TOPIC 7:

Read Mapping

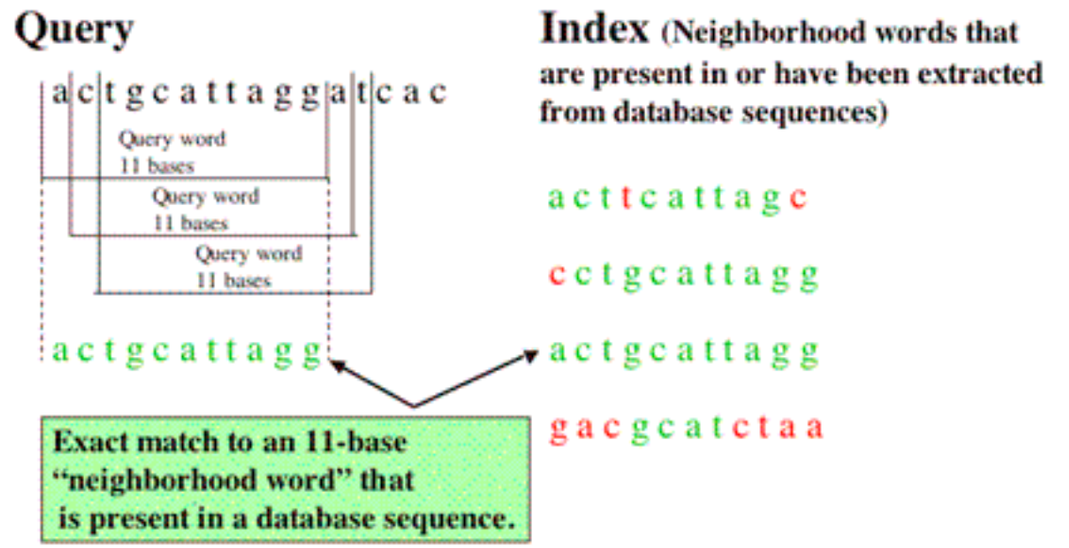
Outline

- Algorithm at the heart of the dominant short read mapper
  - Suffix-trie structures
- Long read alignment - challenges and approaches
- Anatomy of the BAM/SAM file format

BLAST - Best Local Alignment Search Tool

Why not use BLAST for short read data?

- Typically takes 0.1 to 1 second to search 1 sequence against a database
- 60 million reads equates to ~70 CPU days



Approaches to align reads

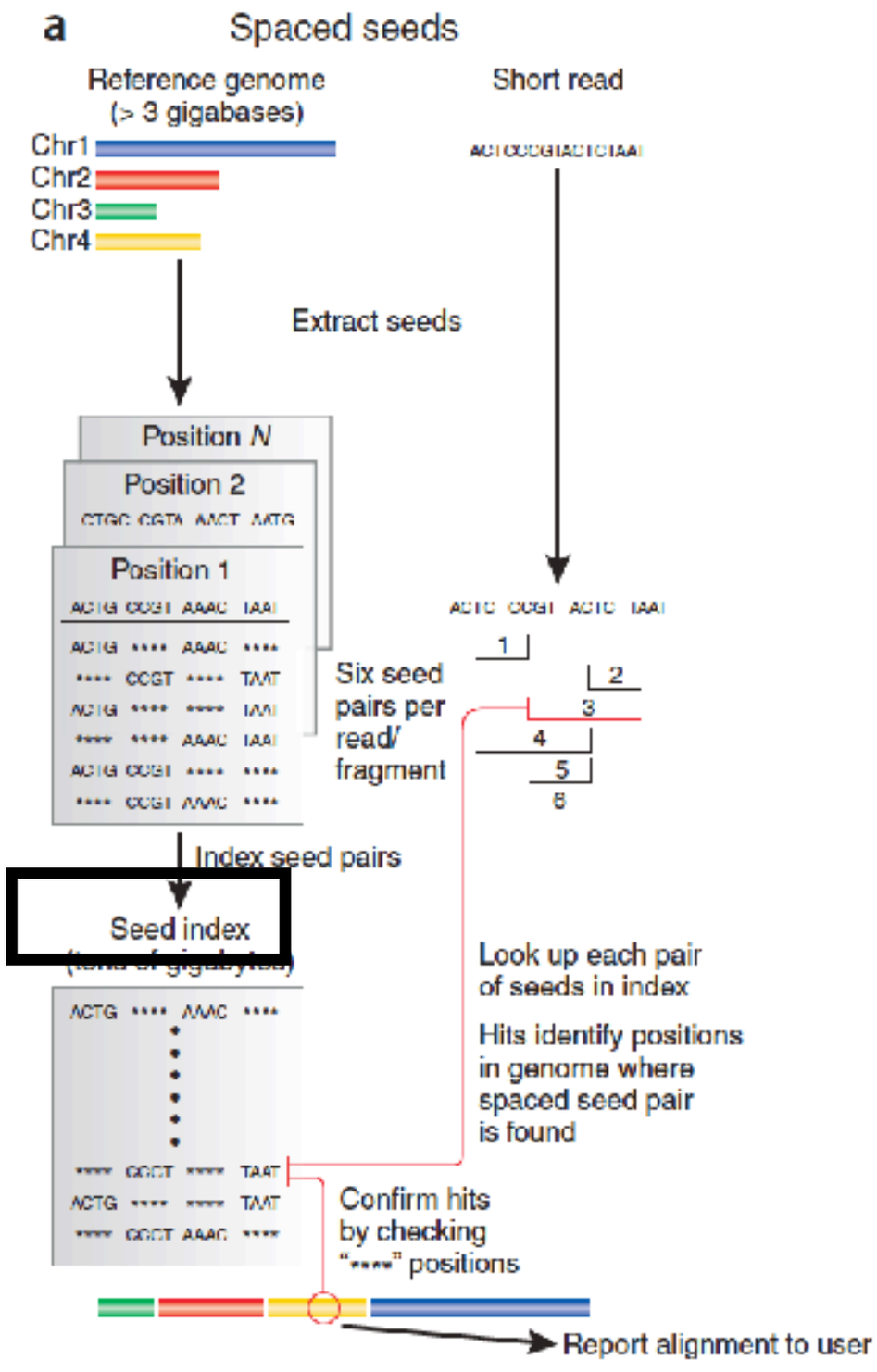


Figure modified from Trapnell & Salzberg 2009

Hashed seed-extend algorithms

Builds a hash of seeds from the reference genome

Then a two step process:

- Identify a match to the seed sequence in the reference

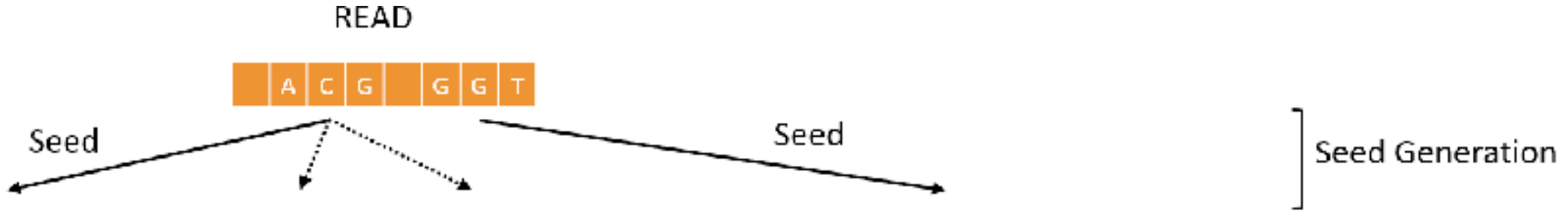
- Extend match using sensitive (but slow) Smith-Waterman algorithm

