Molecular Evolution and **Bioinformatics**

Reference based sequence mapping

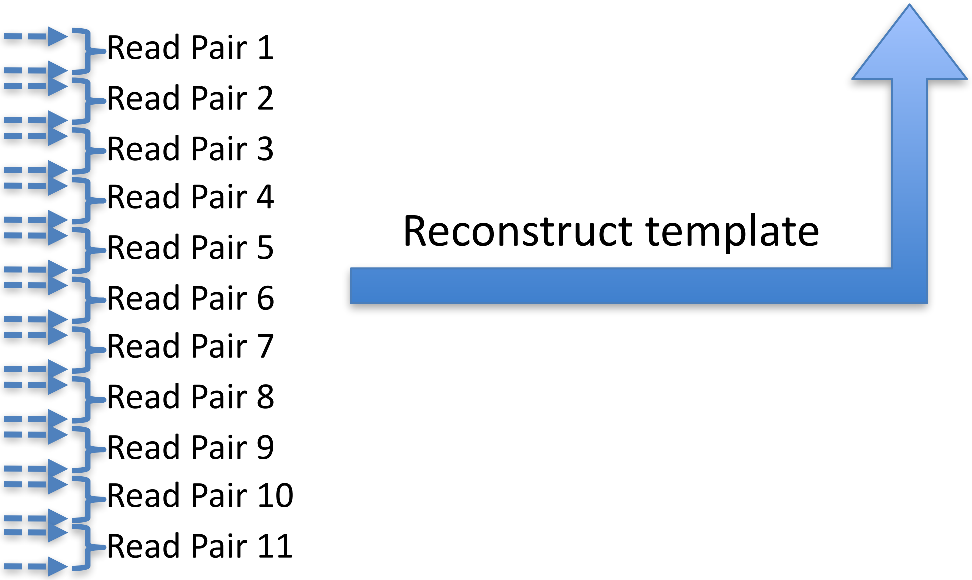
Literature

How to map billions of short reads onto genomes?
Trapnell and Salzberg Nature Biotechnology 27, 455 - 457 (2009)
doi:10.1038/nbt0509-455

Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.
Langmead et al. Genome Biology **10**:R25 (2009)
doi:10.1186/gb-2009-10-3-r25

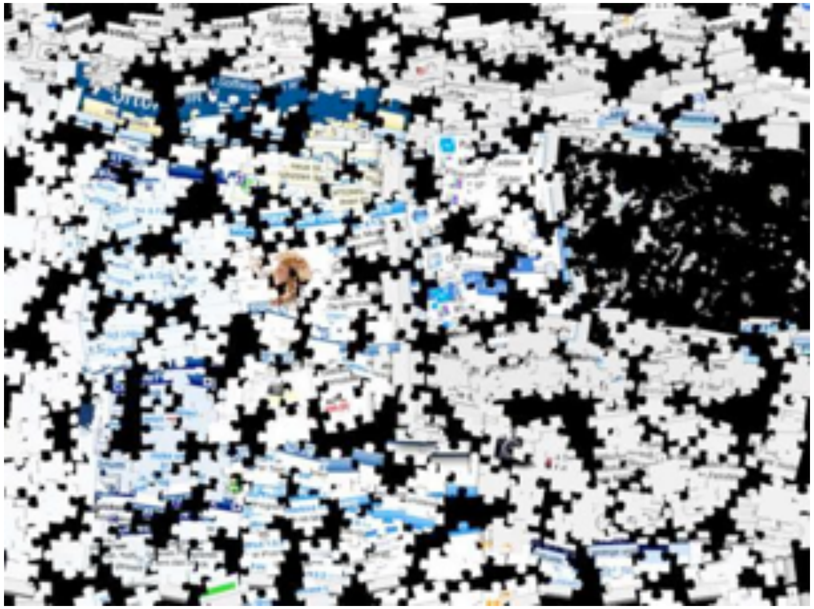
What do we need short read mapping for?

