"THIS IS A PROJECT THAT WILL TOUCH EVERY HUMAN LIFE."

It is now cheap and easy to obtain millions of short reads.

Applied Biosystems
ABI 3730XL

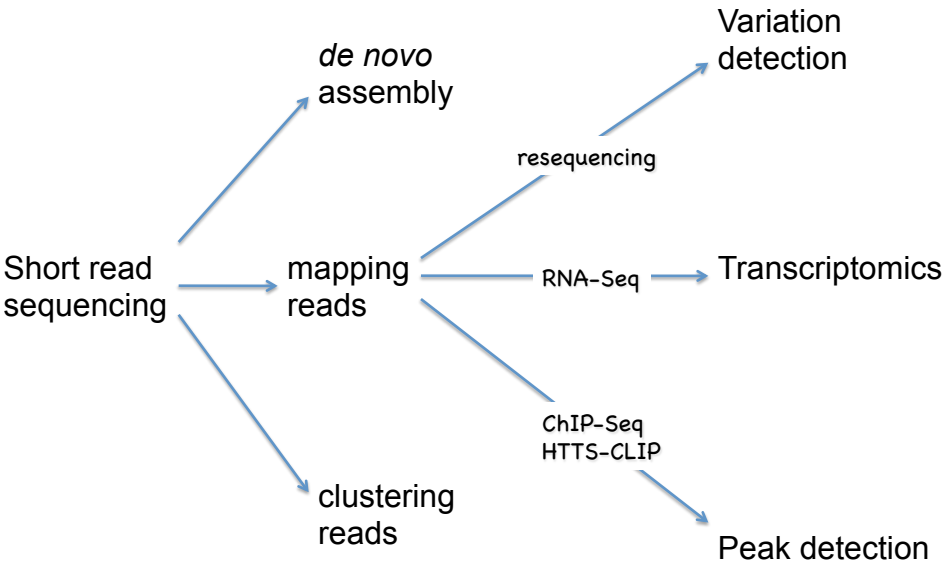
Applied Biosystems
SOLiD

Illumina/Solexa
HiSeq

Helicos
Heliscope

Roche 454
Genome Sequencer FLX

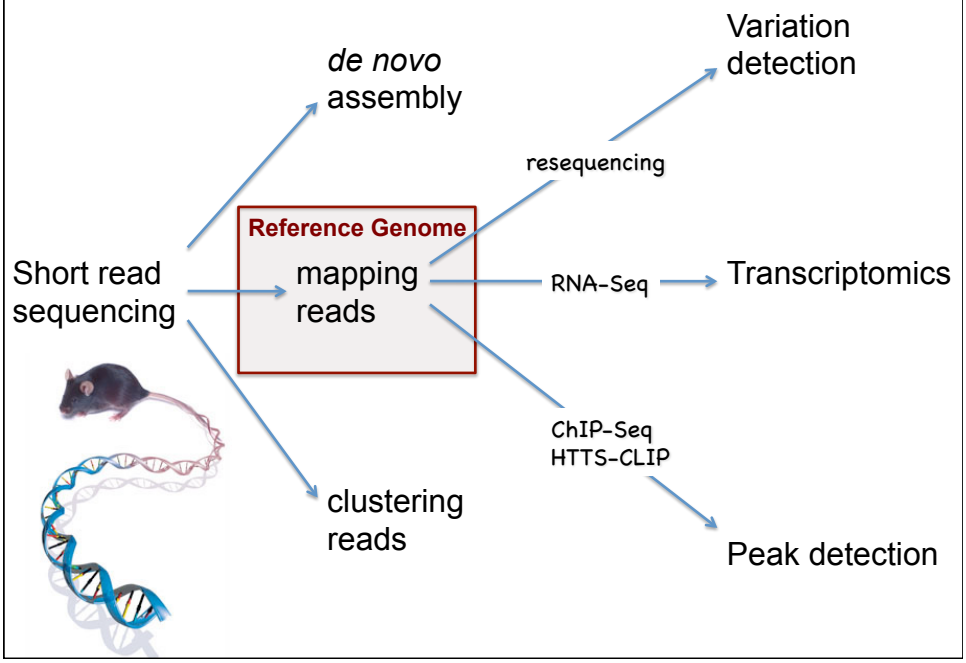
Illumina/Solexa
Genome Analyzer II



BLAST/BLAT too slow for millions of queries

- Noted during EST (expressed sequence tags) projects
  - -- aligning 3-13 million ESTs
  - – BLAT was a specialized aligner for closely related ESTs: > 95% identical, > 40 bp sequences
- Special case: We have a reference genome, so we KNOW apriori that we are dealing with HIGHLY similar sequences.

Mapping reads to a reference genome a common problem in modern sequencing.



Modern short read sequencing

- Average short read sequencing lane produces 50-800 million short (24-250 bp) reads.
- Using BLAT to align one lane (~150 million 50 bp reads) to a reference takes roughly a week, using BLAST takes even longer.
- Need yet another order of magnitude increase in speed – how???

How do we align the bag of reads to the reference?

- Efficiently (memory and time)
- Account for inexact matches and ambiguity?

Read mapping problem

- Sequenced reads are short
- Must be mapped to a unique position in a large genome
- Reads have potential sequencing errors
- Reference genome has repetitive elements
- Orientation of read relative to reference unknown
- Genome origin of read may be diverged from reference genome.

Short read mapping

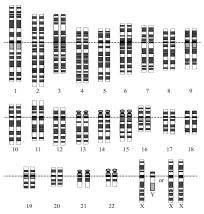
- Input:
  - A reference genome
  - A collection of many 25-250bp tags/reads
  - User-specified parameters
- Output:
  - One or more genomic coordinates for each tag
- In practice, not all reads successfully map to the reference genome. Why?