Seed-extend algorithm

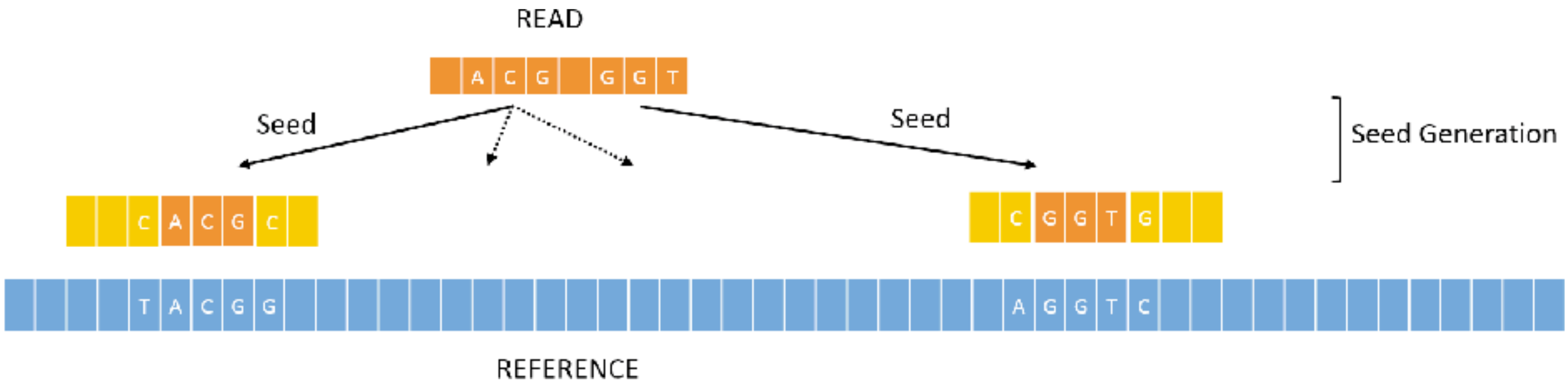
READ

REFERENCE

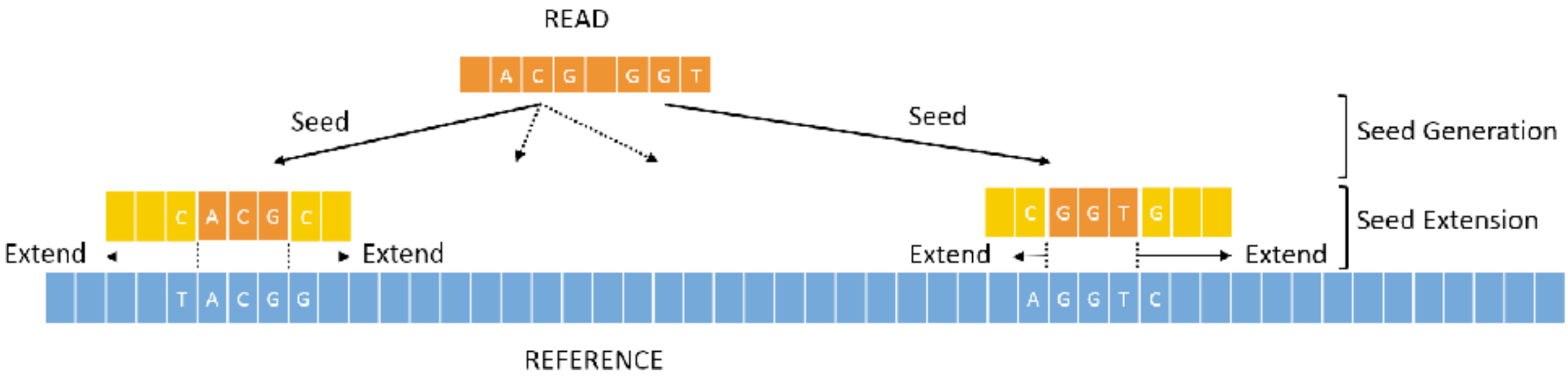
Seed-extend algorithm



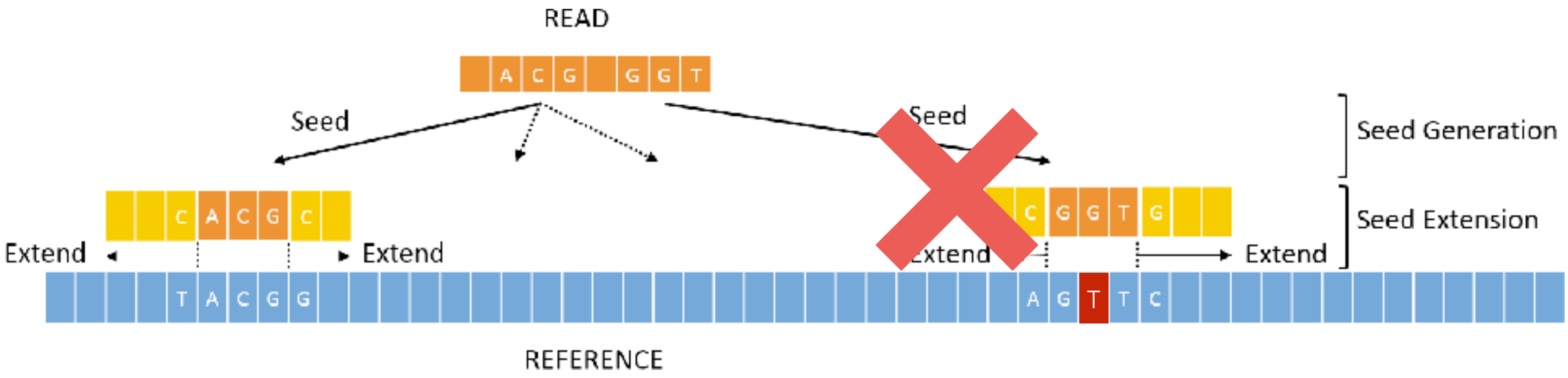
Seed-extend algorithm



Seed-extend algorithm



Seed-extend algorithm



Spaced seeds

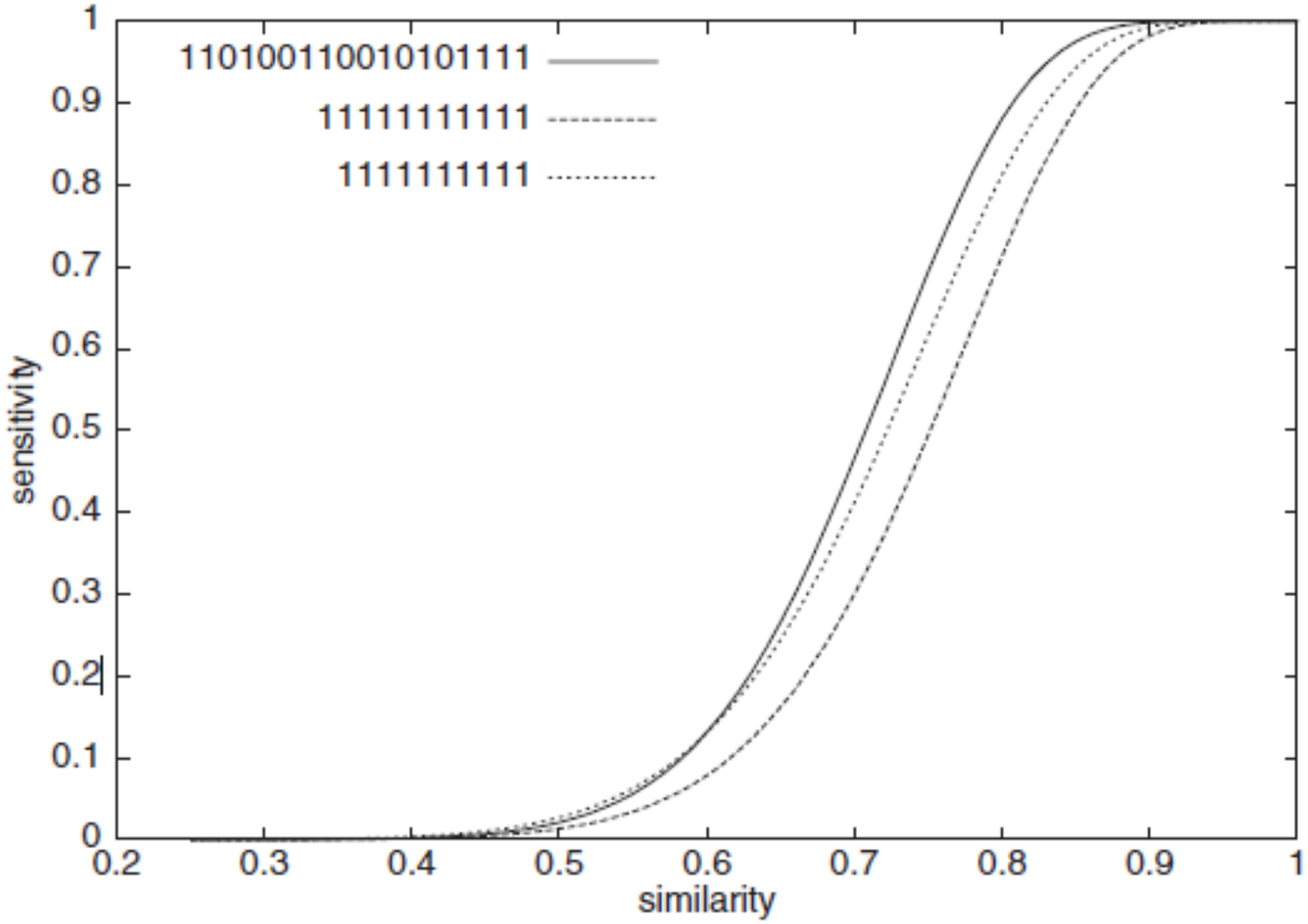
To increase sensitivity we can used spaced-seeds:

```
111111111          Consecutive seed template with length 9bp
GATAGCTAGCTAAT     Reference
  AGCTAGCTA        Query
```

```
101011010111011    Consecutive seed template with weight 9bp
GATAGCTAGCTAAT     Reference
GATAGCGAGCTAAT     Query
```

Spaced seeds

- To increase sensitivity we can used spaced-seeds:



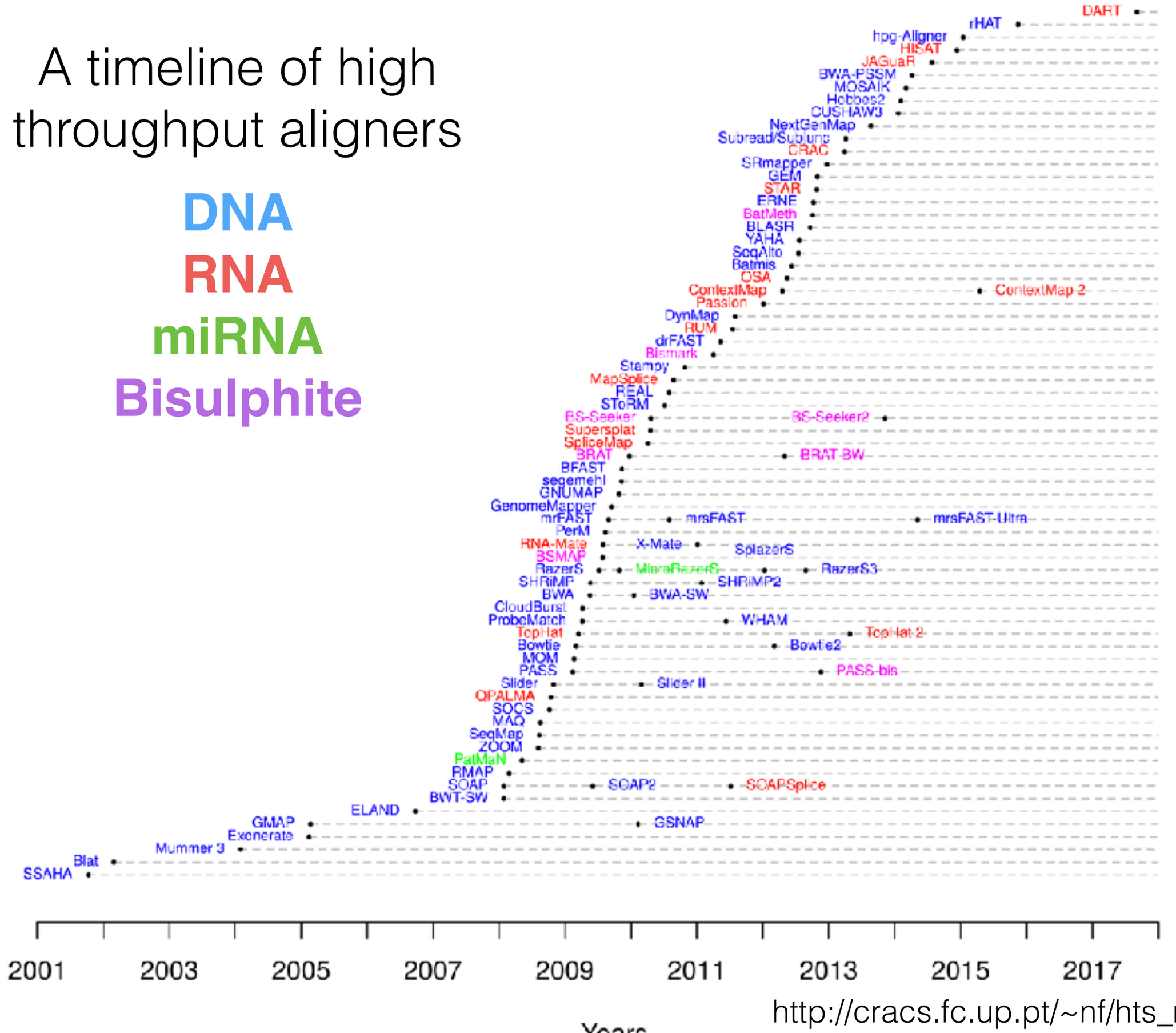
A timeline of high throughput aligners

DNA

RNA

miRNA

Bisulphite



http://cracs.fc.up.pt/~nf/hts_mappers/

Approaches to align short reads