Assembly of whole genome shotgun sequencing data



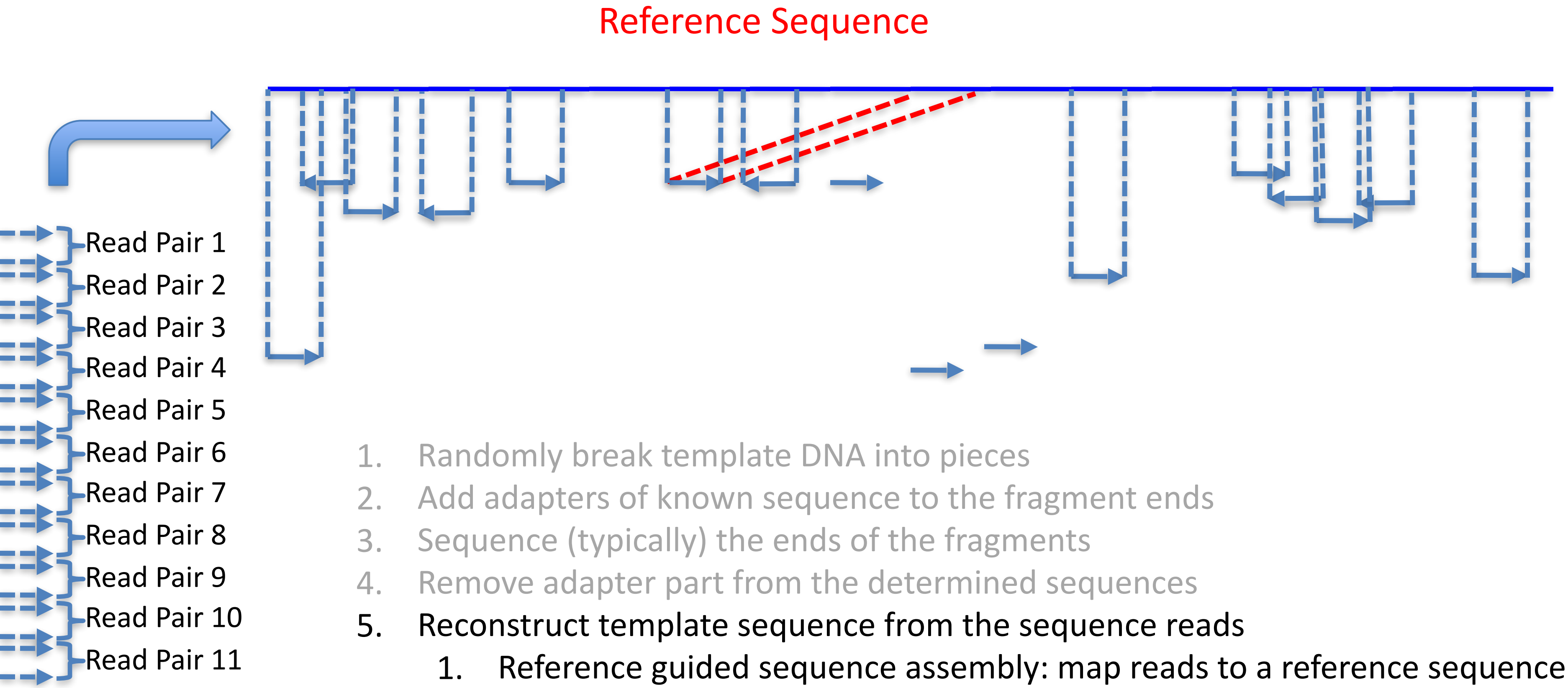
Read Pair 1
Read Pair 2
Read Pair 3
Read Pair 4
Read Pair 5
Read Pair 6
Read Pair 7
Read Pair 8
Read Pair 9
Read Pair 10
Read Pair 11

Reconstruct template



1. Randomly break template DNA into pieces
2. Add adapters of known sequence to the fragment ends
3. Sequence (typically) the ends of the fragments
4. Identify and remove adapter part from the determined sequences
5. Reconstruct template sequence from the sequence reads

