

TOPIC 4: Sequence alignment

Biol 525D - Bioinformatics for Evolutionary Biology
2020

Learning Goals

- Be able to define the two main methods of alignment
- Understand the two main algorithms for NGS alignment, including strengths and weaknesses
- Be able to read SAM format

Outline

- Sequence alignment
- Alignment algorithms
- Dealing with genome-scale alignment
- The SAM/BAM format

Sequence alignment

Sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.

```
Q5E940_DOWIN -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_HUMAN -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_MOUSE -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_RAT -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_CHICK -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_RANBY -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
Q7ZUC3_BARRR -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_ICTPB -----MREDRATNKKNYFLKIIIDLDYKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_DROME -----MYRENKRAKKAQYFIKVVLEFDEFFKCFIVGADNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_DICD1 -----MSGLA-SERKKLPLEKAIELFTYDKMIVALEAUFVGSFOLKIKKSIHLI-GAVLNGKNTMIREVIRDLADSE--VELD 75
Q54LFO_DICD1 -----MSGLA-SERKKLPLEKAIELFTYDKMIVALEAUFVGSFOLKIKKSIHLI-GAVLNGKNTMIREVIRDLADSE--VELD 75
RLA0_ELAFB -----MAKLSKQKKQMYIEKLSLQQYSKILIVHVDNVGSKOMQIRMSLQK-AVVLNGKNTMMRKATRGHLENN--PALE 76
RLA0_SULAC -----MGLAVITTKKIAKXVDEVAELEKLTKEFTIIANIEGFPADKLHEIRKELQK-ADIKYKNNLFNIALKNAQ--DQK 79
RLA0_SULTO -----MRHAVITQERRIAKXIEEVESELEKIREVHTIIANIEGFPADKLHEIRKELQK-AEIKYKNTLFLGIALKNAQ--DQK 80
RLA0_SULSO -----MKHIALALGQKQKVASXKLEFVKELEITKNSHTILGNIIEGFPADKLHEIRKELQK-ADIKYKNNLFNIALKNAQ--DQK 80
RLA0_AKRRK MSVSVIVGQMYKRRKPIFENKFLMERLEKLEPKRRVVLPADELFGTPFVVRVVEKRIAKK-SPMMVAKKRIILHAKKRALE--LDGN 86
RLA0_PYRAE -HMLAIGKRRYVATQYFASRYKLYSENIALLQKYPYFLDNGLESSEILHEVRYLHNY-GYIKIIRKPLLEKIAFTKYVGG--LQAE 85
RLA0_METAC -----MAEERHETEHIDQKKDEIENIKELIQSHKVFQMVGIEGILATKMKIRDLQDY-AVLYKSHNTLLEBALNQLG--ETID 78
RLA0_METMA -----MAEERHETEHIDQKKDEIENIKELIQSHKVFQMVGIEGILATKMKIRDLQDY-AVLYKSHNTLLEBALNQLG--ESIP 78
RLA0_AKCFB -----MAAVRGS-----PPEYVRAVEEIKRMISSEKPVVAIVSPRNVPAQMKIRREFPK-REIKYVNTLLEBALDQLG--DQYL 75
RLA0_METRA MAVKAKGQDDSCYEKKVAENKRRREVLEKLEIMDEVENVGLVLEGIDAPDLCEINAKLEADTEIHSNTLMDRIALEENKDER--PELE 88
RLA0_MKTH -----MAHVAKKKKKEVKELEKLEIKSYFVVLADIPARLQKMKQILHUN-ALIKMKKPLINLALKKAKKEL--ENVD 76
RLA0_METTI -----MTTAREKHTAPKETEFPVKLEKTIKNGQIVAVDHNVEVPADLQETIDKTI-ETMTLKNSNTLITDAYETVAETQHPFA 82
RLA0_METVA -----MIDAKSEKHTAPKETEFPVKLEKTIKNGQIVAVDHNVEVPADLQETIDKTI-DQMTLKNSNTLITDAYETVAETQHPFA 82
RLA0_METJA -----HETKRAHVAKKETEFPVKLEKTIKNGQIVAVDHNVEVPADLQETIDKTI-DQMTLKNSNTLITDAYETVAETQHPFA 81
RLA0_PYRAE -----MAHVAKKKKKEVKELEKLEIKSYFVVLADIPARLQKMKQILHUN-ALIKMKKPLINLALKKAKKEL--ENVD 77
RLA0_PYREG -----MAHVAKKKKKEVKELEKLEIKSYFVVLADIPARLQKMKQILHUN-ALIKMKKPLINLALKKAKKEL--ENVD 77
RLA0_PYKFB -----MAHVAKKKKKEVKELEKLEIKSYFVVLADIPARLQKMKQILHUN-ALIKMKKPLINLALKKAKKEL--ENVD 77
RLA0_PYRKO -----MAHVAKKKKKEVKELEKLEIKSYFVVLADIPARLQKMKQILHUN-ALIKMKKPLINLALKKAKKEL--ENVD 76
RLA0_HALMA -----MSAESERKTETIFENKQEVDAIVEKIESYESVGVVNIAGIPSRQLDMDRDLHST-AELRYSENPLLEBALDQVD--DQLE 79
RLA0_HALVO -----MSAESEVQTEVITQKREFEVDLVDFTESYESVGVVNIAGIPSRQLDMDRDLHST-AELRYSENPLLEBALDQVD--DQFE 79
RLA0_HALSA -----MSAESEVQTEVITQKREFEVDLVDFTESYESVGVVNIAGIPSRQLDMDRDLHST-AELRYSENPLLEBALDQVD--DQTD 79
RLA0_PRRAC -----MRKVSQKKKLLVKKIPHKKARRNVAVDQAGIRHRIQDIEGKHHK-INLKVIRKPLLEKALKKLEK--EKLN 72
RLA0_PRRYO -----MRKVSQKKKLLVKKIPHKKARRNVAVDQAGIRHRIQDIEGKHHK-INLKVIRKPLLEKALKKLEK--EKLN 72
RLA0_PICTO -----MTEDPQNKIDFVKNLSEINERKVAIVSINKLRNNEFPKIRNSIRDK-ARIKYRABLLRLAENTCK--NNIV 72
ruler 1.....10.....20.....30.....40.....50.....60.....70.....80.....90
```

A multiple alignment of protein sequences

From Wikipedia

Pairwise alignment

Alignment of two sequences is a relatively straightforward computational problem, but...

- there are many possible alignments
- there can be a very large reference

NOTE: Two sequences can always be aligned and there can be more than one optimal solution

Methods of alignment

By hand

- Can be accurate, but a bit fishy

Mathematical approach

- Dynamic programming (slow, but optimal)

Heuristic methods (fast, but approximate)

- BLAST, short read aligners

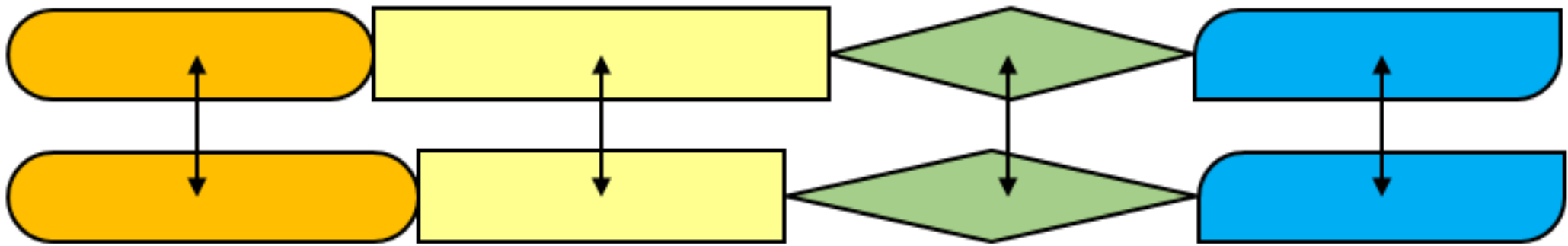


Dynamic programming

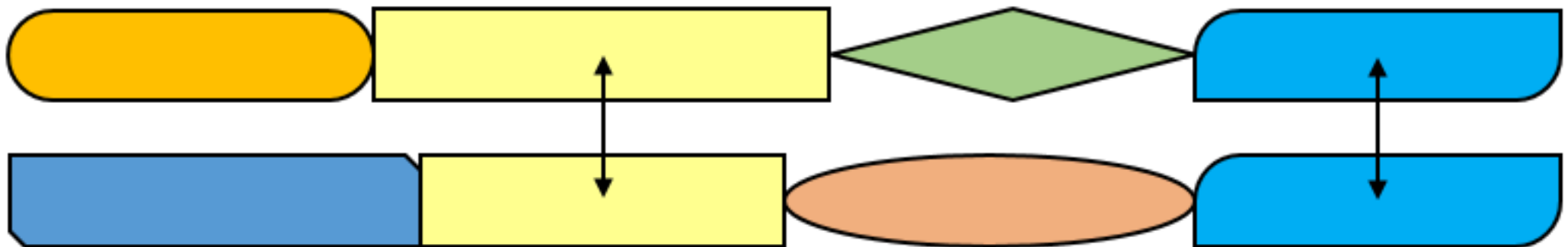
Dynamic programming is a general programming technique

It structures a large search space into a succession of stages

- The initial stage contains trivial solutions to sub-problems
- Each partial solution in a later stage can be calculated by recurring a fixed number of partial solutions in an earlier stage
- The final stage contains the overall solution



Global Alignment



Local Alignment

Global vs Local alignments

Global alignment algorithms start at the beginning of two sequences and add gaps to each until the end of one is reached (Needleman-Wunsch algorithm).