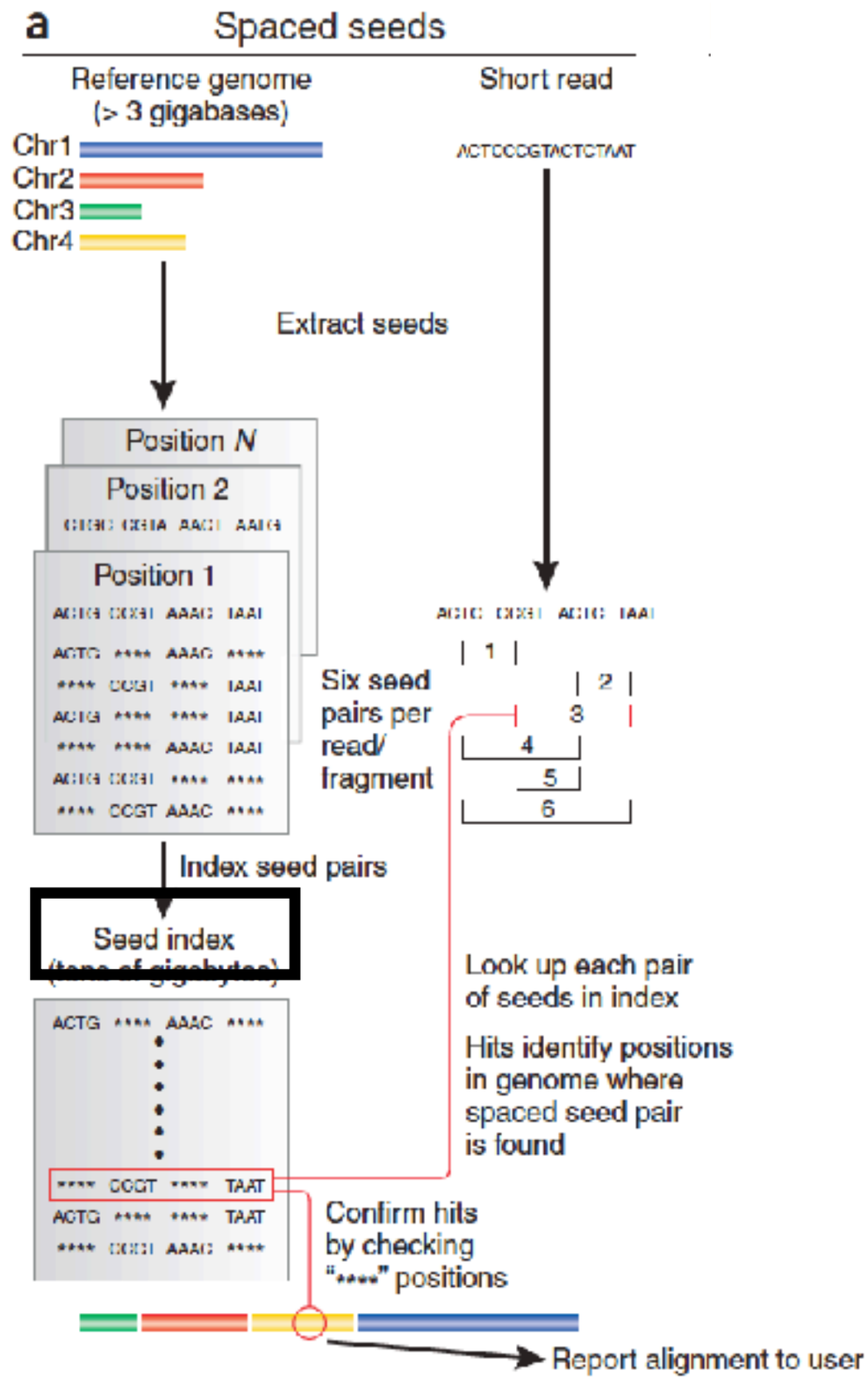


Figure modified from Trapnell & Salzberg 2009

Approaches to align short reads

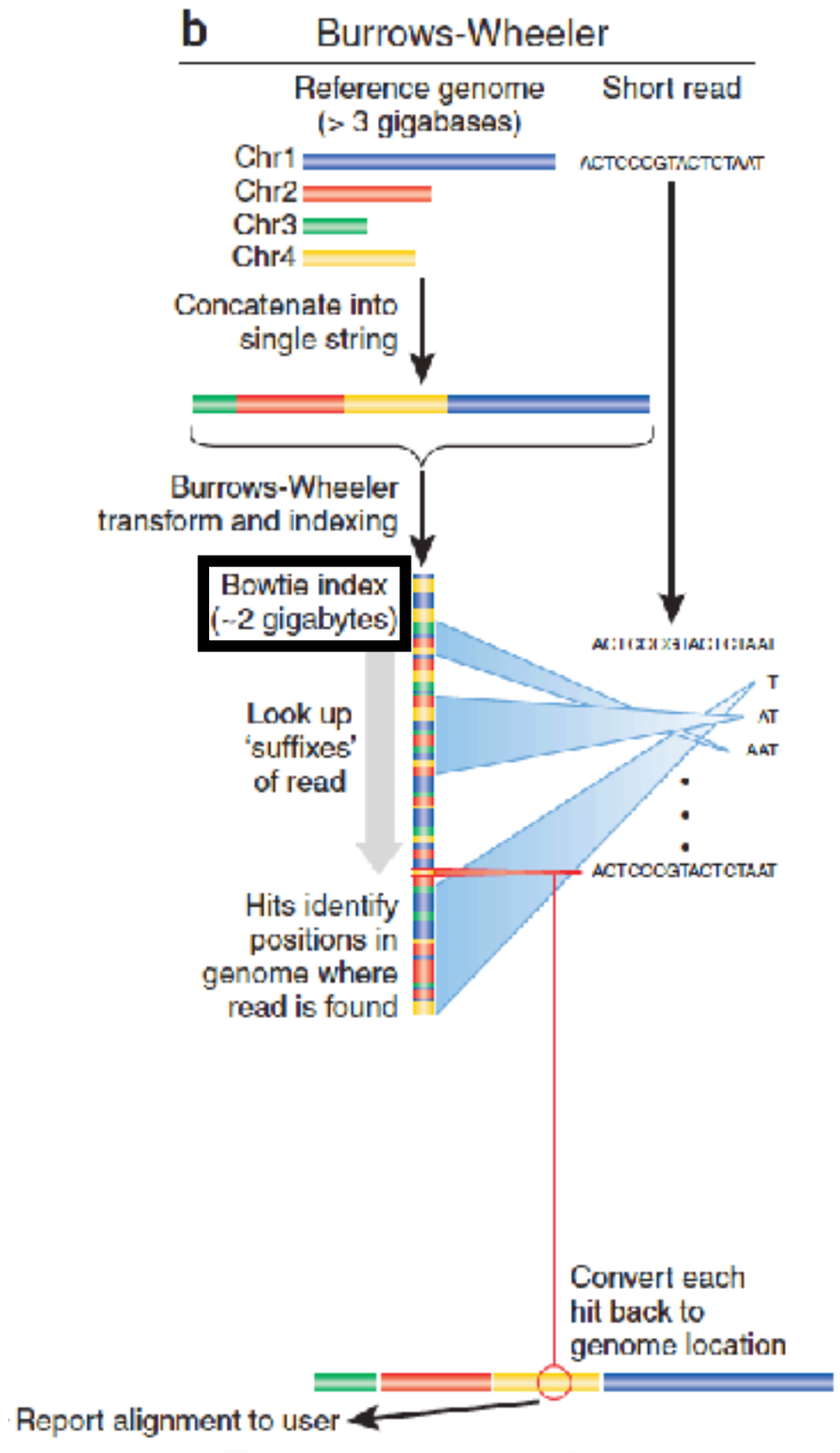
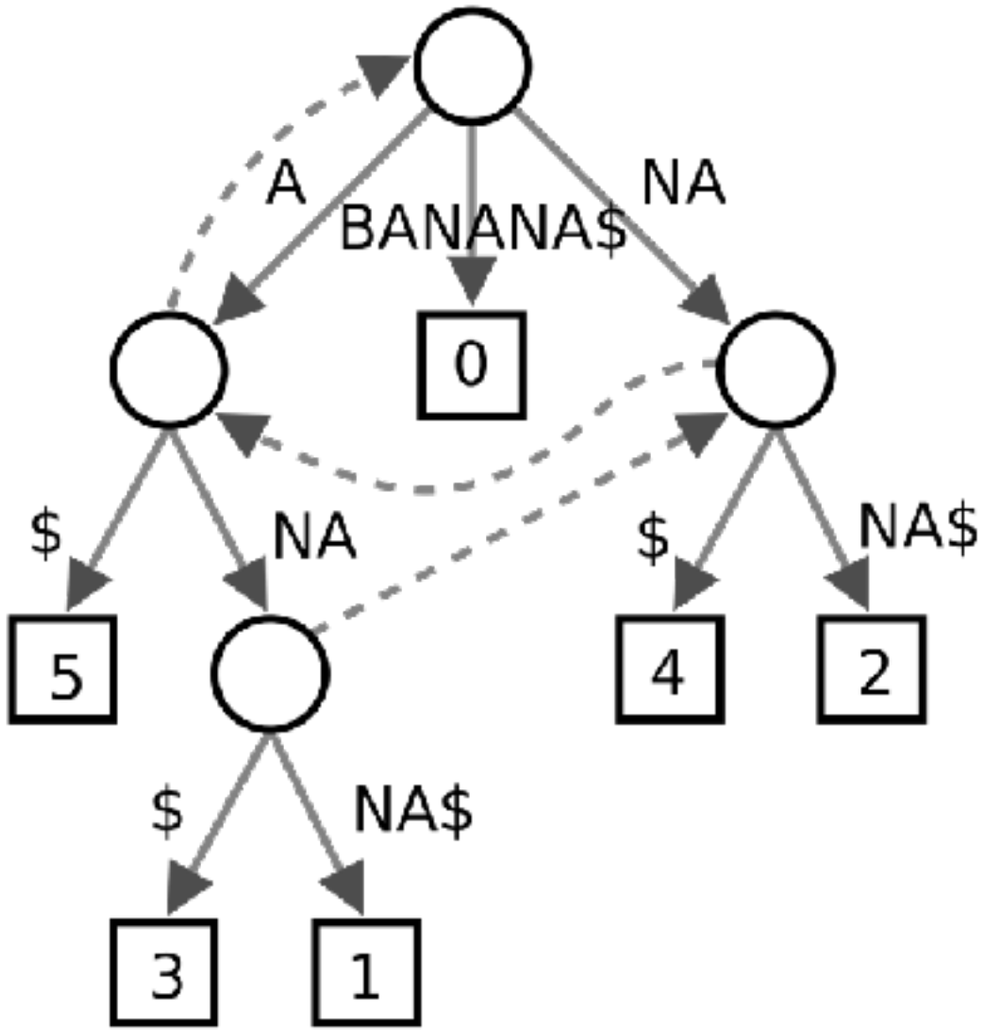


