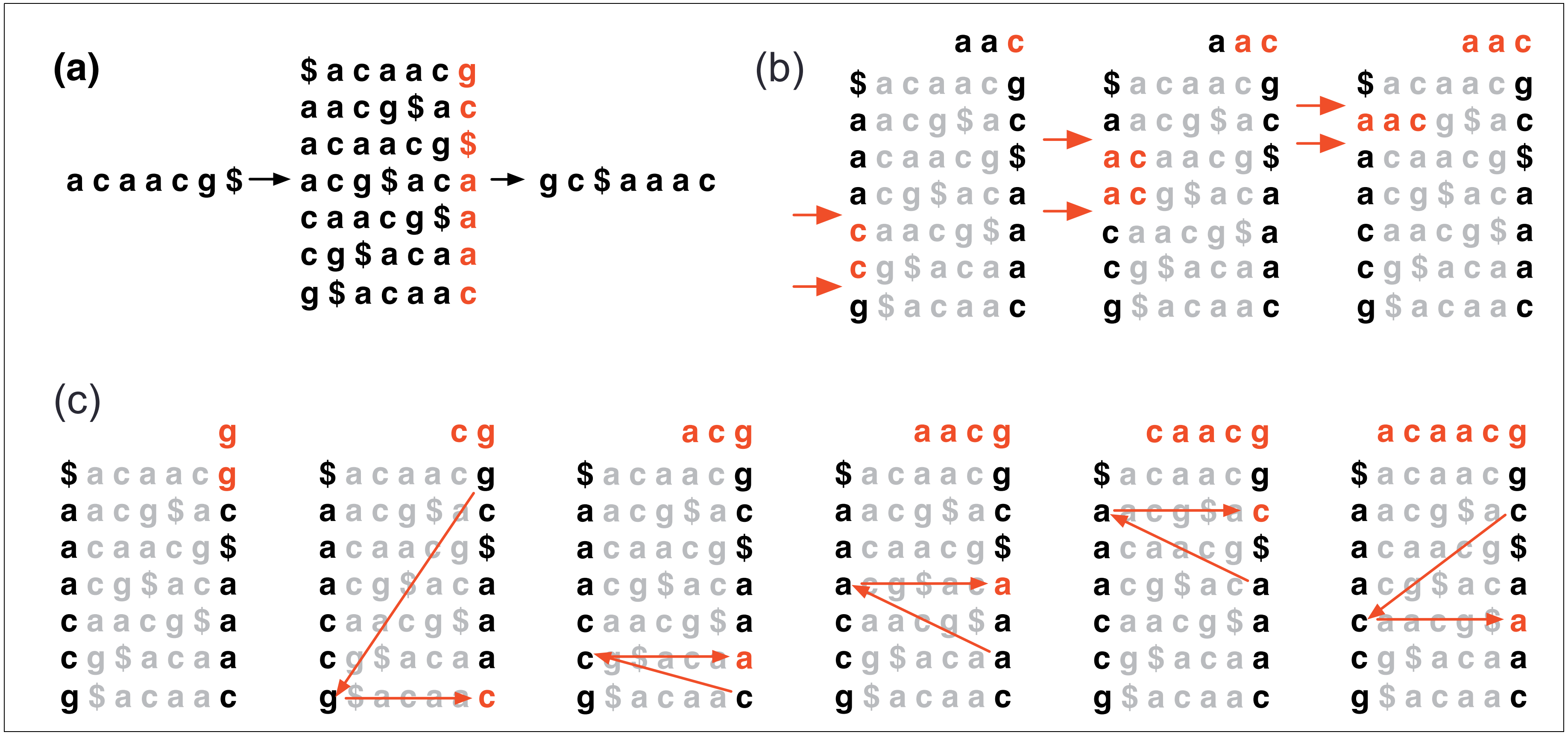

Figure 1

Burrows-Wheeler transform. **(a)** The Burrows-Wheeler matrix and transformation for 'acaacg'. **(b)** Steps taken by EXACTMATCH to identify the range of rows, and thus the set of reference suffixes, prefixed by 'aac'. **(c)** UNPERMUTE repeatedly applies the last first (LF) mapping to recover the original text (in red on the top line) from the Burrows-Wheeler transform (in black in the rightmost column).