Local alignment algorithms finds the region (or regions) of highest similarity between two sequences (e.g. the Smith-Waterman algorithm).

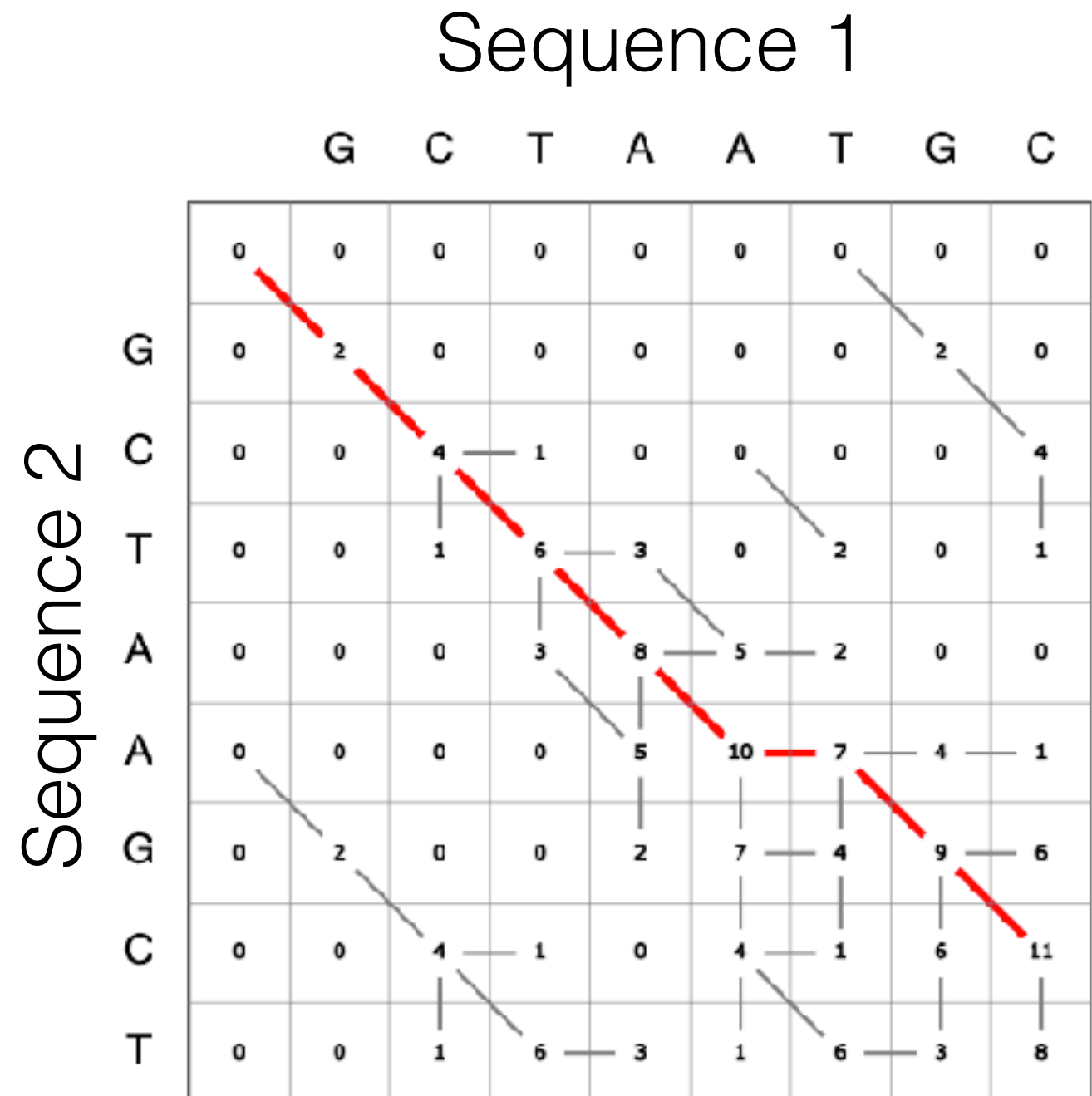
Here's a fun demo of the two algorithms:

<https://gtuckerkellogg.github.io/pairwise/demo/>

Basic principles of dynamic programming

There are too many comparisons to try them all so instead:

- Build alignment path matrix
- Stepwise calculation of score values
- Backtracking (evaluation of optimal path)



Scoring methods

Scoring systems:

- Each symbol pairing is assigned a numerical value, based on a symbol comparison table.
 - nucleotides
 - amino acids (PAM, BLOSUM)

Gap penalties:

- Opening: The cost of introducing a gap.
- Extension: The cost to elongate a gap.

Gap penalties

- Too little gap penalty gives nonsense non-homologous alignments.
- Gaps are common, so too high gap penalty removes real alignments.
- There are multiple gap penalty functions (e.g. constant, linear and “affine”)
- The “affine” is the most commonly used gap penalty function (e.g. BLAST and BWA use it)

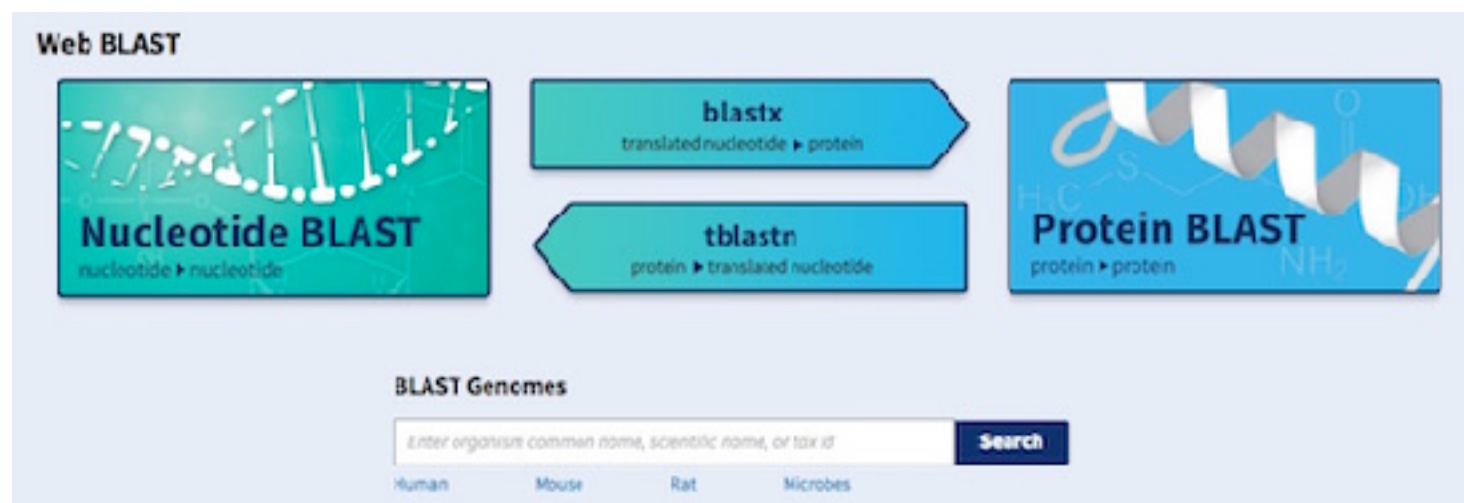
The “affine” gap penalty function

$$GP = A + BL$$

Where A is the penalty for opening a gap, B is the penalty for extending a gap and L is the length of the gap

BLAST - Best Local Alignment Search Tool

A great tool for comparing a small number of sequences against a database (e.g. NCBI BLAST)

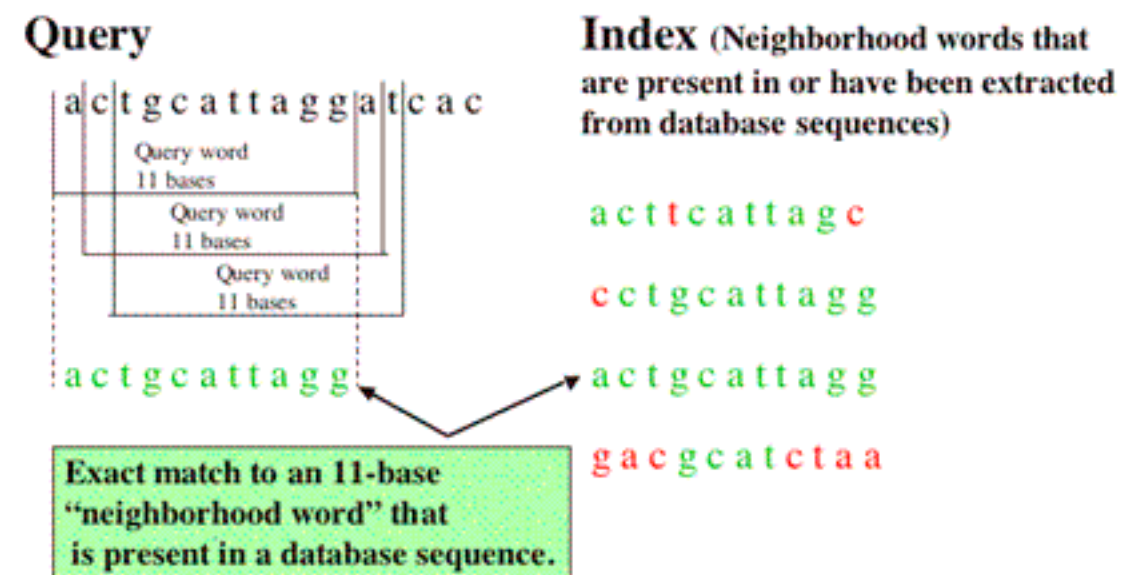


- An example of a “hashed seed-extend algorithm”