Determining the identity and location of short sequence reads in the genome/exome/transcriptome

```
@HWI-ST974:58:C059FACXX:2:1201:10589:110434 1:N:0:TGACCA
TGCACACTGAAGGACCTGGAATATGGCGAGAAAACTGAAAATCATGGAAAATGAGAAATACACACTTTAGGACGTG
```

Aligning short reads to much larger reference

```
          TAGATTACACAGATTAC
          ||||||||||||||||||
<-- TAGT TAGATTACACAGATTAC TAGA -->
```



Need a computationally efficient method to perform accurate alignments of millions of reads

Algorithms for mapping reads …

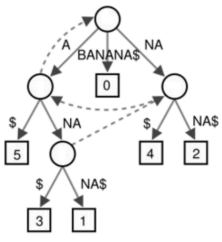
- Dynamic Programming (Smith Waterman)
  - Indels
  - Mathematically optimal solution
  - Slow (most programs, like BLAST, use Hash Mapping as a prefilter) [O(mnN)] m = genome; n = query/read; N = number reads
- Hash Table (Lookup table)
  - FAST, but requires perfect matches. [O(m n + N)]
- Array Scanning (Suffixes)
  - Can handle mismatches, but not gaps. [O(m N)]
- Burrows-Wheeler Transform (BW Transform)
  - FAST. [O(m + N)] (without mismatch/gap)
  - Memory efficient.
  - But for gaps/mismatches, it lacks sensitivity

Indexing

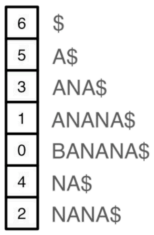
- The key to speeding up matching perfect short reads is to tightly INDEX the genome.
- There are multiple strategies for building indicies:

> 2000X faster than BLAST

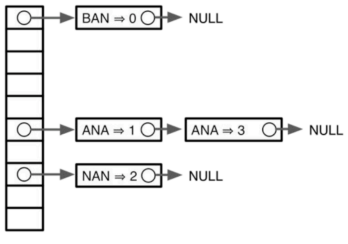
Suffix Tree:



Suffix Array:



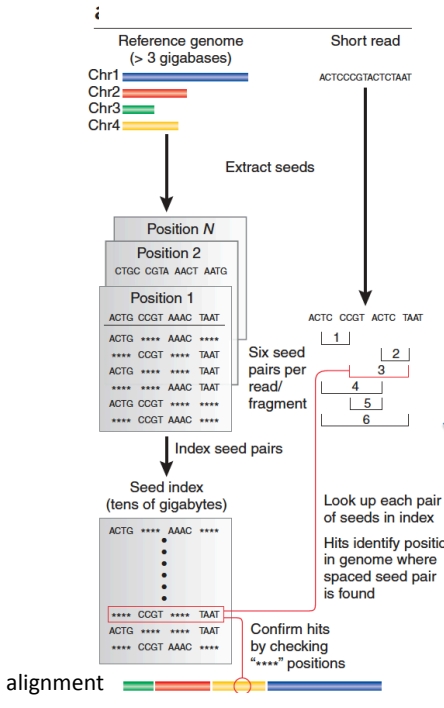
Hashing:



Strategy #1: Hashing

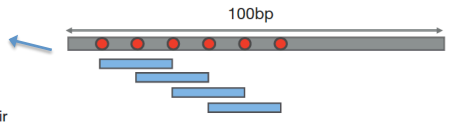
- We've discussed hashing before in class.
- Hashing strategies (seed hits) are used to speed up BLAST.
- For short read mapping, the trick is to rely on the presence of PERFECT matches.

- Gain speed by using larger seeds and requiring fixed spacing (harder to "trick" the hash table)



Spaced-seed indexing of the reference genome

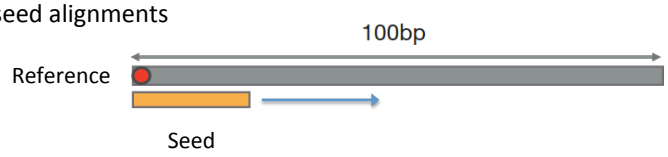
- Need to break up the genome into manageable segments
- Create index of short sequences
- Match seeds against genome index



Trapnell and Salzberg, *Nat Biotechnology* 27:455 (2009)

Seed-based alignment strategy

Single seed alignments

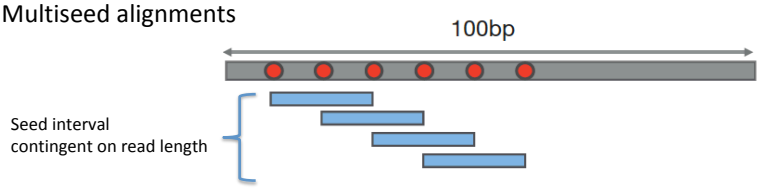


Critical values are seed length and number of mismatches allowed

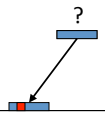
Example:

Seed length = 32

Multiseed alignments



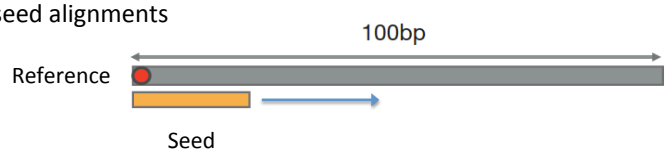
Difficulty #1: Inexact matching



- An observed tag may not exactly match any position in the reference genome.
- Sometimes, the tag *almost* matches one or more positions.
- Such mismatches may represent a SNP (single-nucleotide polymorphism) or a bad read-out.
- The user can specify the maximum number of mismatches, or a phred-style quality score threshold.
- As the number of allowed mismatches goes up, the number of mapped tags increases, but so does the number of incorrectly mapped tags.

Seed-based alignment strategy

Single seed alignments

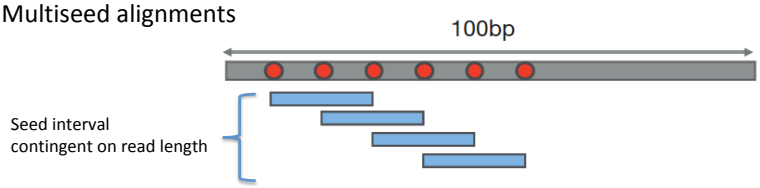


Critical values are seed length and number of mismatches allowed

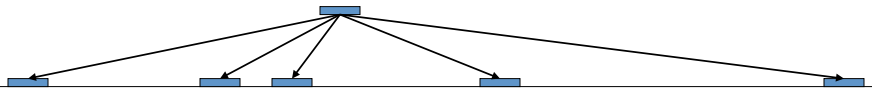
Example:
Seed length = 32
mismatch = 2

BUT mismatches increase index size and slow the whole algorithm down.