Bowtie: Figure 2

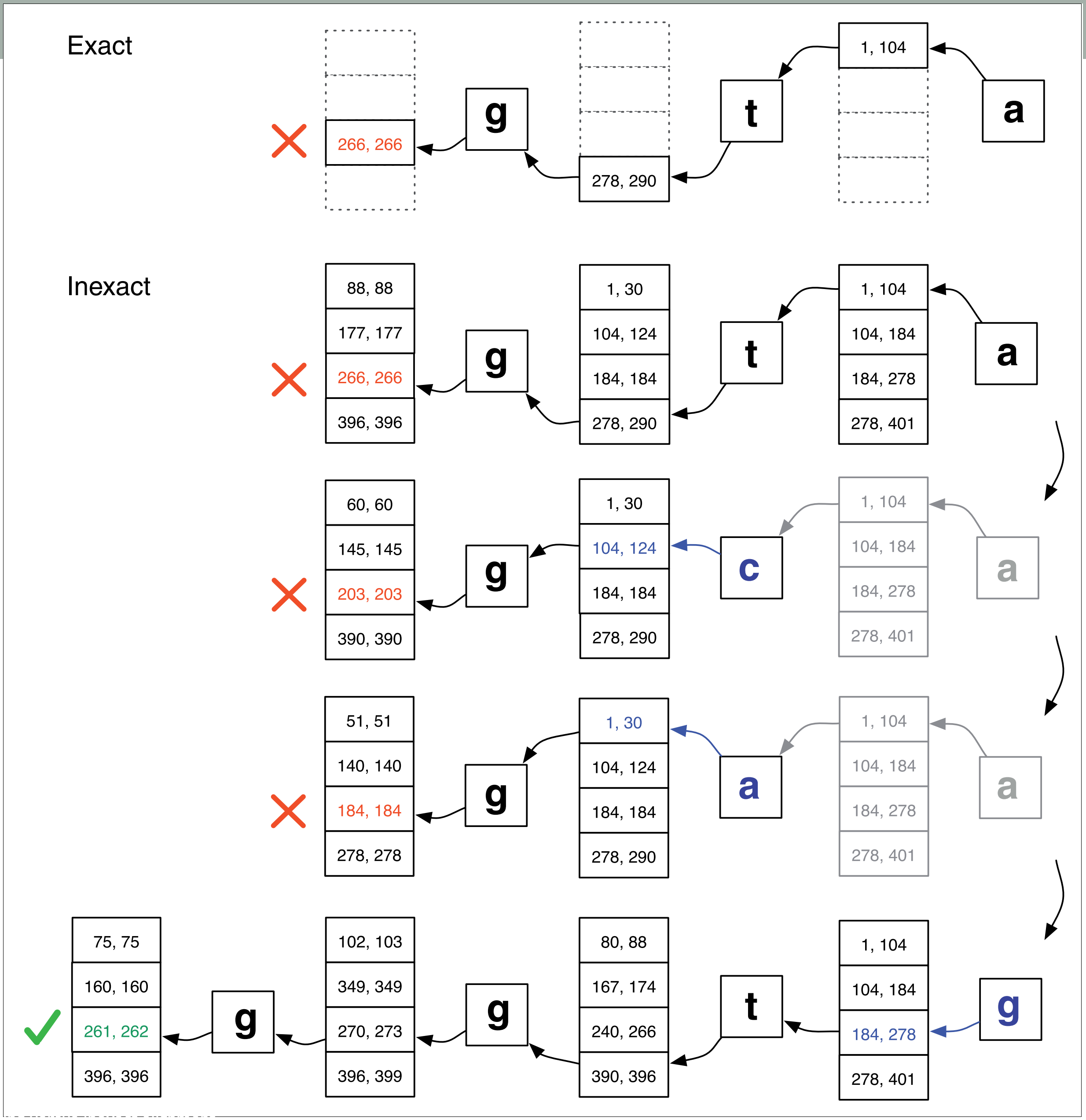

Figure 2
Exact matching versus inexact alignment. Illustration of how EXACTMATCH (top) and Bowtie's aligner (bottom) proceed when there is no exact match for query 'ggta' but there is a one-mismatch alignment when 'a' is replaced by 'g'. Boxed pairs of numbers denote ranges of matrix rows beginning with the suffix observed up to that point. A red X marks where the algorithm encounters an empty range and either aborts (as in EXACTMATCH) or backtracks (as in the inexact algorithm). A green check marks where the algorithm finds a nonempty range delimiting one or more occurrences of a reportable alignment for the query.