BLAST - Best Local Alignment Search Tool

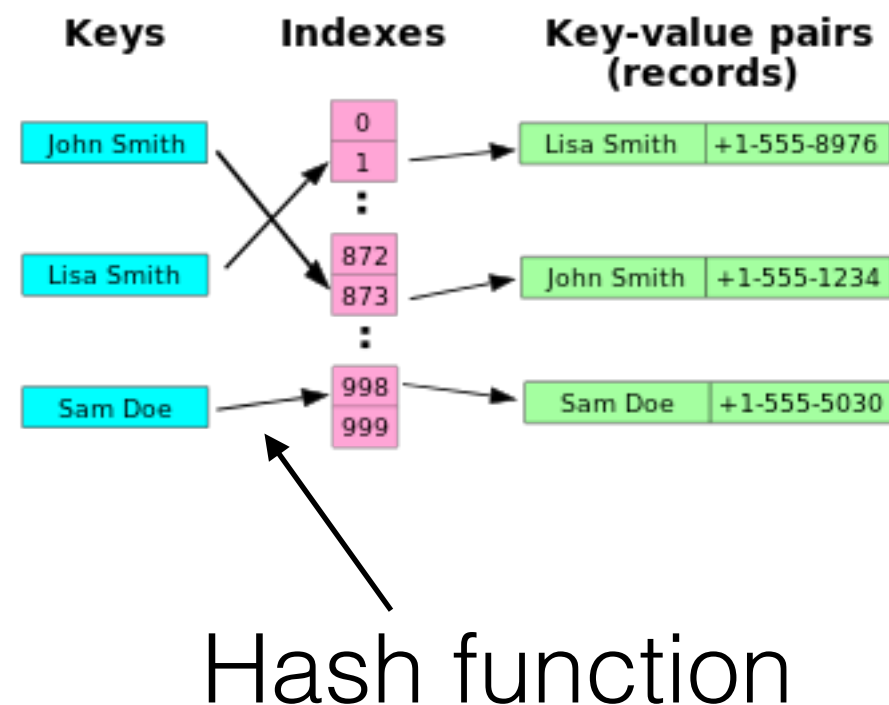
Designed to identify homologous sequences.

First finds highly conserved or identical sequences which are then extended with a local alignment



Hashed seed-extend algorithm

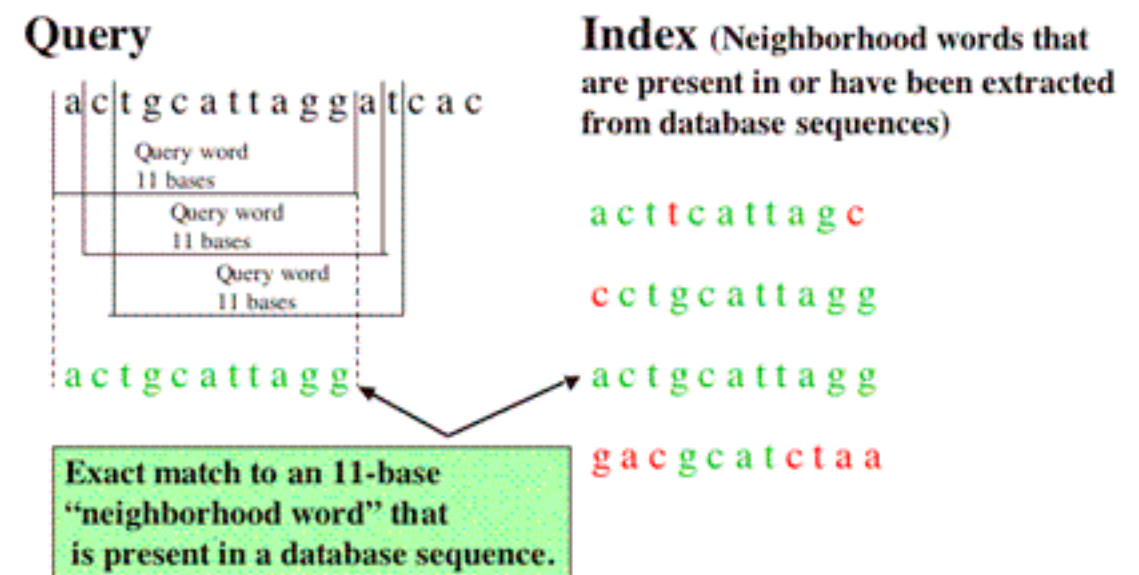
- A “hash” is a structure used in computer programming
- It is a way of storing information in a look-up table
- Allows efficient searching