Figure modified from Trapnell & Salzberg 2009

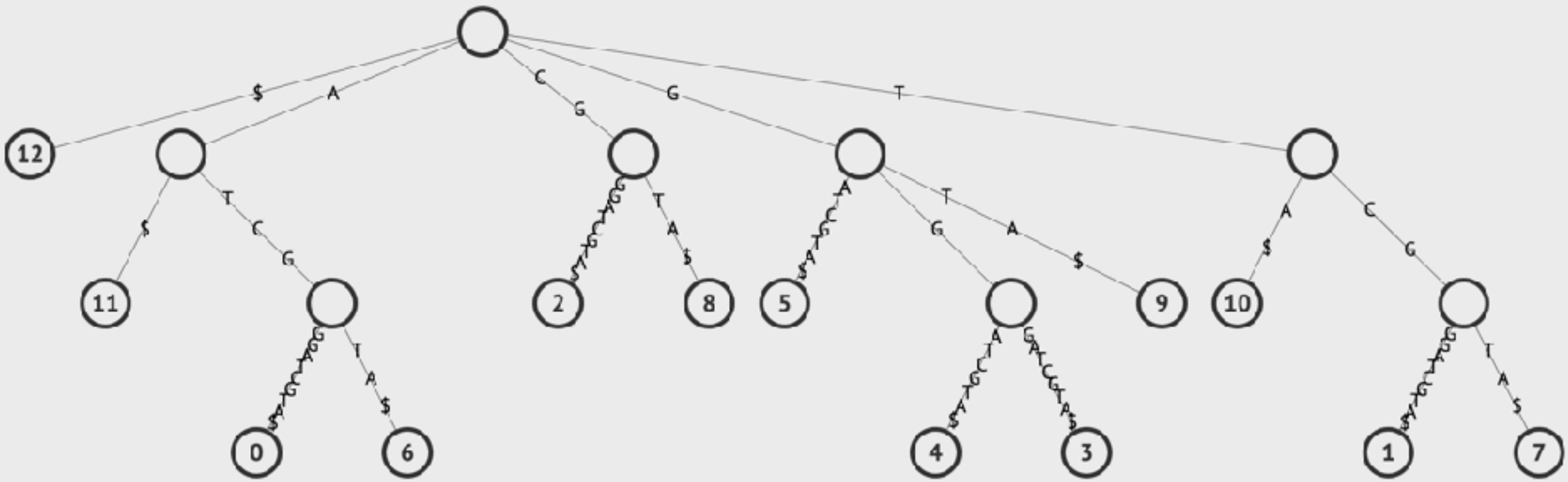
Suffix-Trie

A data structure that contains all suffixes and their locations in the text



Suffix trie for the sequence ATCGGGATCGTA

A	T	C	G	G	G	A	T	C	G	T	A	$
0	1	2	3	4	5	6	7	8	9	10	11	12

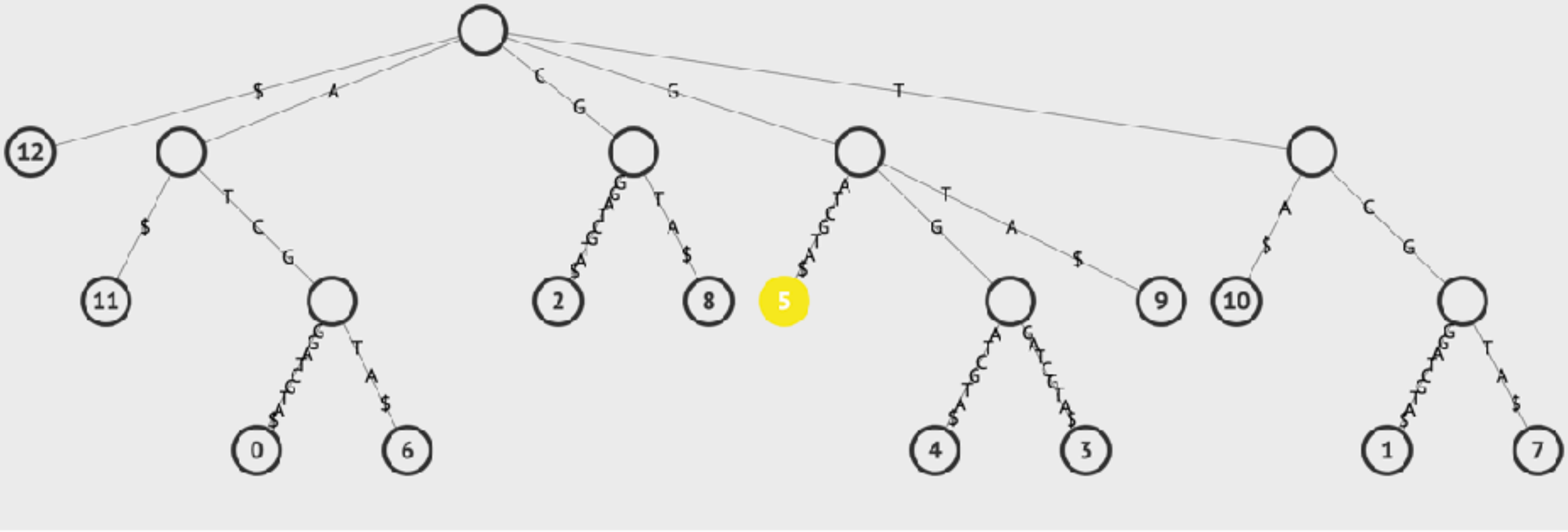


Tries built using https://visualgo.net/en/suffixtree

Suffix trie for the sequence ATCGGGATCGTA

Search for the Substring "GAT"