Bowtie: Figure 3

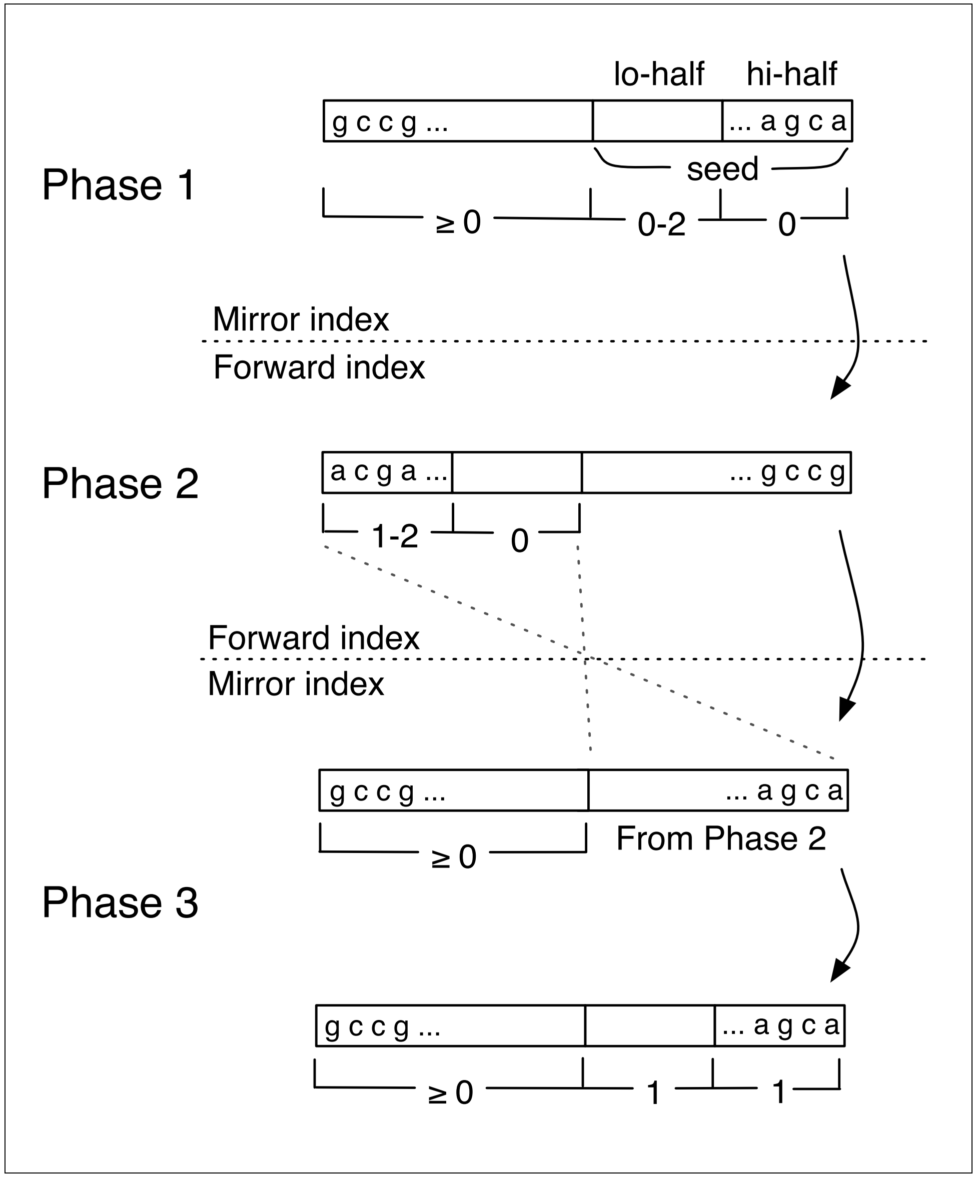

Figure 3
The three phases of the Bowtie algorithm for the Maq-like policy. A three-phase approach finds alignments for two-mismatch cases 1 to 4 while minimizing backtracking. Phase 1 uses the mirror index and invokes the aligner to find alignments for cases 1 and 2. Phases 2 and 3 cooperate to find alignments for case 3: Phase 2 finds partial alignments with mismatches only in the hi-half, and phase 3 attempts to extend those partial alignments into full alignments. Finally, phase 3 invokes the aligner to find alignments for case 4.