Shotgun sequencing and **reference guided** sequence assembly



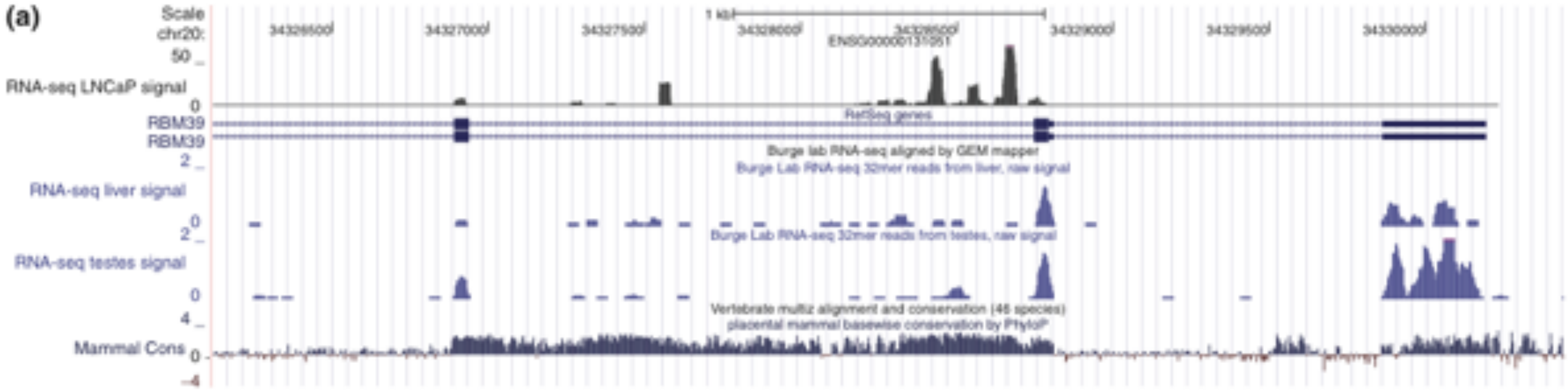
1. Randomly break template DNA into pieces
2. Add adapters of known sequence to the fragment ends
3. Sequence (typically) the ends of the fragments
4. Remove adapter part from the determined sequences
5. Reconstruct template sequence from the sequence reads