Multiseed alignments

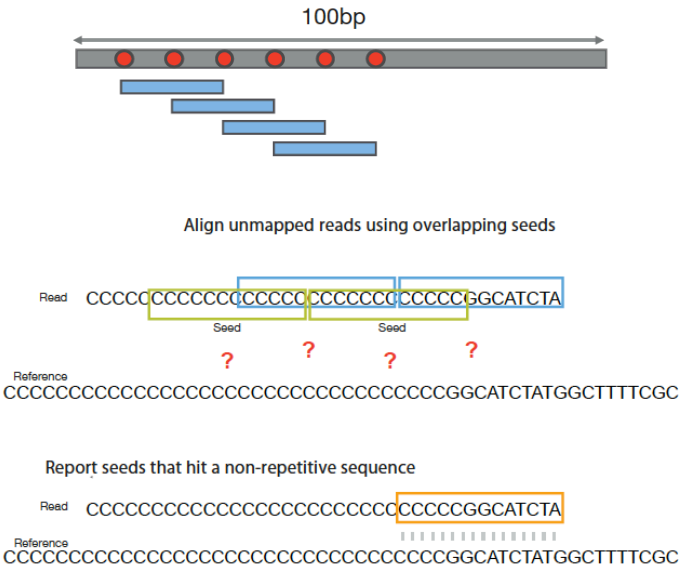


Difficulty #2: Multiple mapping



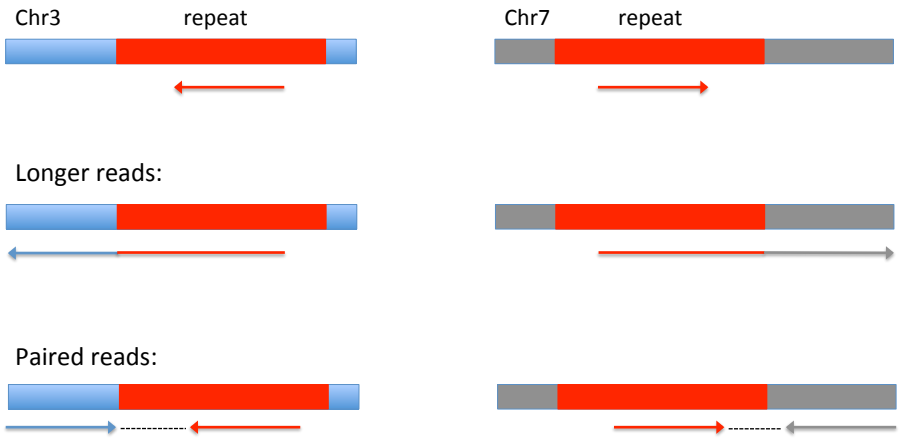
- A single tag may occur more than once in the reference genome.
- The user may choose to ignore tags that appear more than *n* times.
- As *n* gets large, you get more data, but also more noise in the data.

Resolving ambiguous read alignments with multiple seeds



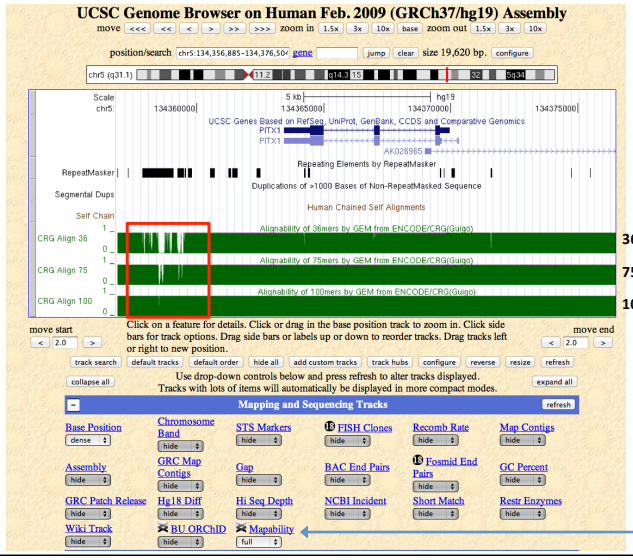
Mapability

- The genome contains non-unique sequences (repeats, segmental duplications)
- Short reads derived from repetitive regions are difficult to map

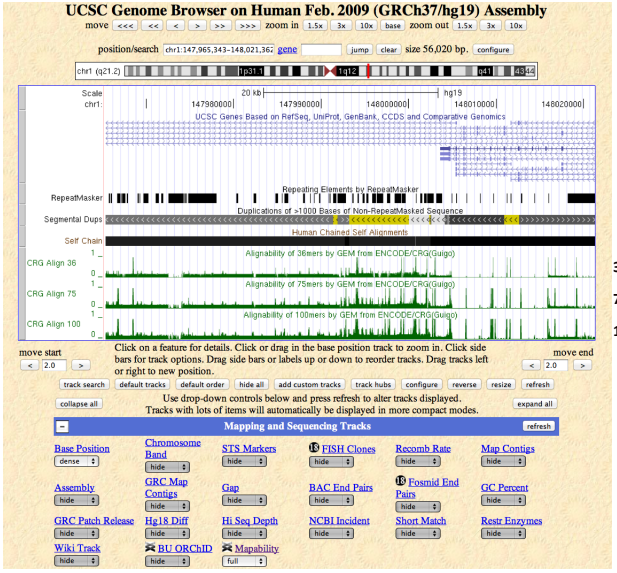


Mapability scores at UCSC

- The genome contains non-unique sequences (repeats, segmental duplications)
- Short reads derived from repetitive regions are difficult to map



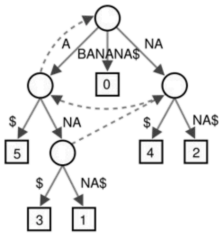
Poorly mappable regions of the genome



Indexing

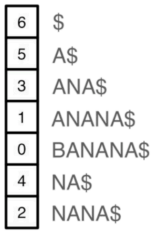
- The problem with these indexing strategies is that the size of the index is LARGE.