A	T	C	G	G	**G**	**A**	**T**	C	G	T	A	$
0	1	2	3	4	**5**	**6**	**7**	8	9	10	11	12



Tries built using https://visualgo.net/en/suffixtree

Suffix-Prefix Trie

A family of methods which uses a Trie structure to search a reference sequence (e.g. Bowtie, BWA, SOAP2)

Trie – data structure which stores the suffixes (i.e. ends of a sequence)

Key advantage over hashed algorithms:

- Alignment of multiple copies of an identical sequence in the reference only needs to be done once

- Bowtie uses something called an *FM-Index* to store Tries and drastically reduces memory requirements (e.g. Human genome can be stored in 2Gb of RAM)

- Burrows Wheeler Transform to perform fast lookups, replacing the hash

Burrows-Wheeler Algorithm

- Encodes data so that it is easier to compress
- Can be reversed to recover the original word

Transformation				
Input	All Rotations	Sorting All Rows in Alphabetical Order by their first letters	Taking Last Column	Output Last Column
^BANANA|	^BANANA| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	BNN^AA|A

Suffix-Prefix Trie

Less sensitive for sequences that are more different from the reference so problems can arise with:

- Sequencing errors
- Query - Reference differences (i.e. divergence)

Comparison

Table 1.
Benchmark of short read alignment tools

Software	Reads aligned (%)	Time (paired, s)	Time (single, s)	Memory usage (GB)
SOAP2	93.6	828	478	5.4
SOAP	93.8	19 234	14 328	14.7
MAQ	93.2	22 506	19 847	1.2
Bowtie	91.7	–	405	2.3

Suffix

Hash

Hash

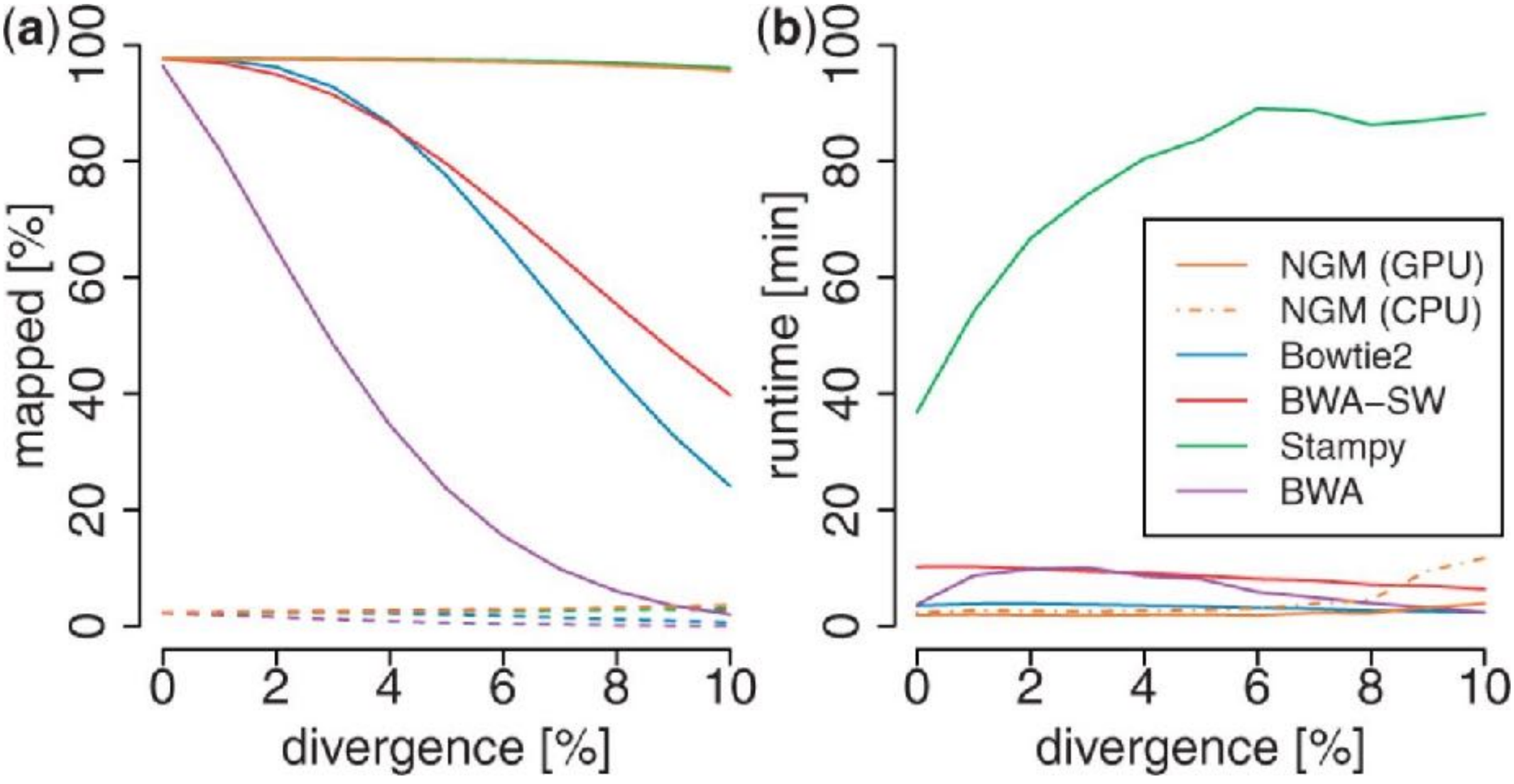
Suffix

Table from
Li et al Bioinformatics. 2009

Popular short read aligners

Program	Algorithm	Speed	Accuracy for divergent sequences
Bowtie2	Suffix/Prefix	Very fast	Low
BWA	Suffix/Prefix	Fast	Medium
Stampy	Hashing ref	Slow	High
Soap2	Suffix/Prefix	Fast	Low
Novoalign	Hashing ref	Slow	High
NextGenMap	Hashing ref	Med	High

Alignment stats



*From NextGenMap paper

Long read alignment

Long reads might contain structural variation that makes it hard to form a linear alignment

For example, a read containing a large inversion would contain 3 linear alignments

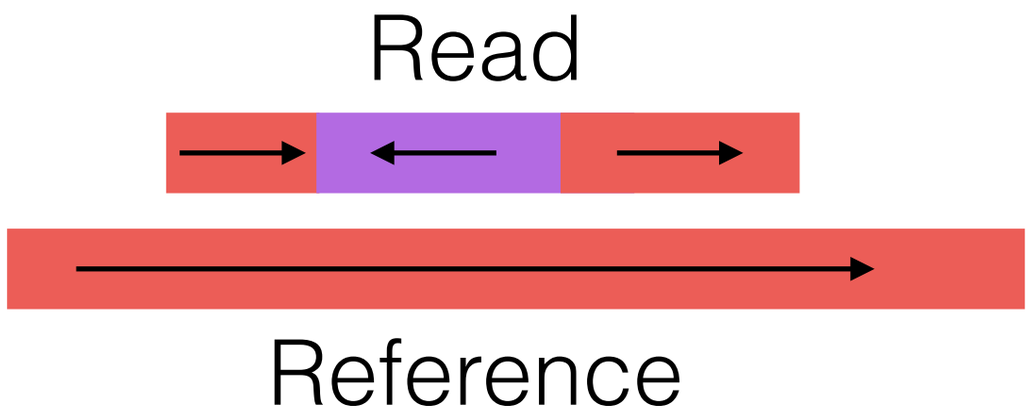
Add to that, that long read technologies have v. high error rates