    1. Reference guided sequence assembly: map reads to a reference sequence. Note that reads can map equally good to more than one position in the reference genome, e.g. due to repeats in the reference.

—— Reference sequence

a) e.g. genome of a different individual from the same species to study species diversity

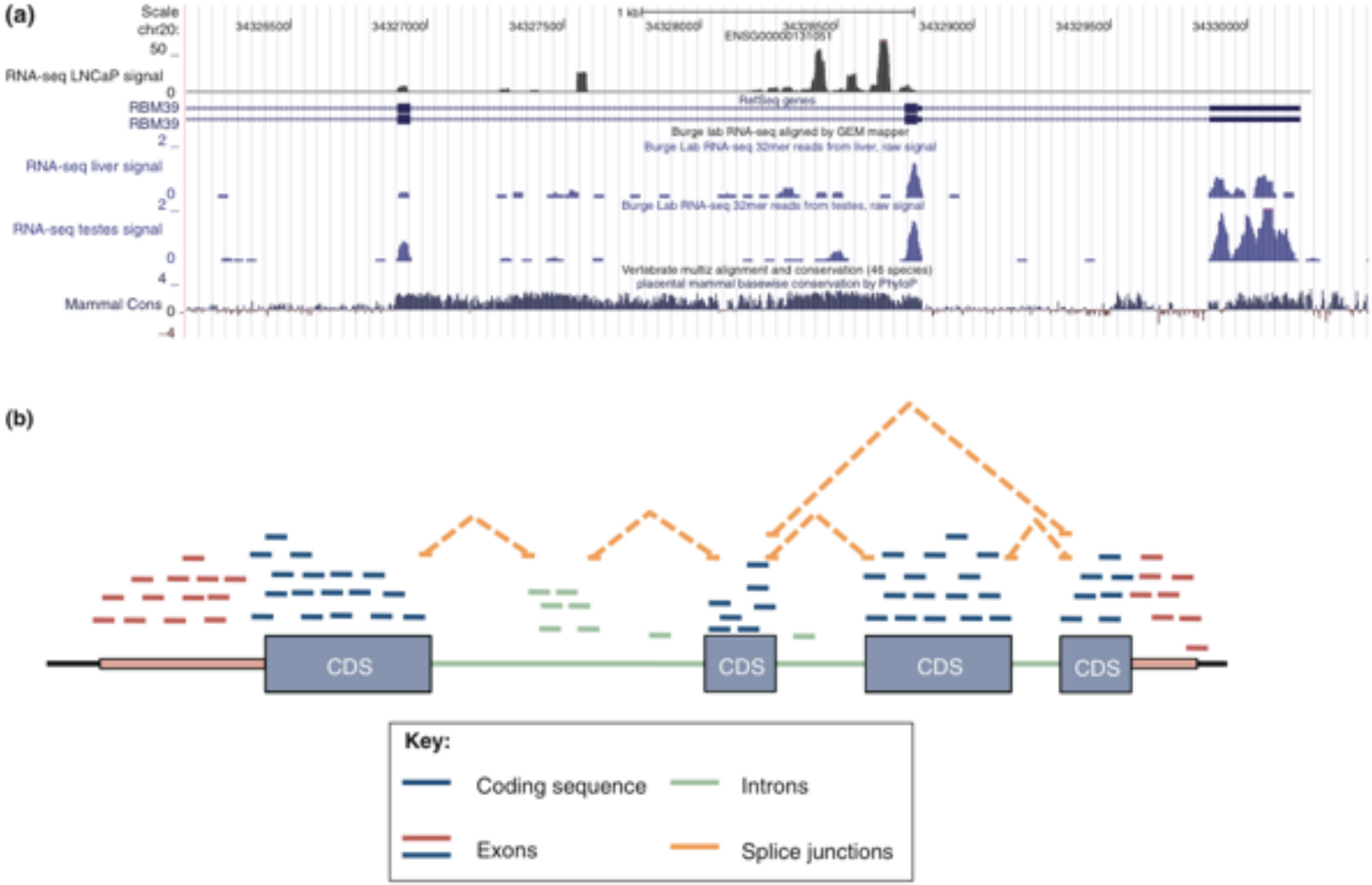
b) e.g. genome of a closely related species