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



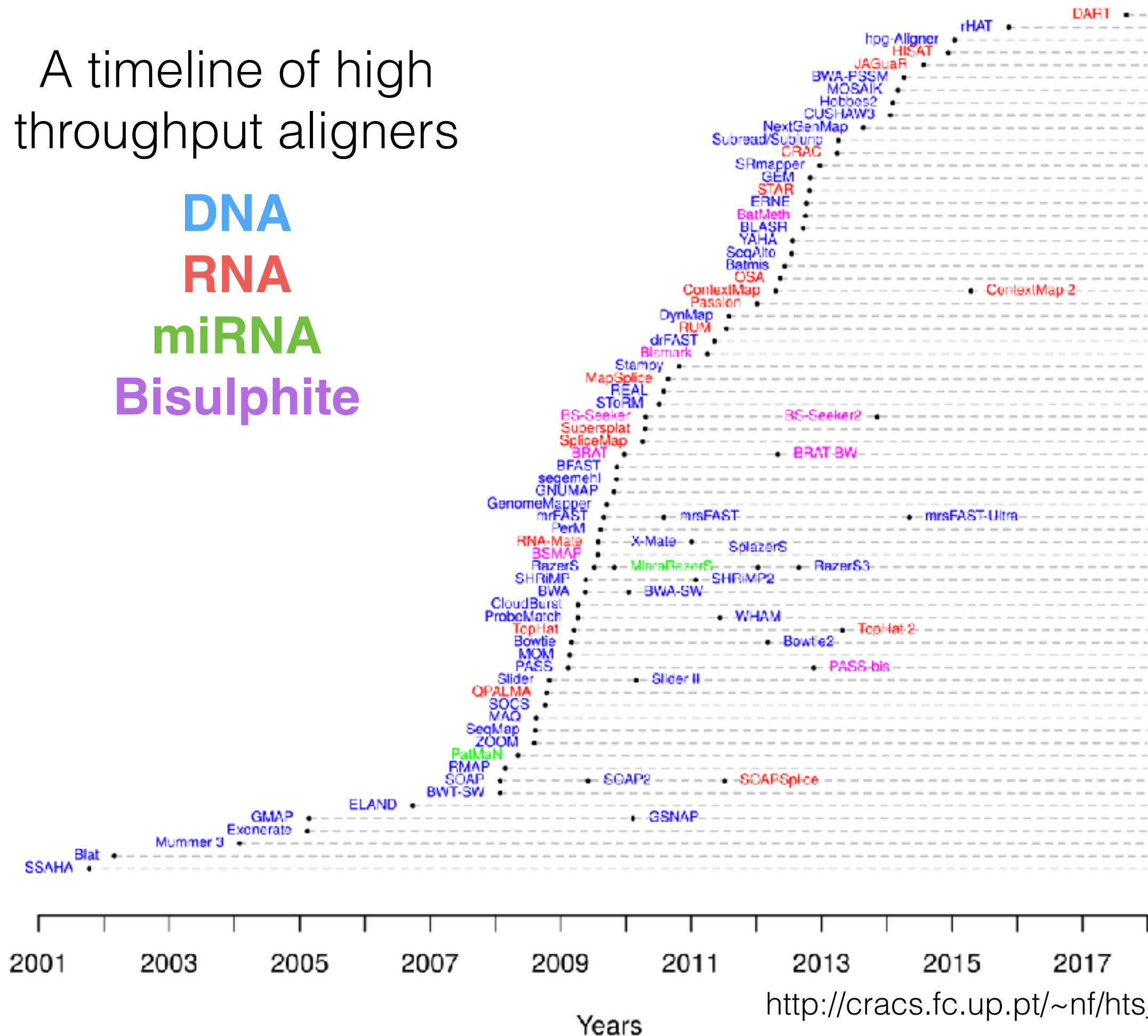
A timeline of high throughput aligners

DNA

RNA

miRNA

Bisulphite



Approaches to align short reads

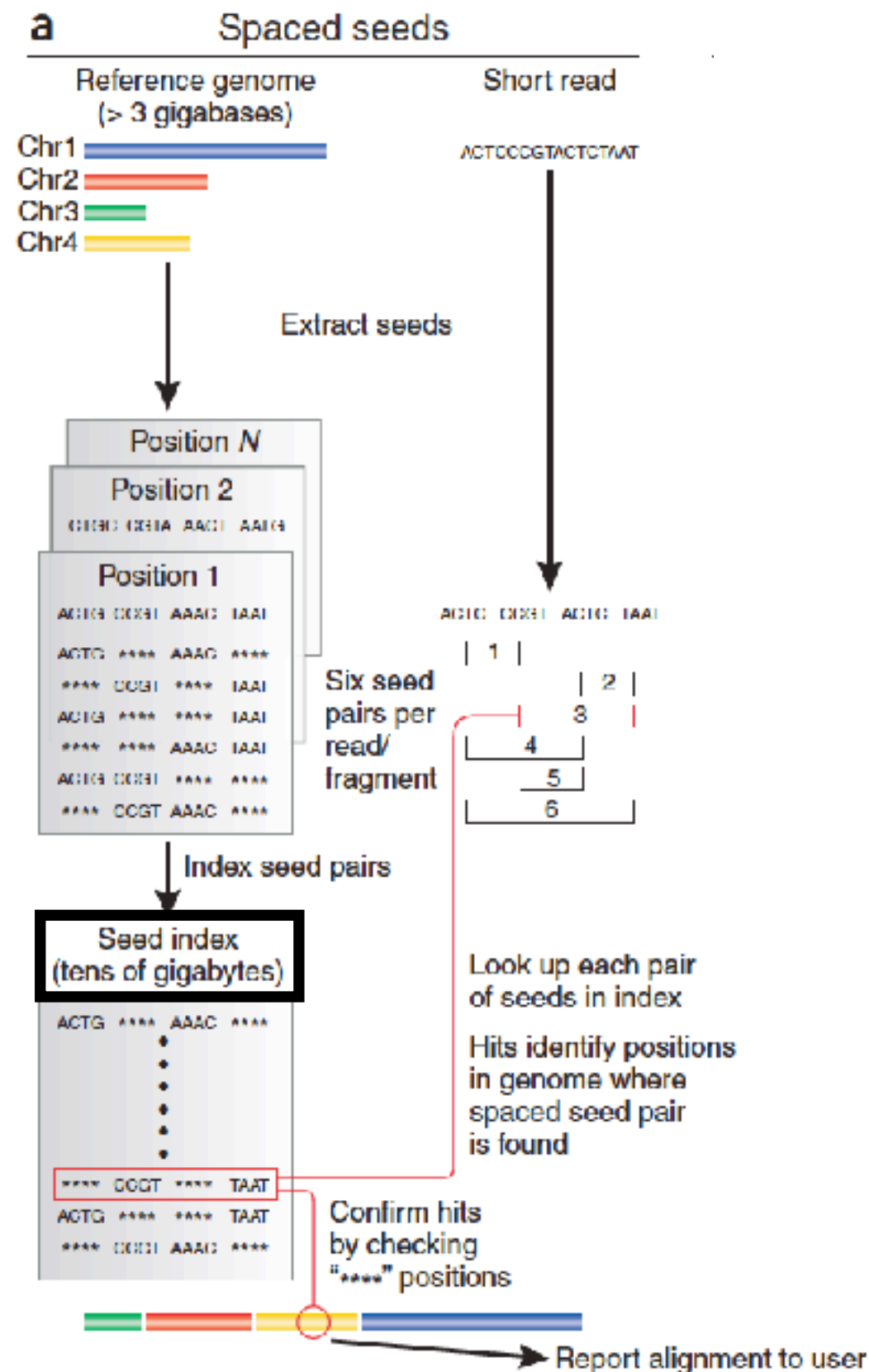


Figure modified from Trapnell & Salzberg 2009

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

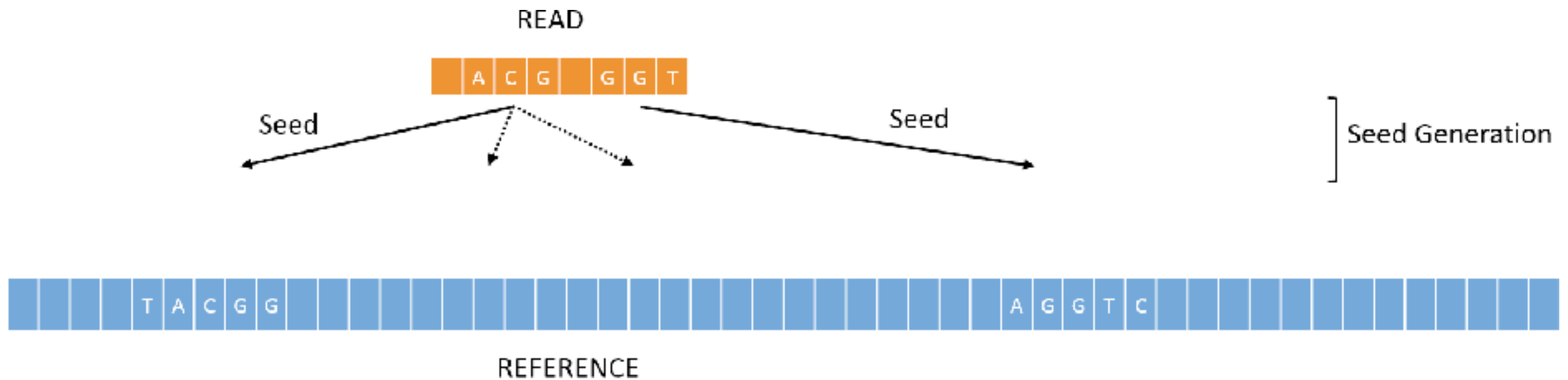
READ

A C G G G T

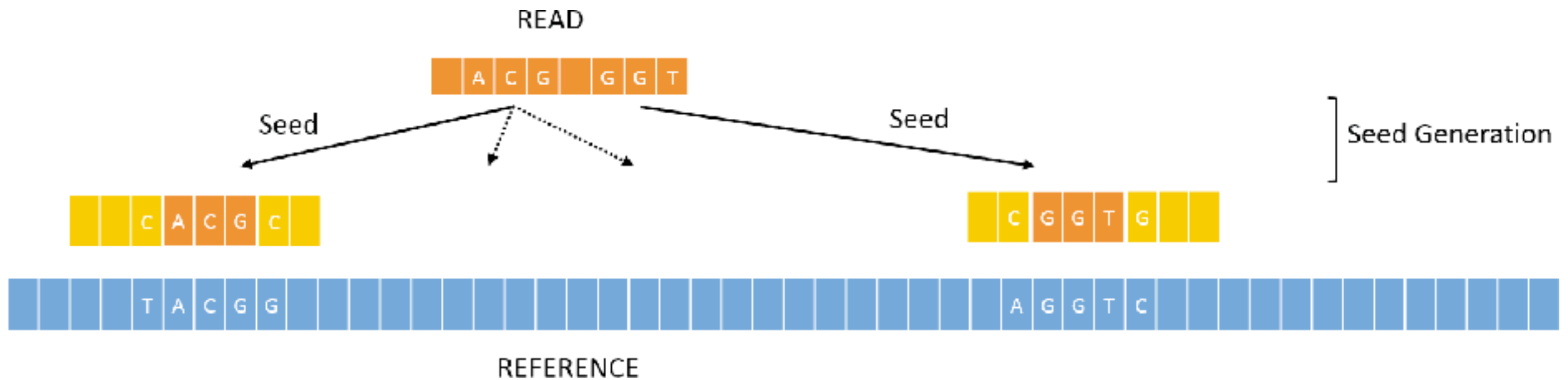
T A C G G A G G T C

REFERENCE

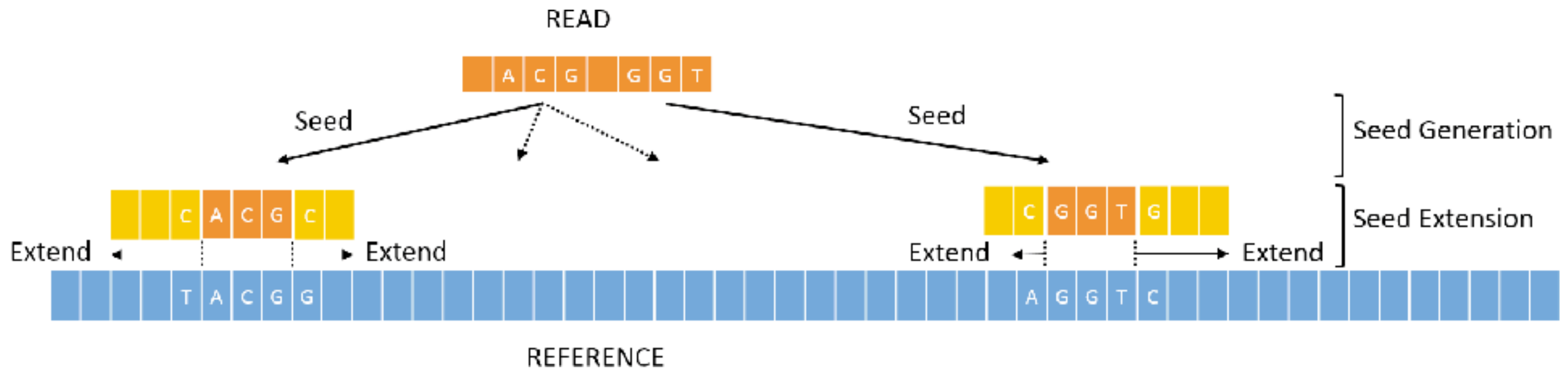