Suffix Tree:



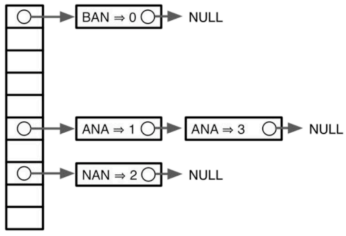
> 35 GBs

Suffix Array:



> 12 GBs

Hashing:

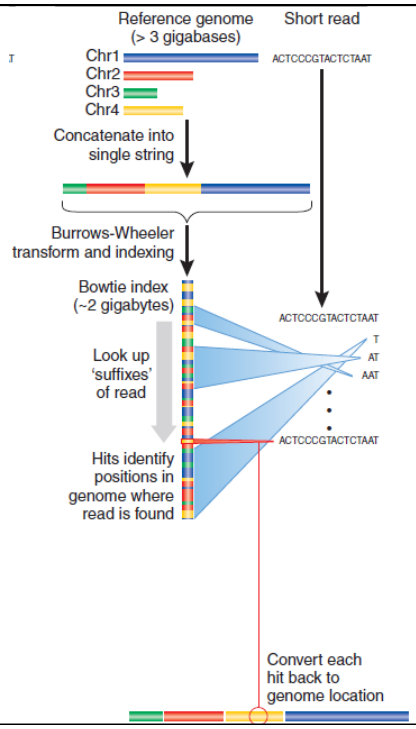


> 12 GBs

- Earliest short read aligners used assorted tricks and heuristics, but suffered from size.

Strategy #2: Burrows-Wheeler Transform

- This is a REVERSIBLE transformation of a string that produces RUNs of similar characters.
- Consequently it his HIGHLY compressible.
  - Approximately ½ byte per base
  - As large as the original text, plus a few "extras"
  - Can fit onto a standard computer with 2GB of memory
- Linear-time search algorithm (FM-index)
  - proportional to length of query for exact matches

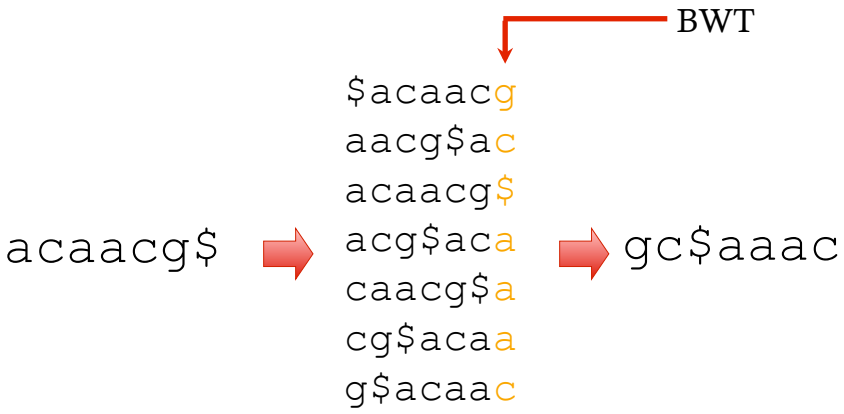


Burrows-Wheeler

- Store entire reference genome (encoded) with small auxiliary info (FM-index)
- Align tag base by base from the end.
- When tag is traversed, all active locations are reported.
- If no match is found, then back up and try a substitution.

> 200X faster than Hash Strategies

Burrows-Wheeler Transform (BWT)

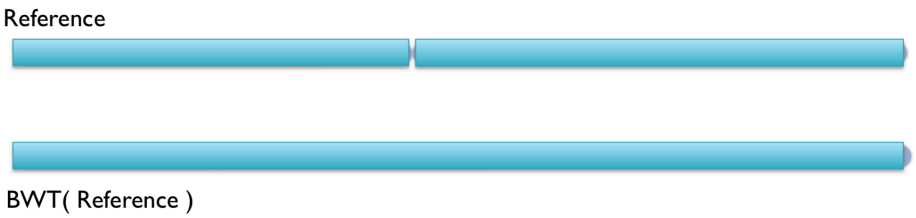


A suffix tree is much more efficient if it is created from the BWT sequence, rather than from the original sequence!