Further short read applications: Mapping of RNA seq



from Oshlack et al. *Genome Biology* 2010, **11**:220

RNA-seq mapping helps building hypotheses concerning gene structure

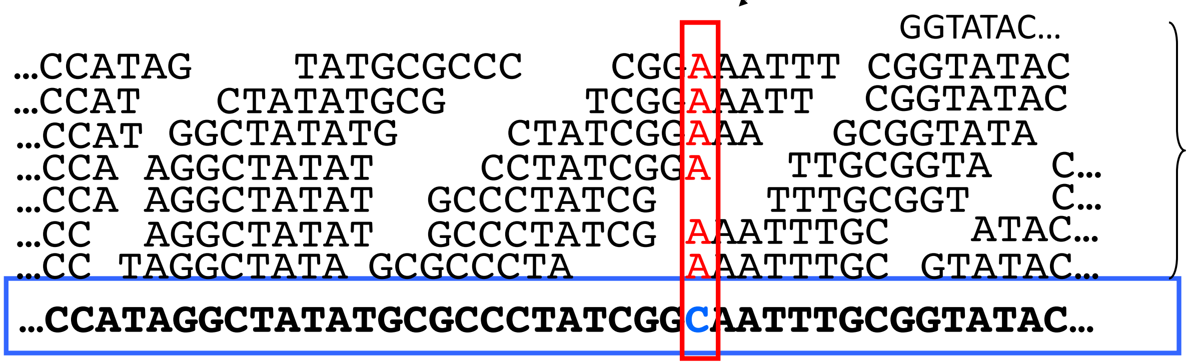


from Oshlack et al. *Genome Biology* 2010, **11**:220

Further short read applications

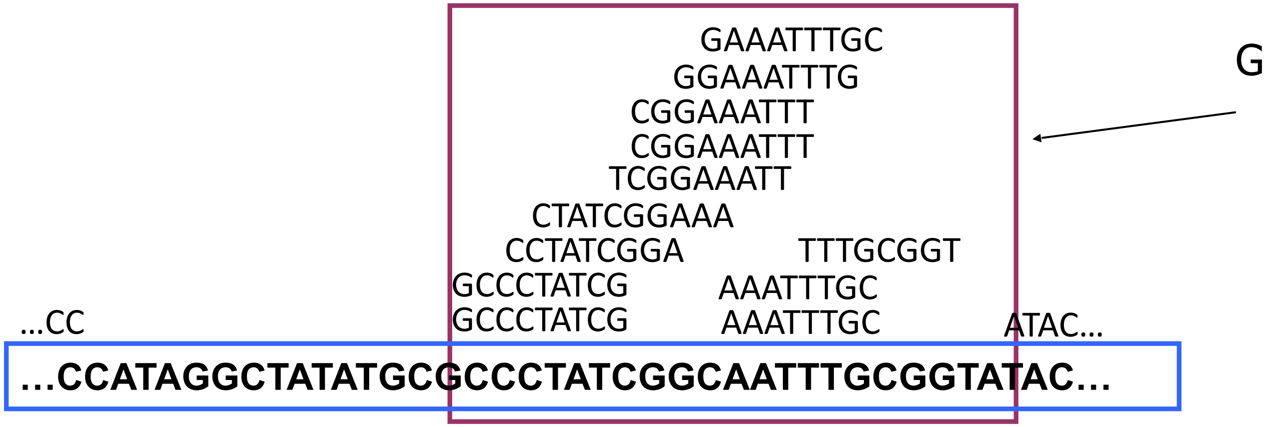
- Genotyping

Goal: identify variations



```
                                                    GGTATAC…
…CCATAG     TATGCGCCC    CGGAAATTT CGGTATAC
…CCAT    CTATATGCG      TCGGAAATT  CGGTATAC
…CCAT GGCTATATG     CTATCGGAAA   GCGGTATA
…CCA  AGGCTATAT    CCTATCGGA    TTGCGGTA   C…
…CCA  AGGCTATAT  GCCCTATCG     TTTGCGGT    C…
…CC   AGGCTATAT  GCCCTATCG  AAATTTGC    ATAC…
…CC  TAGGCTATA GCGCCCTA     AAATTTGC  GTATAC…
…CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC…
```

- ChIP-seq, CLIP-Seq, Methyl-seq



```
                      GAAATTTGC
                     GGAAATTTG
                   CGGAAATTT
                   CGGAAATTT
                  TCGGAAATT
               CTATCGGAAA
              CCTATCGGA      TTTGCGGT
            GCCCTATCG      AAATTTGC
…CC         GCCCTATCG      AAATTTGC          ATAC…
…CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC…
```

Goal: classify, measure significant peaks

ChIP: Chromatin Immunoprecipitation, **CLIP:** Cross-linking immunoprecipitation

Challenges

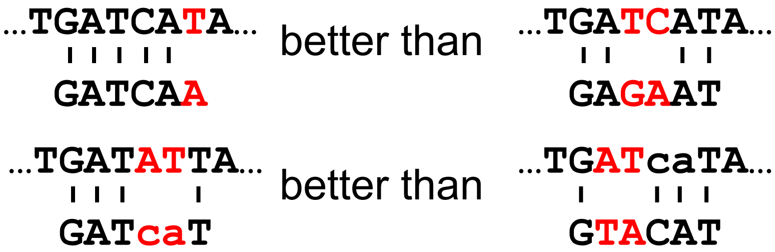
- mapping millions/billions of reads to a large genome is hard
  - how quickly can we map the reads to the genome?
  - how do we deal with multiple mapping positions?
  - how do we deal with sequencing errors and genetic divergence/diversity
  - how do we deal with reads that span intron-exon boundaries?

Short Read Alignment

- Given a reference and a set of reads, report at least one "good" local alignment for each read if one exists
  - Approximate answer to: from where in genome did the read originate?

What is "good"?

- – Fewer mismatches is better
- – Failing to align a low-quality base is better than failing to align a high-quality base



```
...TGATCATA...                 ...TGATCATA...
    |||||       better than        ||  ||
    GATCAA                         GAGAAT

...TGATATTA...                 ...TGATcaTA...
    |||  |      better than        |  |||
    GATcaT                         GTACAT
```

Finding mapping positions of reads in genomes is, in principle, a pattern matching problem. We search for the occurrence of short words in long sentences.