Seed-extend algorithm



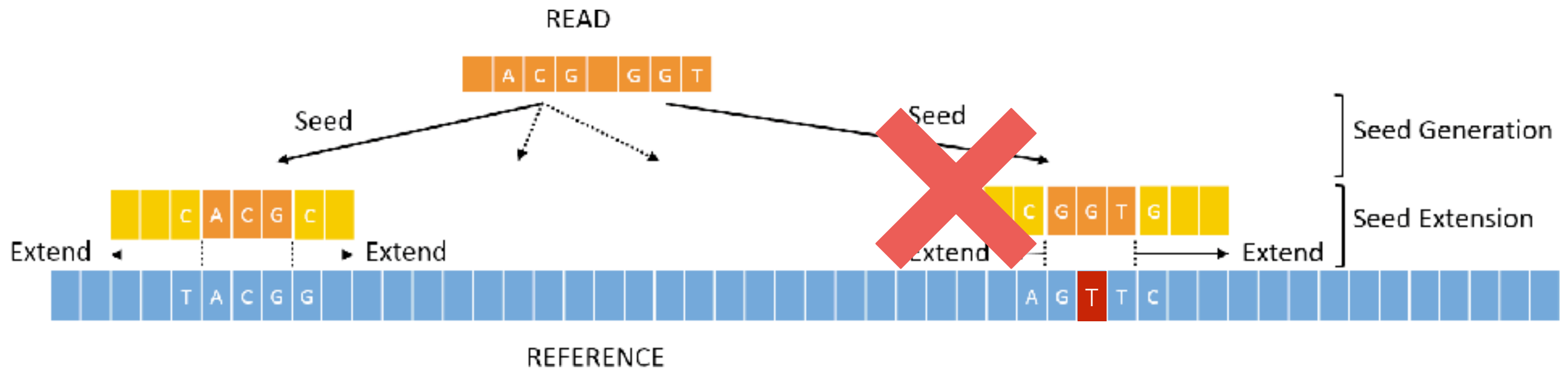
Seed-extend algorithm



Seed-extend algorithm



Seed-extend algorithm



Spaced seeds

To increase sensitivity we can use spaced-seeds:

11111111

Consecutive seed template with **length** 9bp

GATAGCTAGCTAAT

Reference

AGCTAGCTA

Query

10101101011011

Consecutive seed template with **weight** 9bp

GATAGCTAGCTAAT

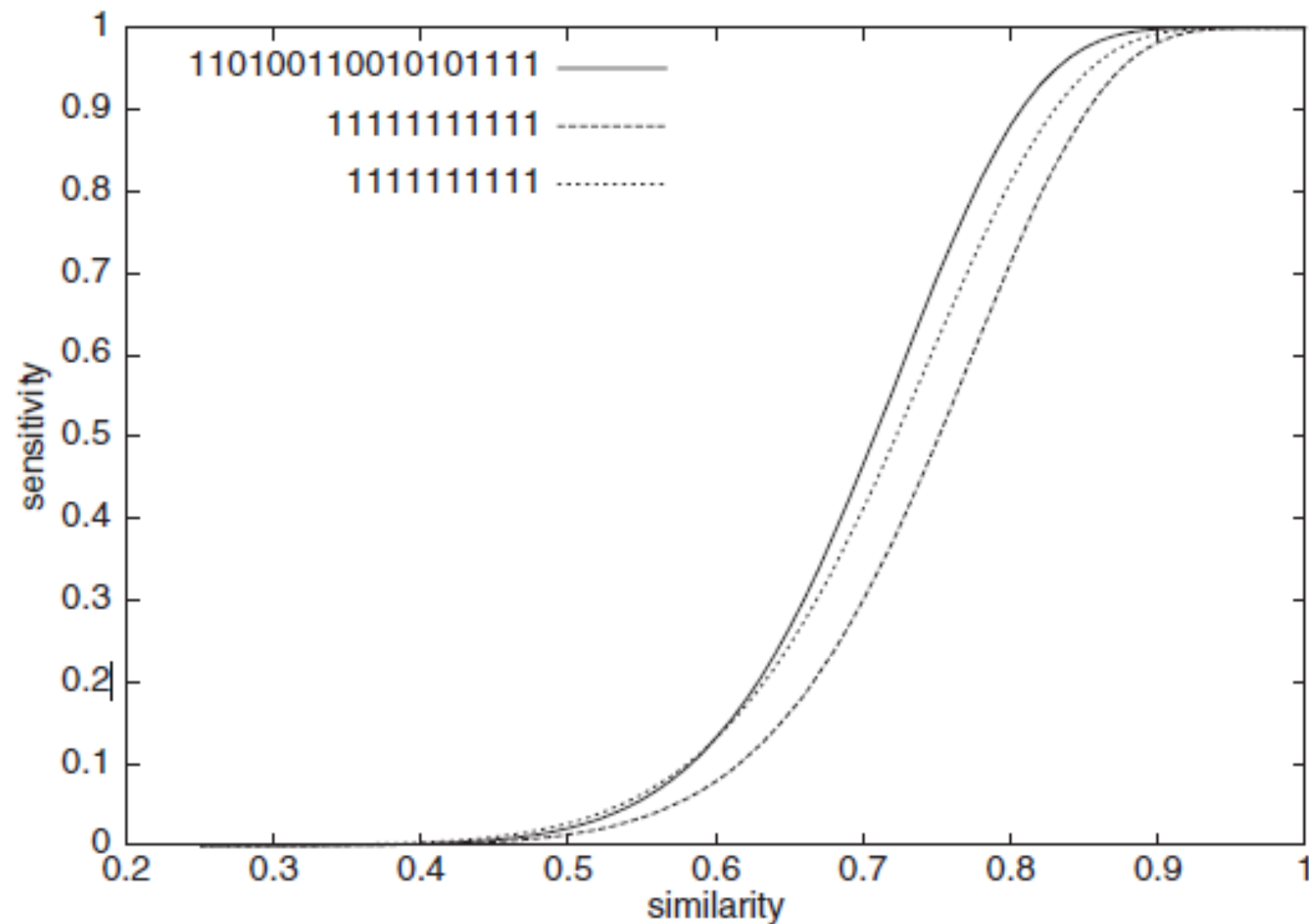
Reference

GATAGC**G**AGCTAAT

Query

Spaced seeds

- To increase sensitivity we can use spaced-seeds:



Approaches to align short reads

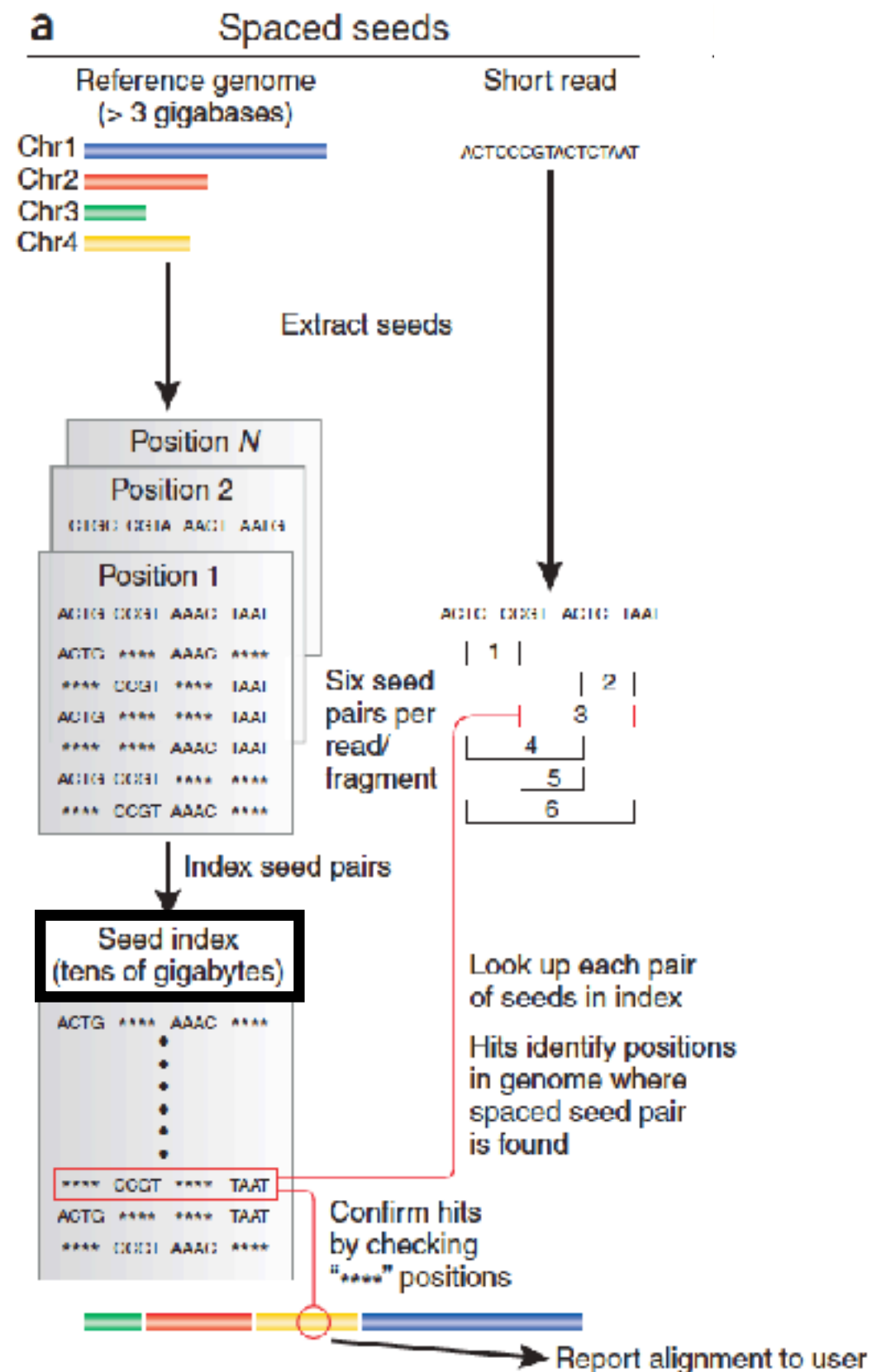


Figure modified from Trapnell & Salzberg 2009

Approaches to align short reads

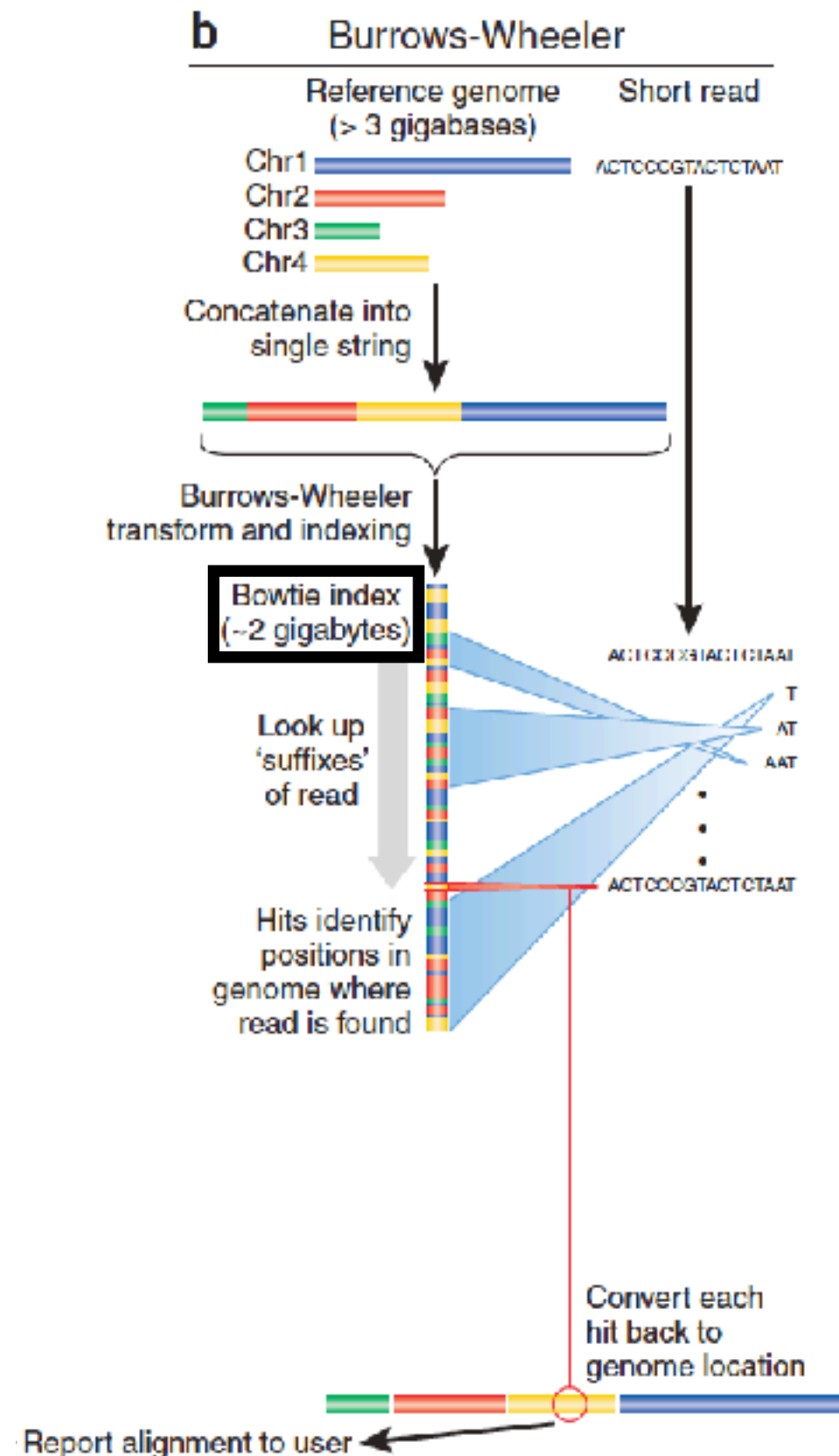
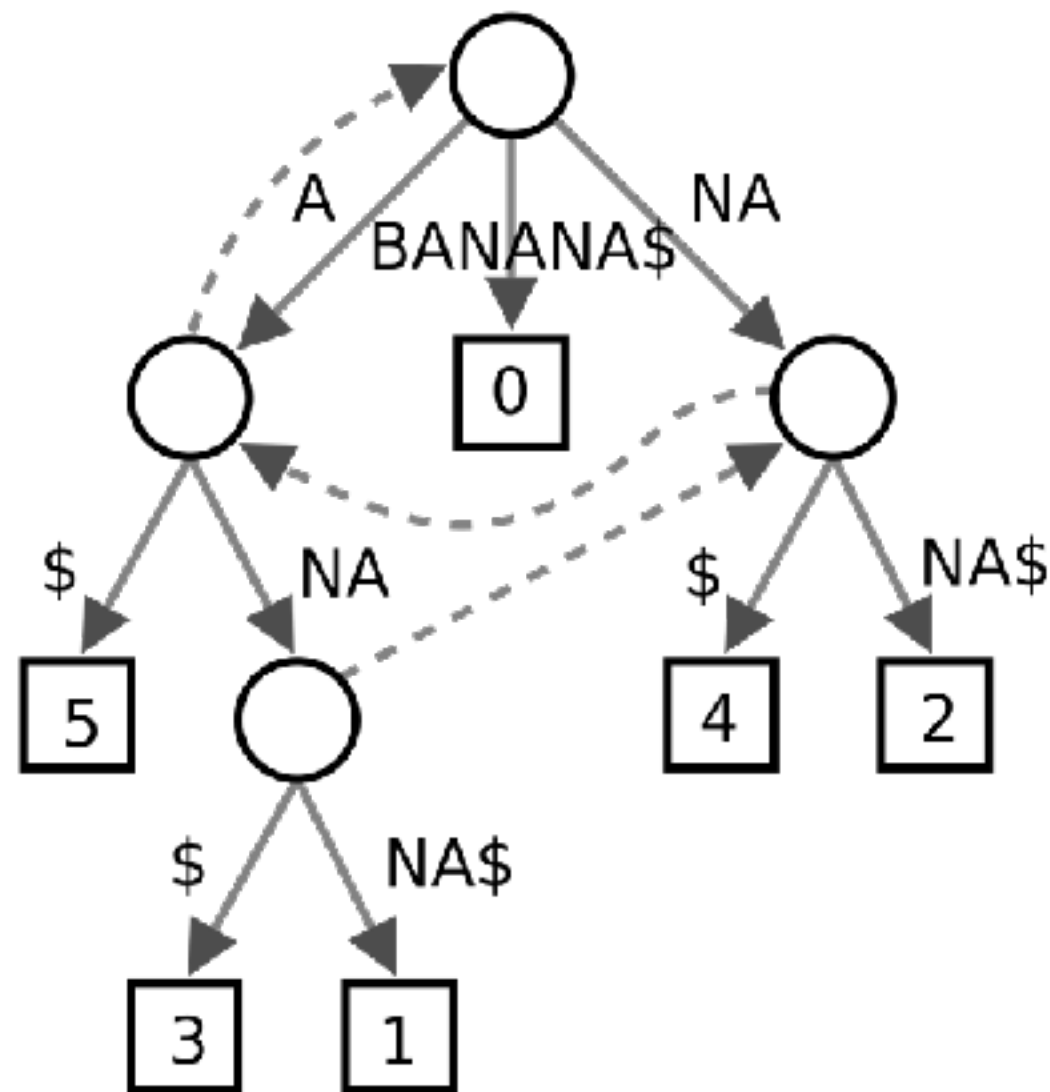