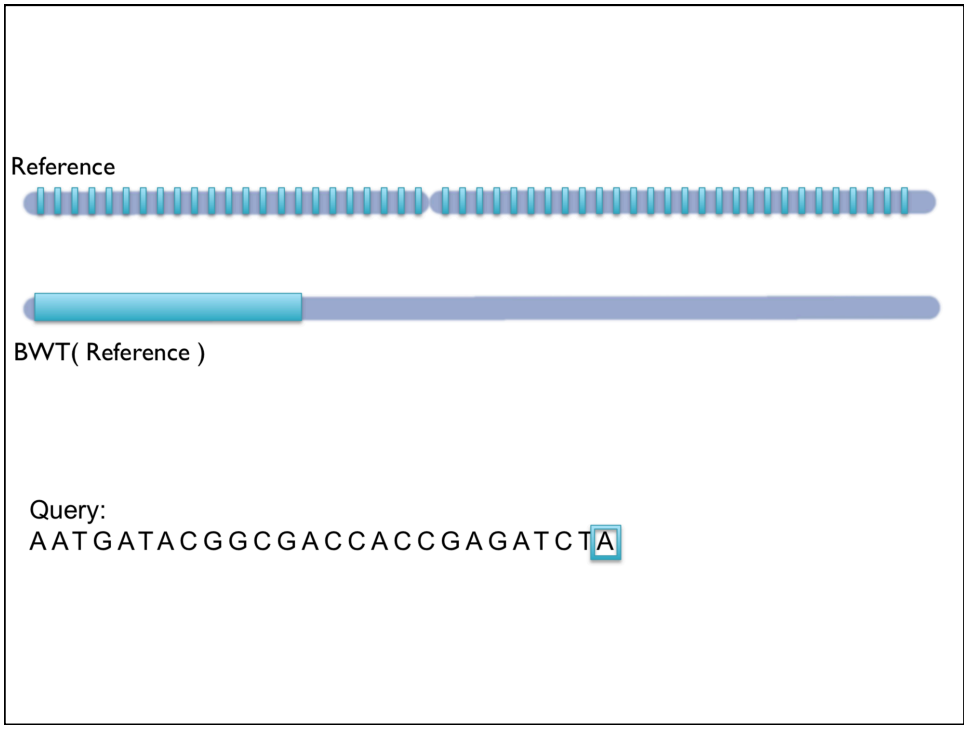
Bowtie

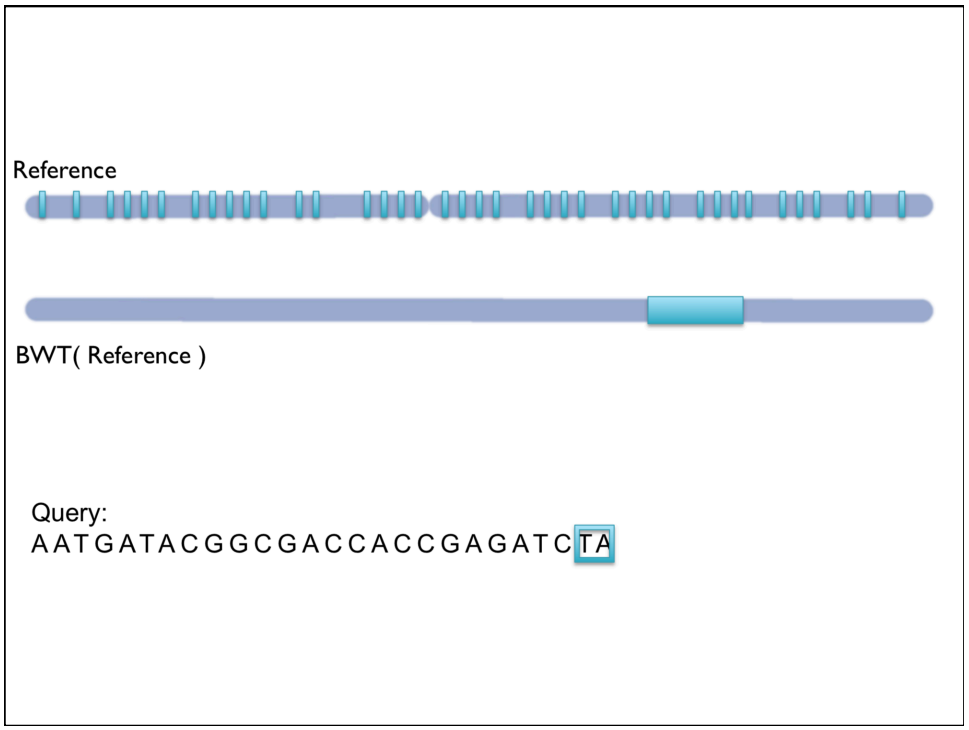
- Indexes the reference genome using a scheme based on the Burrows-Wheeler transform (BWT).
- A quality-aware backtracking algorithm that allows mismatches and favors high-quality alignments.
- Double indexing', a strategy to avoid excessive backtracking

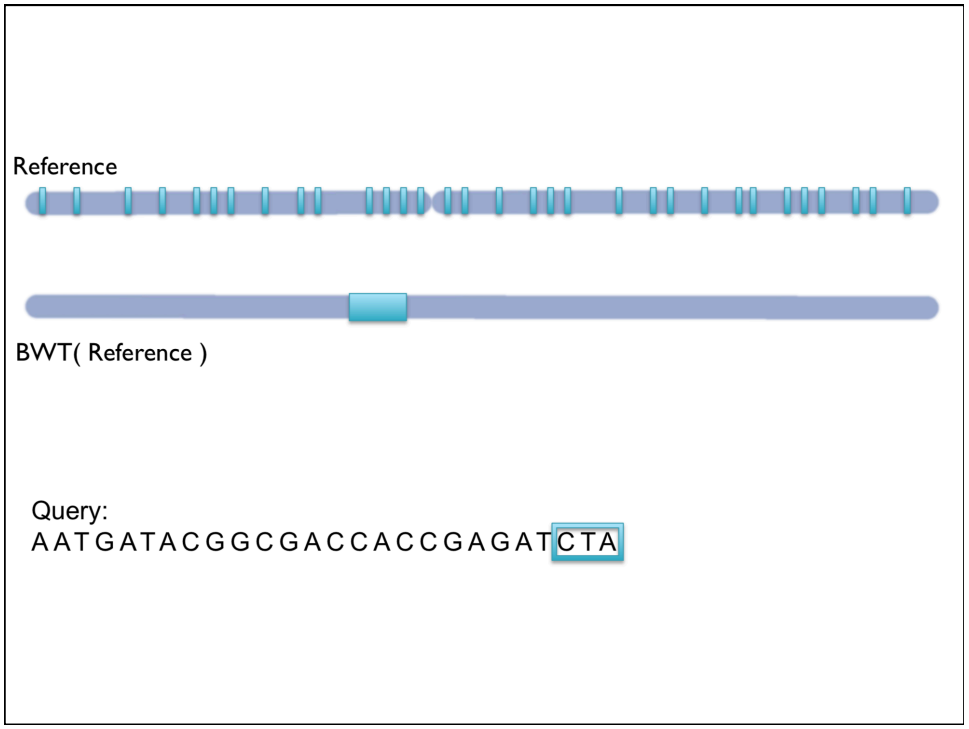
Bowtie conducts a quality-aware, greedy, randomized, depth-first search through the space of possible alignments.