Comparison with other aligners

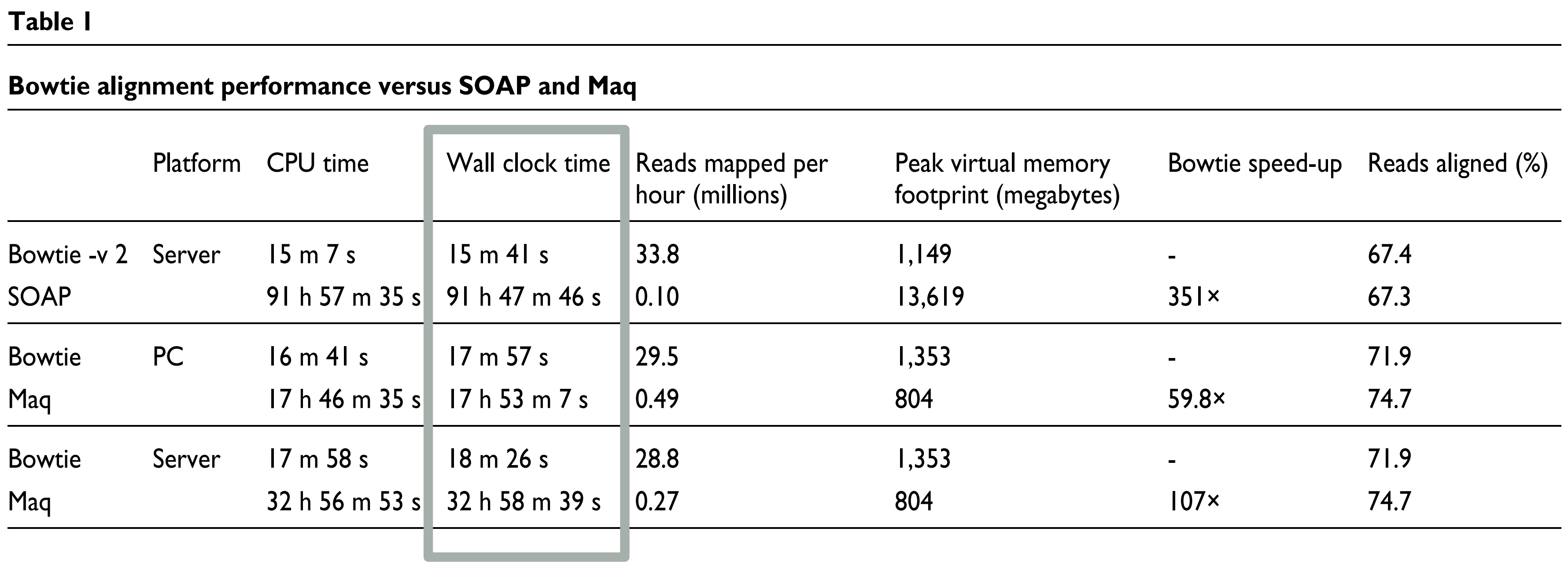

Table 1

Bowtie alignment performance versus SOAP and Maq

	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

Comparison with other aligners

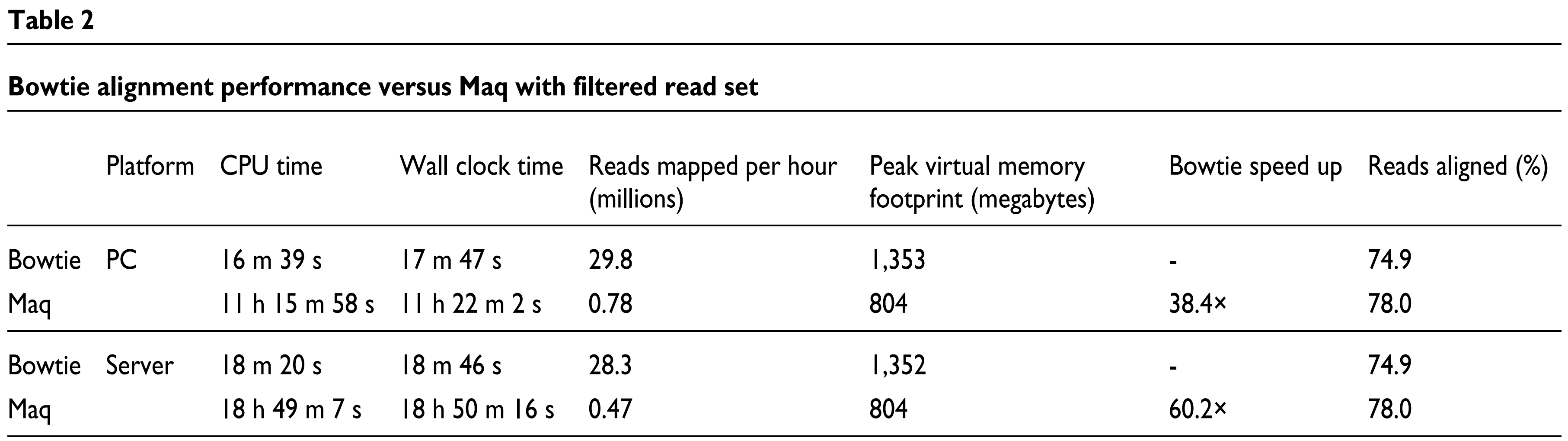

Table 2

Bowtie alignment performance versus Maq with filtered read set

	Platform	CPU time	Wall clock time	Reads mapped per hour (millions)	Peak virtual memory footprint (megabytes)	Bowtie speed up	Reads aligned (%)
Bowtie	PC	16 m 39 s	17 m 47 s	29.8	1,353	-	74.9
Maq		11 h 15 m 58 s	11 h 22 m 2 s	0.78	804	38.4×	78.0
Bowtie	Server	18 m 20 s	18 m 46 s	28.3	1,352	-	74.9
Maq		18 h 49 m 7 s	18 h 50 m 16 s	0.47	804	60.2×	78.0