Figure modified from Trapnell & Salzberg 2009

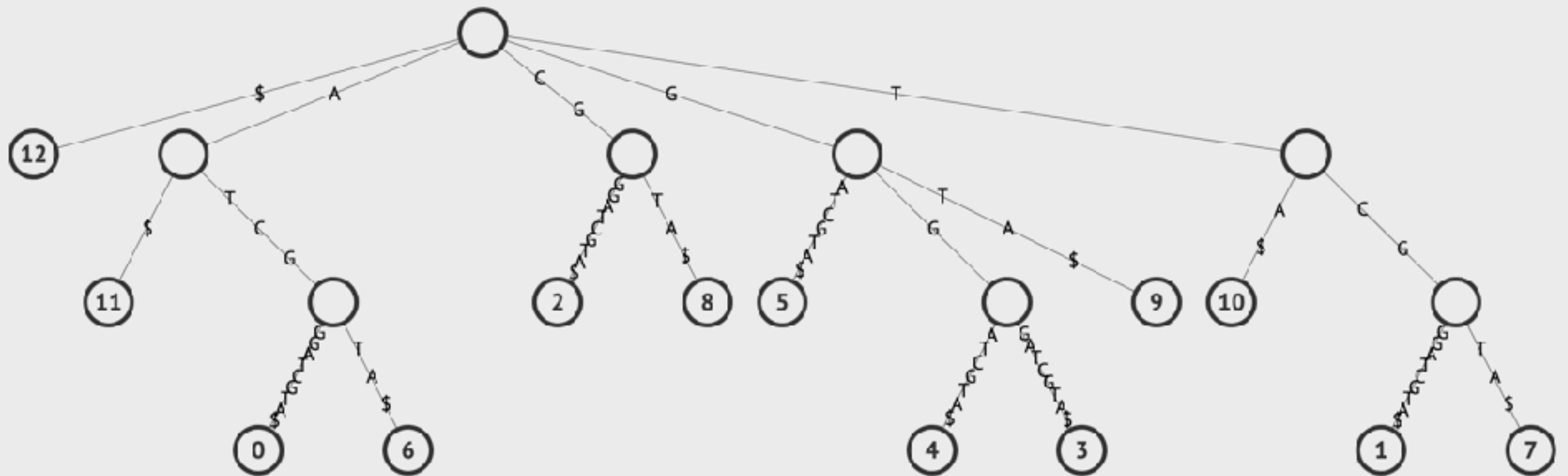
Suffix-Trie

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

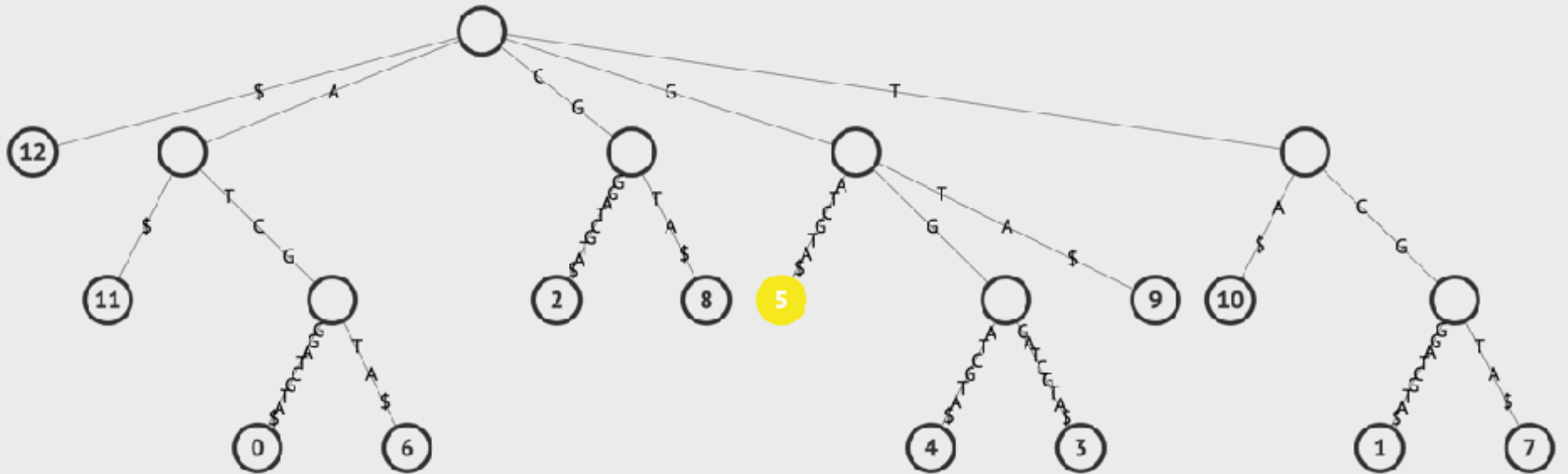


Tries built using <https://visualgo.net/en/suffixtree>

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
<div><div><div>^BANANA </div></div></div>	<div><div><div>^BANANA </div><div> ^BANANA</div><div>A ^BANAN</div><div>NA ^BANA</div><div>ANA ^BAN</div><div>NANA ^BA</div><div>ANANA ^B</div><div>BANANA ^</div></div></div>	<div><div><div>ANANA ^B</div><div>ANA ^BAN</div><div>A ^BANAN</div><div>BANANA ^</div><div>NANA ^BA</div><div>NA ^BANA</div><div>^BANANA </div><div> ^BANANA</div></div></div>	<div><div><div>ANANA ^B</div><div>ANA ^BAN</div><div>A ^BANAN</div><div>BANANA ^</div><div>NANA ^BA</div><div>NA ^BANA</div><div>^BANANA </div><div> ^BANANA</div></div></div>	<div><div><div>BNN^AA A</div></div></div>

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

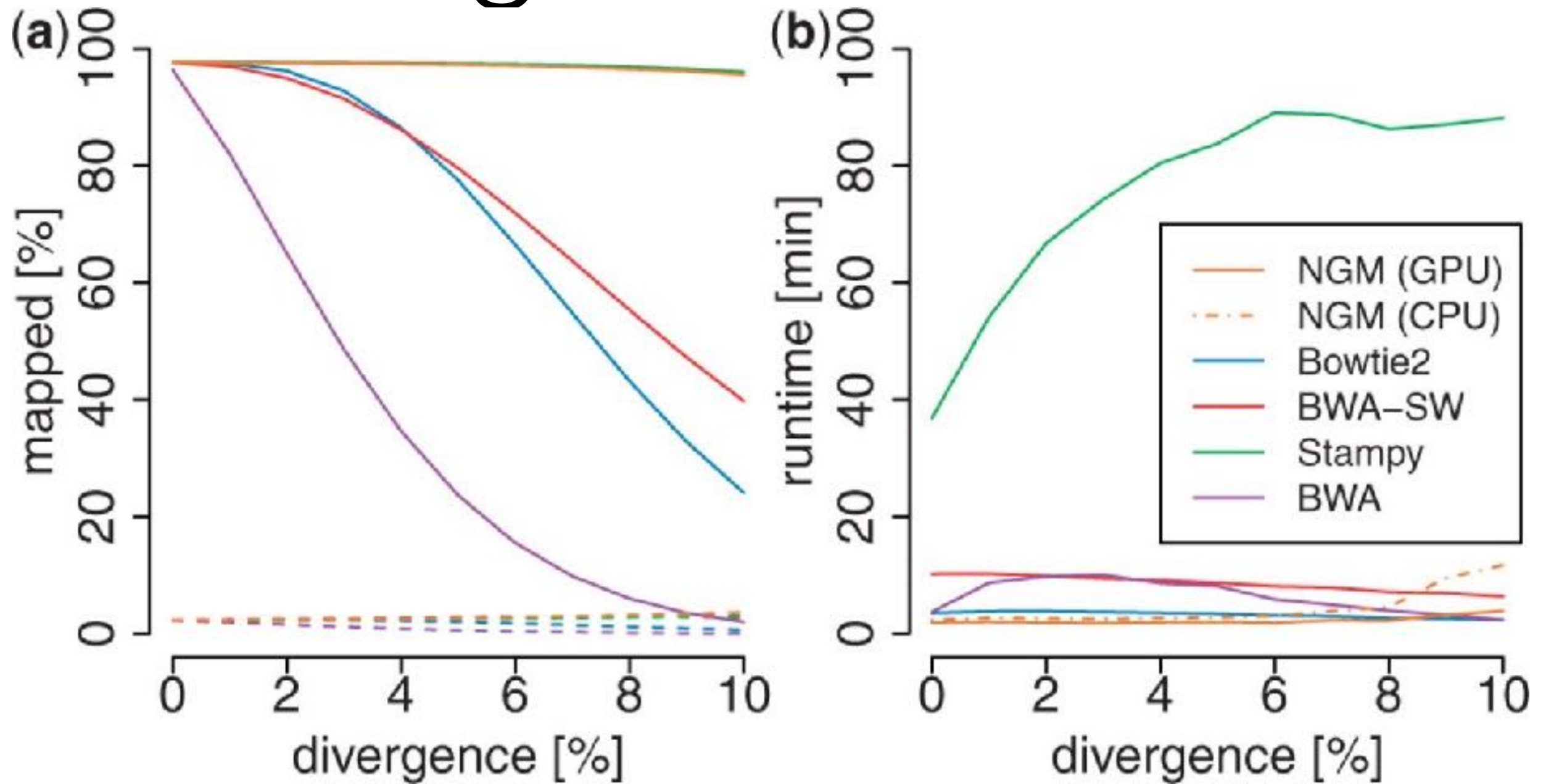
	Software	Reads aligned (%)	Time (paired, s)	Time (single, s)	Memory usage (GB)
Suffix Hash	SOAP2	93.6	828	478	5.4
	SOAP	93.8	19 234	14 328	14.7
Hash Suffix	MAQ	93.2	22 506	19 847	1.2
	Bowtie	91.7	–	405	2.3

Table from
Li et al Bioinformatics. 2009

Popular short read aligners

Program	Algorithm	Speed	Accuracy in 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