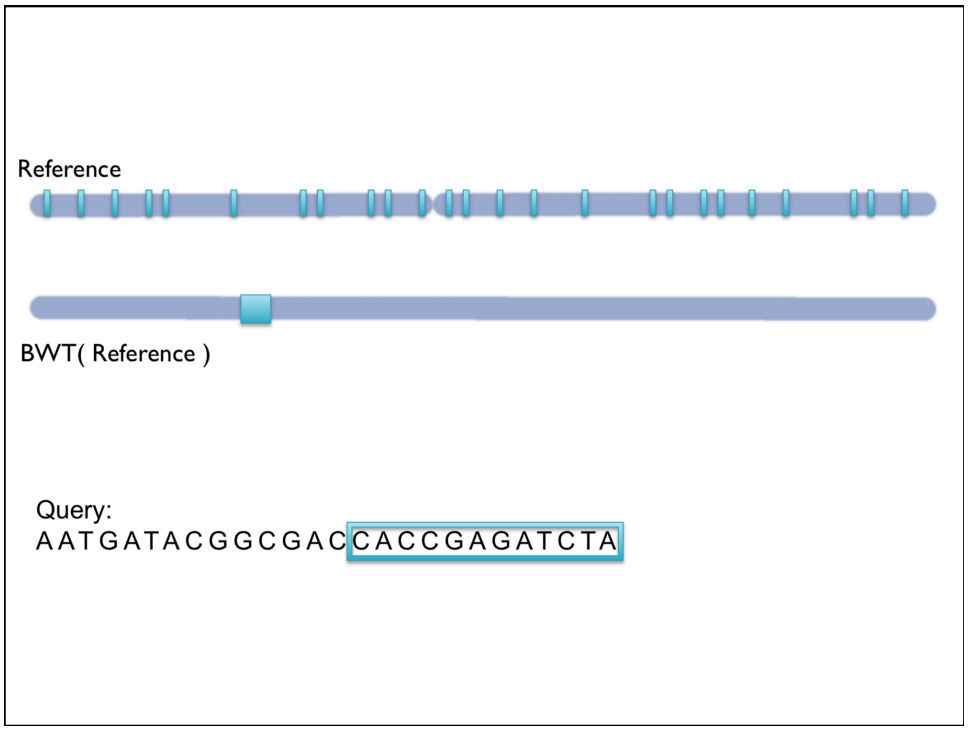


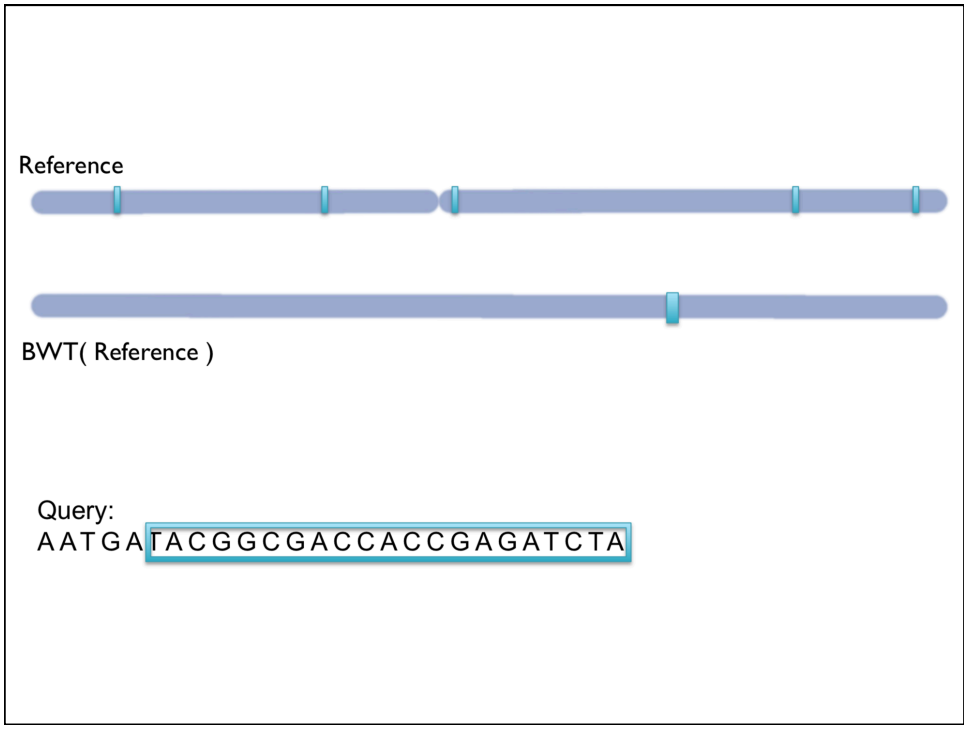
Query:
AATGATACGGCGACCACCGAGATCTA

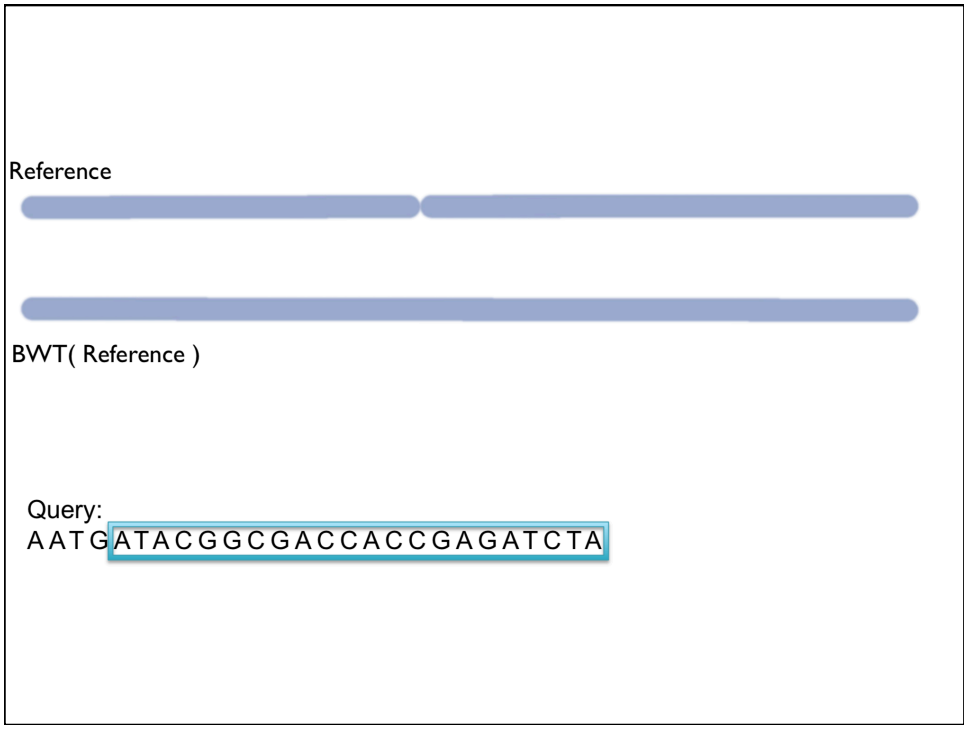

Reference
BWT(Reference)
Query:
AATGATACGGCGACCACCGAGATCTA

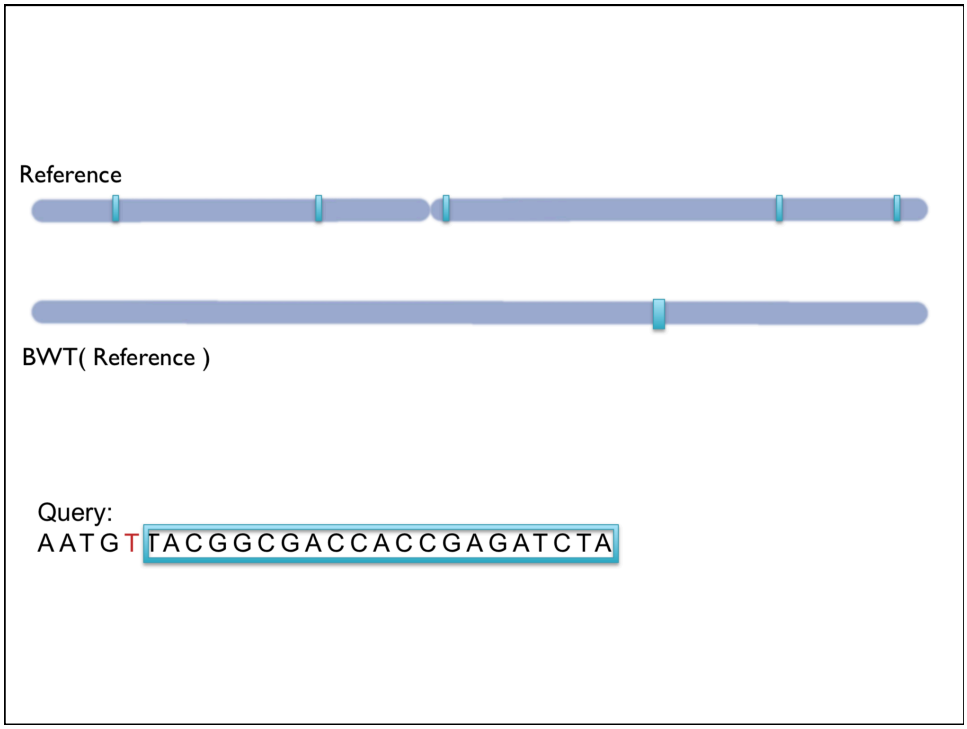

Reference
BWT(Reference)
Query:
AATGATACGGCGACCACCGAGATCTA

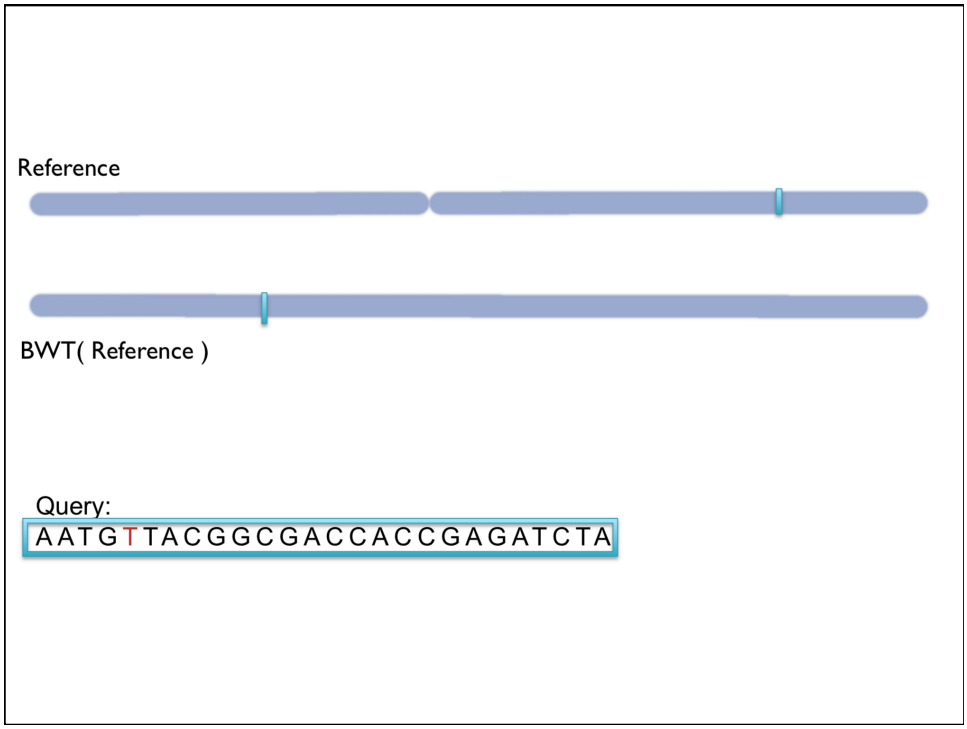

Reference
BWT(Reference)
Query:
AATGATACGGCGACCACCGAGATCTA


Reference
BWT(Reference)
Query:
AATGATACGGCGACCACCGAGATCTA


Reference
BWT(Reference)
Query:
AATGATACGGCGACCACCGAGATCTA


Reference
BWT(Reference)
Query:
AATGATACGGCGACCACCGAGATCTA


Reference
BWT(Reference)
Query:
AATGTTACGGCGACCACCGAGATCTA


Reference
BWT(Reference)
Query:
AATGTTACGGCGACCACCGAGATCTA

Bowtie caveat

"If one or more exact matches exist for a read, then Bowtie is guaranteed to report one, but if the best match is an inexact one then Bowtie is not guaranteed in all cases to find the highest quality alignment."

…unless you use the MUCH slower "best" option

Comparison

Spaced seeds (Hash strategy)

- Requires ~50Gb of memory.
- Runs 30-fold slower.
- Is much simpler to program.