Long read alignment

Long reads might contain structural variation that makes it hard to form a linear alignment

For example, a read containing a large inversion would contain 3 linear alignments

Add to that, that long read technologies have v. high error rates

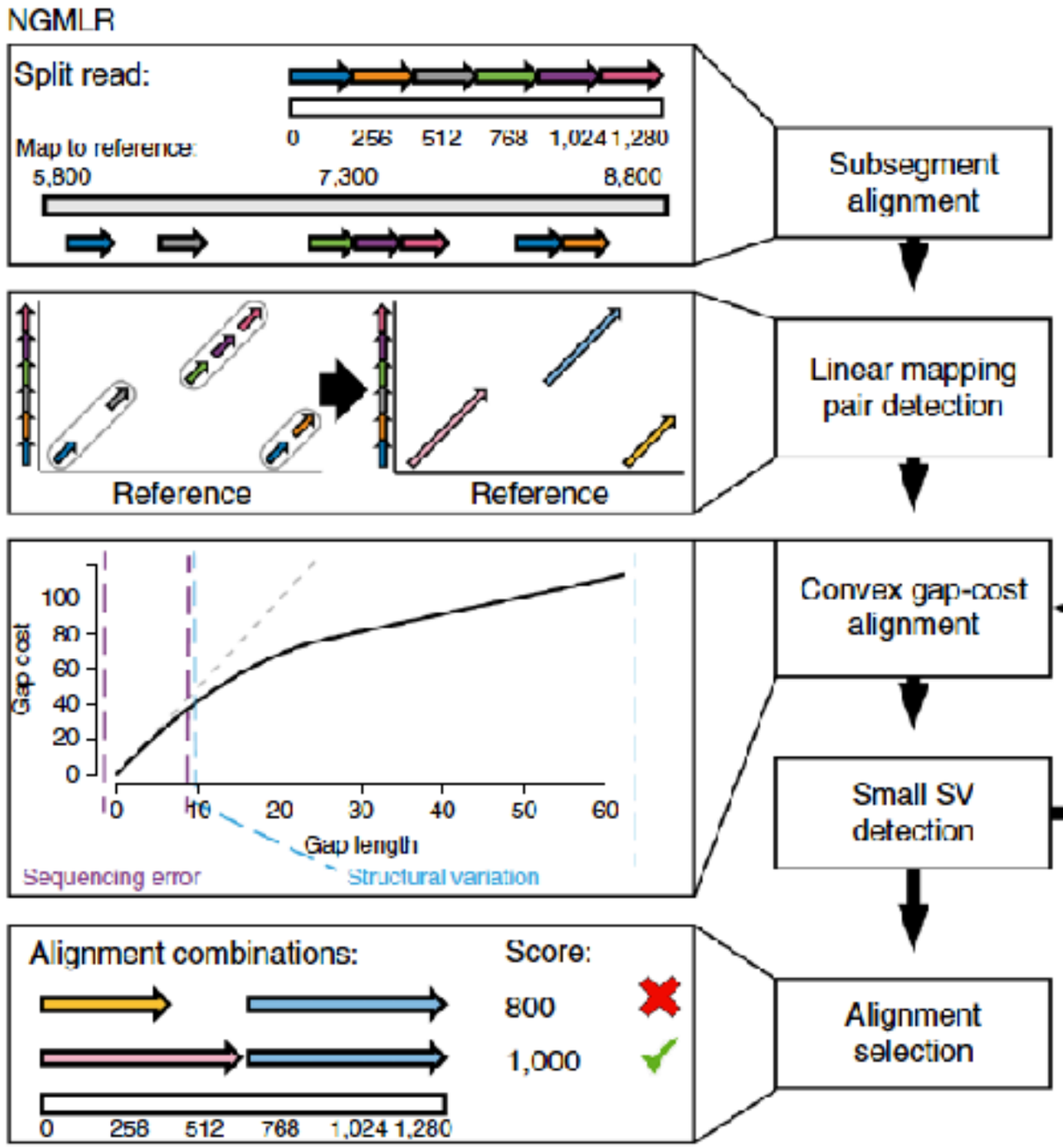


Long read alignment with NGMLR

Find exact matches between read fragment and reference

Look for chains of matches

Use local alignment of read to best reference region.



A similar approach is taken by minimap2

From NGMLR paper - Sedlazeck et al 2018 Nat. Methods

Long read alignment

Longer reads have more information, but potentially more error.

Example: The **NGMLR** software uses k-mers to map seeds then uses the Smith-Waterman algorithm for exact placements

Example: The **minimap2** software uses a hashed-seed extend approach, but a subset of all possible seeds are used. Seed matches are then chained together

Other programs:

- KART, BWA-MEM, BLASR

Alignment choice

- Speed needed?
- How divergent is sequence from reference? Same species or relative?
- How much variation in your samples?
- Genome size of reference?

SAM (BAM) format

Sequence Alignment/Map format

- Universal standard.
- Generally aligned to reference, but not necessarily
- Human-readable (SAM) and compressed (BAM) forms

Structure:

- Header: Version, sort order, reference sequences, read groups, program/processing history
- Alignment records

Keeps a track of the reference genome

And individual read records

SAM format: Header

```
@HD     VN:1.5  GO:none SO:coordinate
@SQ     SN:cp_gi_88656873       LN:151104
@SQ     SN:mt_gi_571031384      LN:300945
@SQ     SN:rDNA_gi_563582565    LN:9814
@SQ     SN:Ha1  LN:175985764
@SQ     SN:Ha2  LN:209013747
@SQ     SN:Ha3  LN:203472901
@SQ     SN:Ha4  LN:216026857
@SQ     SN:Ha5  LN:271056985
@SQ     SN:Ha6  LN:100519666
@SQ     SN:Ha7  LN:109221022
@SQ     SN:Ha8  LN:192129815
@SQ     SN:Ha9  LN:253478808
@SQ     SN:Ha10 LN:327788049
@SQ     SN:Ha11 LN:208730832
@SQ     SN:Ha12 LN:208068730
@SQ     SN:Ha13 LN:239367298
@SQ     SN:Ha14 LN:230295834
@SQ     SN:Ha15 LN:202246870
@SQ     SN:Ha16 LN:226777971
@SQ     SN:Ha17 LN:267415242
@SQ     SN:Ha0_73Ns     LN:359367108
@RG     ID:HI.2034.006.Index_18.W70_NHK_2013_5  LB:Anomalus     PL:ILLUMINA     SM:HI.2034.006.Index_18.W70_NHK_2013_5  PU:Anomalus
@PG     ID:ngm  PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --dualstrand 1
@PG     ID:ngm.1        PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --
@PG     ID:ngm.2        PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --
```