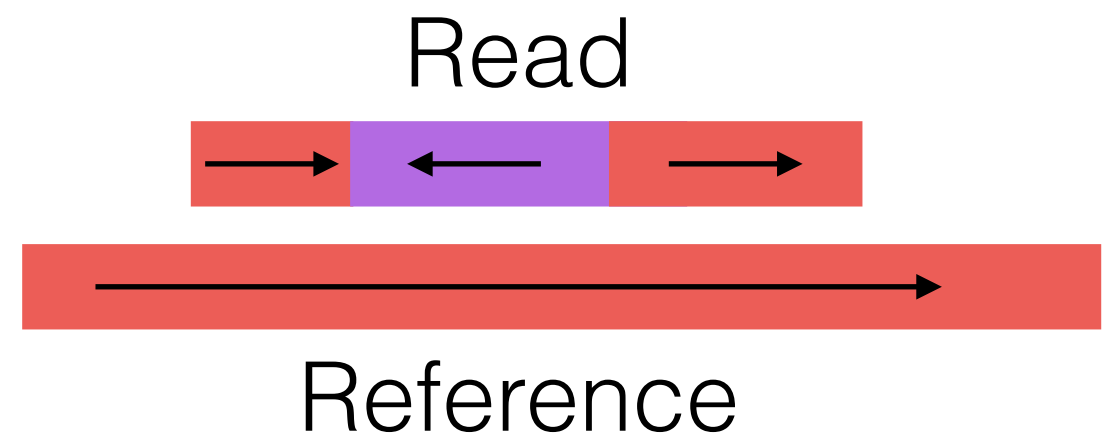
*From NextGenMap paper

Long read alignment with NGMLR

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

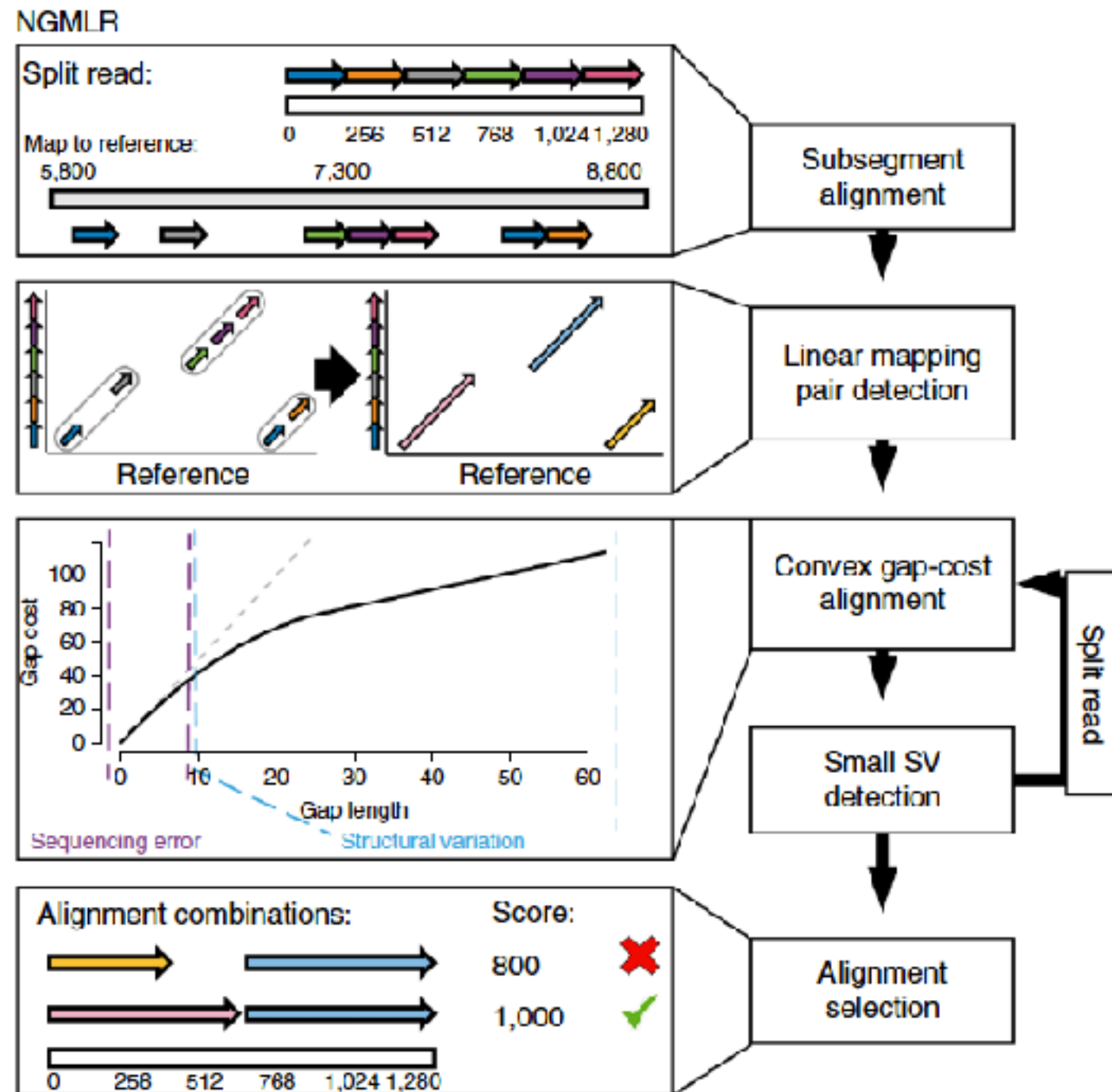


Long read alignment with NGMLR

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

Sort order - in this case coordinate based

```
@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 --
```

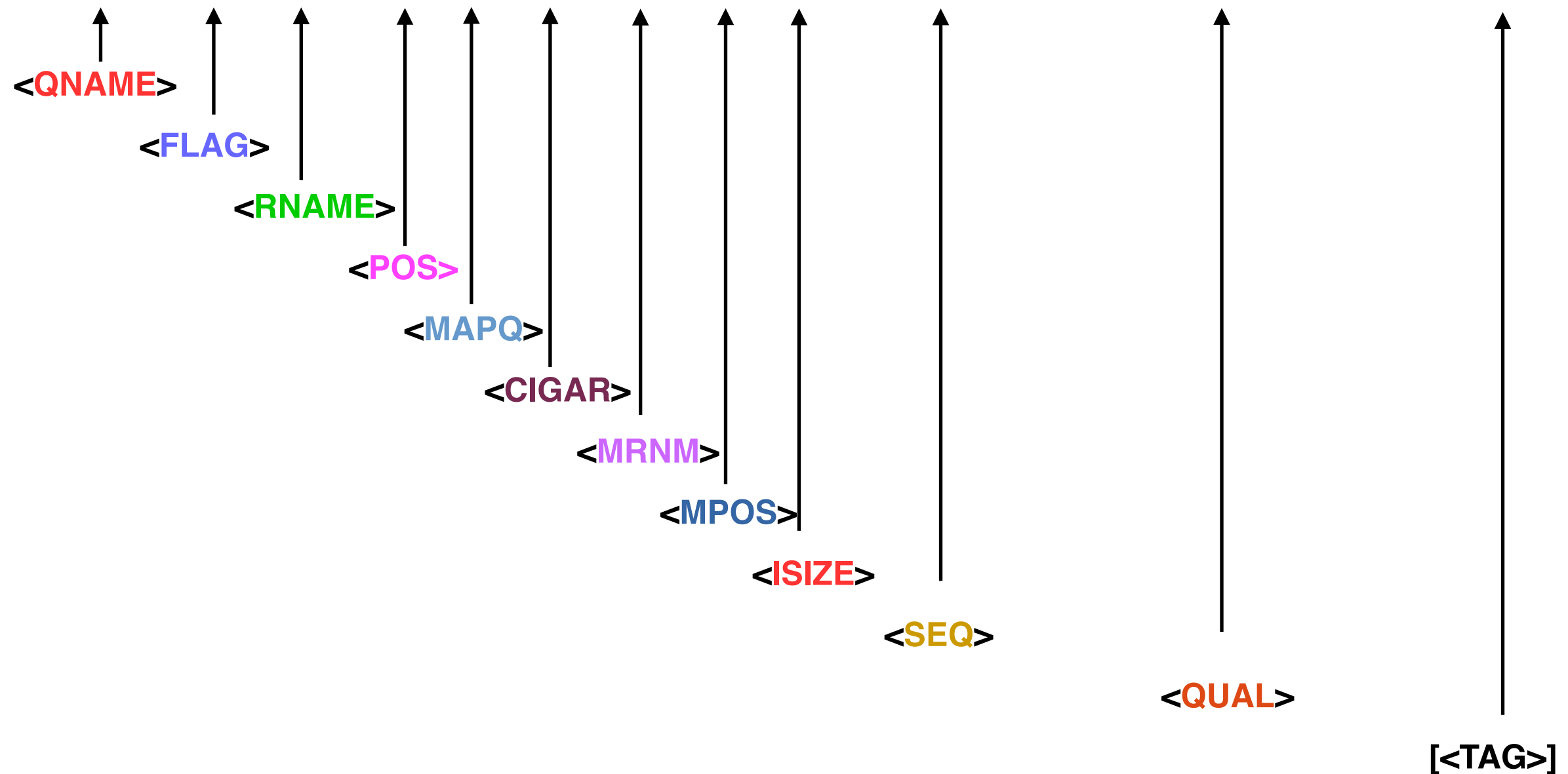
Reference sequence name (SN) and length (LN)
e.g. Chromosome Ha7, which is 109,221,022bp long

Read group (@RG) information

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

SAM format: Read lines

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



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> For an explanation of bitwise flags:

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

<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)

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

Mapping Quality

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

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