Eland, MAQ, Soap

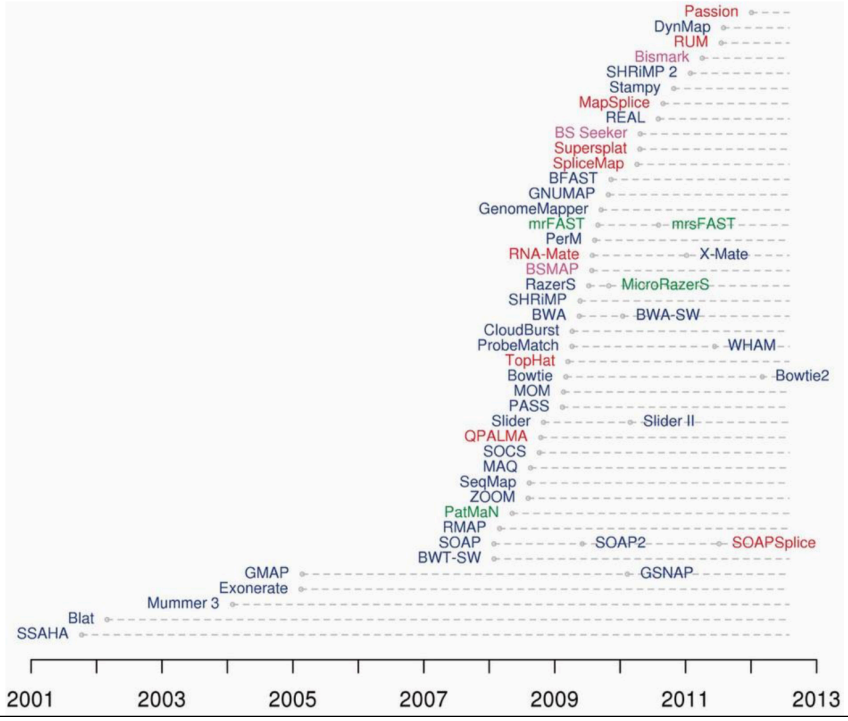
Burrows-Wheeler

- Requires <2Gb of memory.
- Runs 30-fold faster.
- Is much more complicated to program.

Bowtie, BWA

Short-read mapping software

Software	Technique	Developer	License
Eland	Hashing reads	Illumnia	?
SOAP	Hashing refs	BGI	Academic
Maq	Hashing reads	Sanger (Li, Heng)	GNUPL
Bowtie	BWT	Salzberg/UMD	GNUPL
BWA	BWT	Sanger (Li, Heng)	GNUPL
SOAP2	BWT & hashing	BGI	Academic



Alignment quality score

Base quality values and mismatch positions in a candidate alignment are used to assign a probability value

Probability reflect likelihood that candidate position in genome would give rise to the observed read if its bases were sequenced at error rates corresponding to the read's quality values. Should also reflect "uniqueness".

Alignment score for a read is computed from probability values of all candidate alignments.

If there are two candidate alignments for a read with probabilities values 0.9 and 0.3:

- 0.9/(0.9+0.3) = 0.75, chance highest scoring alignment is correct
- 1- 0.75, chance highest scoring alignment is wrong
- Alignment score = -10 log(0.25) = 6.

Sequence Alignment/Map (SAM)

```
Coor     12345678901234  5678901234567890123456789012345
ref      AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1        TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                     ATAGCT..............TCAGC
-r003                           ttagctTAGGC
-r001/2                                        CAGCGGCAT
```

The corresponding SAM format is:[1]

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

Sequence Alignment/Map (SAM) format

http://samtools.sourceforge.net/

Standard format for reporting short read alignment data

- BAM is compressed version

Header / Alignment info

```
@HD VN:1.3 SO:coordinate
@SQ SN:ref LN:45
r001 163 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002   0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003   0 ref  9 30 5H6M       *  0   0 AGCTAA   *   NM:i:1
r004   0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003  16 ref 29 30 6H5M       *  0   0 TAGGC    *   NM:i:0
r001  83 ref 37 30 9M         =  7 -39 CAGCGCCAT         *
```

Header:

```
@HD VN:1.0 SO:coordinate
@SQ SN:chr1 LN:249250621 AS:NCBI37
@SQ SN:chr2 LN:243199373 AS:NCBI37
@SQ SN:chr3 LN:198022430 AS:NCBI37
@RG ID:UM0098:1 PL:ILLUMINA PU:HWUSI-EAS17071
@PG ID:bwa VN:0.5.4
```

← Sort order

← Chromosomes and length

← read description line

← program info

Read Information

Field	Description	Alignment
QNAME	Query template name	**HWI-ST1359:47:H1178ADXX:2:2201:20747:42719**
FLAG	bitwise flag	**99**
RNAME	Reference name	**chr13**
POS	Reference position	**28540518**
MAPQ	Mapping quality	**60**
CIGAR	CIGAR string	**101M**
MRNM/RNEXT	Reference next read/mate	**=**
MPOS/PNEXT	Position next/read mate	**28540656**
ISIZE/TLEN	Template length	**239**

Read Information

Field	Description	Alignment
SEQ	Sequence	CCTCAAACCCACTCCAGGCTGCCATT GGTACTCGCCCCTTTTTACAGATGAG GAAATGGAGAATCAGACCGGGTCACG CAGATAGTATCAGGCGGGGTTGG
QUAL	Base qualities	7783>446=;@6;B;@B56/>=7>;8 <;?@0>A>=;;7A==@@9BB<;497> 565179==2864:>C=083&&42;37 69<8==>=;=>>D=768==8:79
TAGs	Tags	XO:i:1 X1:i:0 AM:i:37 NM:i:0 SM:i:37 XM:i:0 XO:i:0 MQ:i:60 XT:A:U

Mapping Flags

Flag	Description
0x0001	the read is paired in sequencing, no matter whether it is mapped in a pair
0x0002	the read is mapped in a proper pair (depends on the protocol, normally inferred during alignment) [1]
0x0004	the query sequence itself is unmapped
0x0008	the mate is unmapped [1]
0x0010	strand of the query (0 for forward; 1 for reverse strand)
0x0020	strand of the mate [1]
0x0040	the read is the first read in a pair [1,2]
0x0080	the read is the second read in a pair [1,2]
0x0100	the alignment is not primary (a read having split hits may have multiple primary alignment records)
0x0200	the read fails platform/vendor quality checks
0x0400	the read is either a PCR duplicate or an optical duplicate