SAM format: Header

```
@HD     VN:1.5  GO:none SO:coordinate
@SQ     SN:cp_gi_88656873       LN:151104
@SQ     SN:mt_gi_571031384      LN:300945
@SQ     SN:rDNA_gi_563582565    LN:9814
@SQ     SN:Ha1  LN:175985764
@SQ     SN:Ha2  LN:209013747
@SQ     SN:Ha3  LN:203472901
@SQ     SN:Ha4  LN:216026857
@SQ     SN:Ha5  LN:271056985
@SQ     SN:Ha6  LN:100519666
@SQ     SN:Ha7  LN:109221022
@SQ     SN:Ha8  LN:192129815
@SQ     SN:Ha9  LN:253478808
@SQ     SN:Ha10 LN:327788049
@SQ     SN:Ha11 LN:208730832
@SQ     SN:Ha12 LN:208068730
@SQ     SN:Ha13 LN:239367298
@SQ     SN:Ha14 LN:230295834
@SQ     SN:Ha15 LN:202246870
@SQ     SN:Ha16 LN:226777971
@SQ     SN:Ha17 LN:267415242
@SQ     SN:Ha0_73Ns     LN:359367108
@RG     ID:HI.2034.006.Index_18.W70_NHK_2013_5  LB:Anomalus     PL:ILLUMINA     SM:HI.2034.006.Index_18.W70_NHK_2013_5  PU:Anomalus
@PG     ID:ngm  PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --dualstrand 1
@PG     ID:ngm.1        PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --
@PG     ID:ngm.2        PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --
```

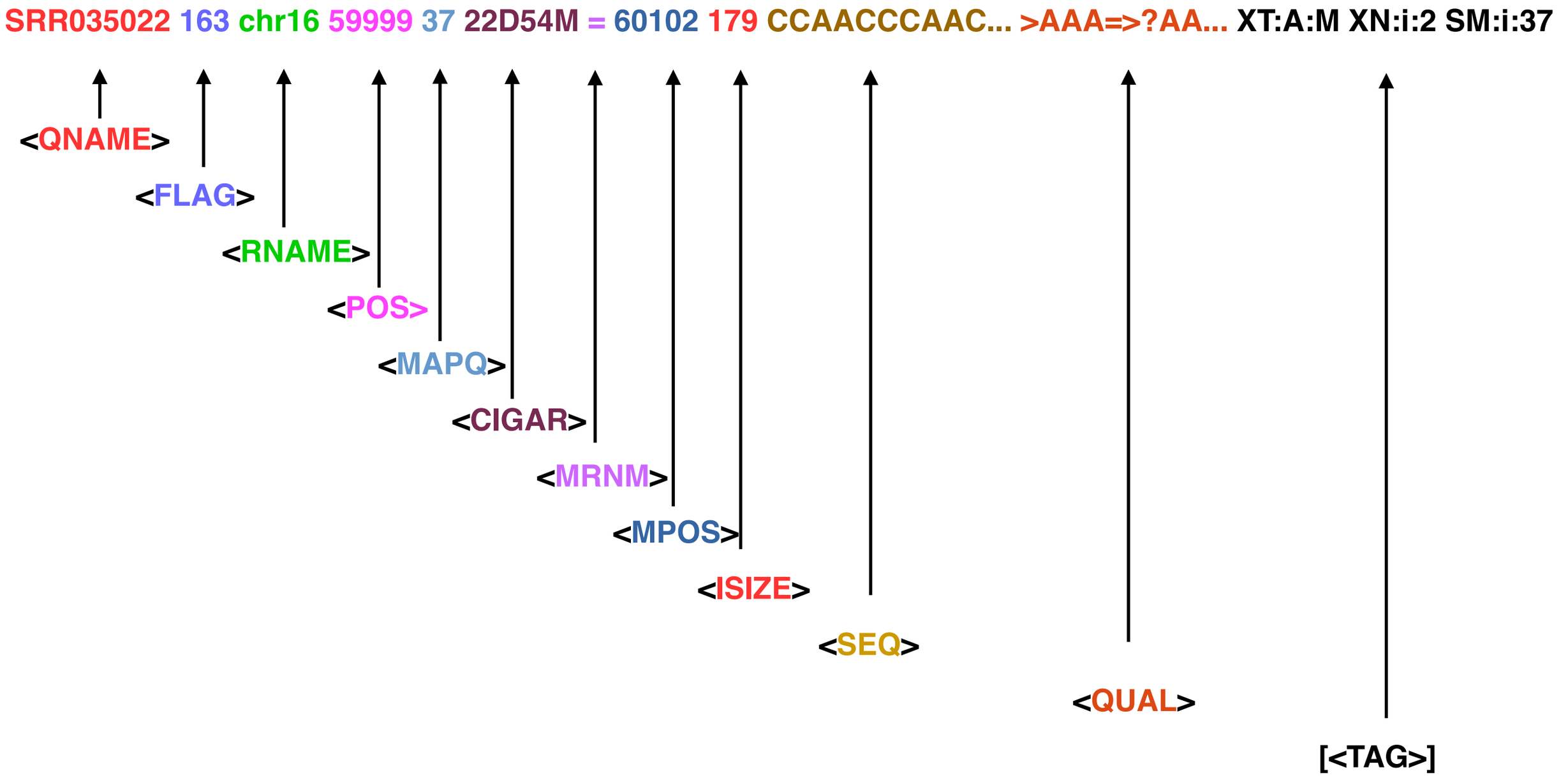
Sort order - in this case coordinate based

Reference sequence name (SN) and length (LN)
e.g. Chromosome Ha7, which is 109,221,022bp long

Read group (@RG) information

Program (@PG) information - what you used to map the reads

SAM format: Read lines



SAM format: Read lines

SRR035022 163 chr16 59999 37 22D54M = 60102 179 CCAACCCAAC... >AAA=>?AA... XT:A:M XN:i:2 SM:i:37

<QNAME> Query name - i.e. the name of the read

<FLAG> A combination of bitwise flags that indicate properties of the alignment (complement, strand etc.)

<RNAME> Reference sequence name

<POS> Position in the reference (1-based)

<MAPQ> Mapping quality

<CIGAR> Concise Idiosyncratic Gapped Alignment Report (CIGAR) string - tells you about gaps in the alignment

<MRNM> Read-mate reference sequence

<MPOS> Read-mate position in the reference

<ISIZE> Insert size

<SEQ> The segment's sequence

<QUAL> An ASCII sequence containing quality information for each base in the sequence

[<TAG>] These are optional tags that get added and contain user specified data (For example, SM is the mapping quality of this sequence only - ignoring the read mate)

SAM format: Read lines

SRR035022 163 chr16 59999 37 22D54M = 60102 179 CCAACCCAAC... >AAA=>?AA... XT:A:M XN:i:2 SM:i:37

<QNAME> Query name - i.e. the name of the read

<FLAG> A combination of bitwise flags that indicate properties of the alignment (complement, strand etc.)

<RNAME> Reference sequence name

<POS> Position in the reference (1-based)

<M

<C ent

<M

<M

<IS

<SEQ> The segment's sequence

<QUAL> An ASCII sequence containing quality information for each base in the sequence

[<TAG>] These are optional tags that get added and contain user specified data (For example, SM is the mapping quality of this sequence only - ignoring the read mate)

For an explanation of bitwise flags:

https://broadinstitute.github.io/picard/explain-flags.html

Mapping Quality

- MapQ = Qs = -10 $\log_{10}(P)$
- P = probability that this mapping is NOT the correct one
- MapQ = 0 = equally likely to map somewhere else
- Different programs use different formulas for P
- A value of 255 indicates that the information is not available

Tutorial:

Work through the tutorial associated with this session

Further Reading

A concise read:

Trapnell, C., & Salzberg, S. L. (2009). How to map billions of short reads onto genomes. *Nature biotechnology*, *27*(5), 455-457.

A more in depth read:

Reinert, K., Langmead, B., Weese, D., & Evers, D. J. (2015). Alignment of next-generation sequencing reads. *Annual review of genomics and human genetics*, *16*, 133-151.

All available on the GitHub

Seed-extend algorithm