Finding mapping positions is, in principle, very easy!

```
Searchstring: AATGAGACATGAA

    Query1: CATG
    Query2: ATGT
```

Finding mapping positions is, in principle, very easy. Just slide the word along the sequence and stop when either end of sequence is reached or mapping position is found

```
Searchstring: AATGAGACATGAA

   Query1: CATG
   Query2: ATGT
```

Naïve approach

```
              1234567891234
Searchstring: AATGAGACATGAA
              CATG →
```

Finding mapping positions is, in principle, very easy. Just slide the word along the sequence and stop when either end of sequence is reached or mapping position is found

```
Searchstring: AATGAGACATGAA
```

```
Query1: CATG
Query2: ATGT
```

Naïve approach

```
              12345678912334
Searchstring: AATGAGACATGAA
              CATG
              ✖✚✚✚
```

Finding mapping positions is, in principle, very easy. Just slide the word along the sequence and stop when either end of sequence is reached or mapping position is found

```
Searchstring: AATGAGACATGAA

   Query1: CATG
   Query2: ATGT
```

Naïve approach

```
              1234567891234
Searchstring: AATGAGACATGAA
               CATG
               ✖✖✖✖
```

Finding mapping positions is, in principle, very easy. Just slide the word along the sequence and stop when either end of sequence is reached or mapping position is found

```
Searchstring: AATGAGACATGAA
```

```
Query1: CATG
Query2: ATGT
```

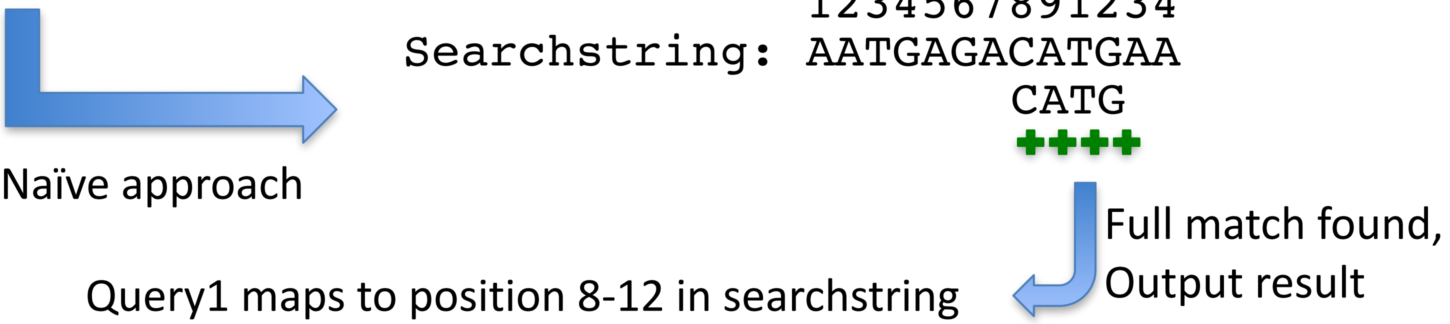
Naïve approach

```
              12345678912344
Searchstring: AATGAGACATGAA
                CATG
                ✖✖✖✚
```

Finding mapping positions is, in principle, very easy. Just slide the word along the sequence and stop when either end of sequence is reached or mapping position is found

```
Searchstring: AATGAGACATGAA
```

```
    Query1: CATG
    Query2: ATGT
```



Naïve approach

```
              1234567891234
Searchstring: AATGAGACATGAA
                     CATG
                     ++++
```

Full match found,
Output result

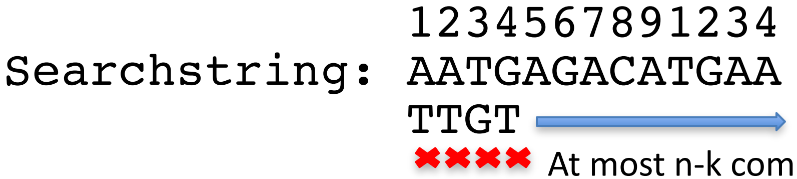
Query1 maps to position 8-12 in searchstring

Finding mapping positions is, in principle, very easy. Just slide the word along the sequence and stop when either end of sequence is reached or mapping position is found

```
Searchstring: AATGAGACATGAA
```

```
Query1: CATG
Query2: ATGT
```

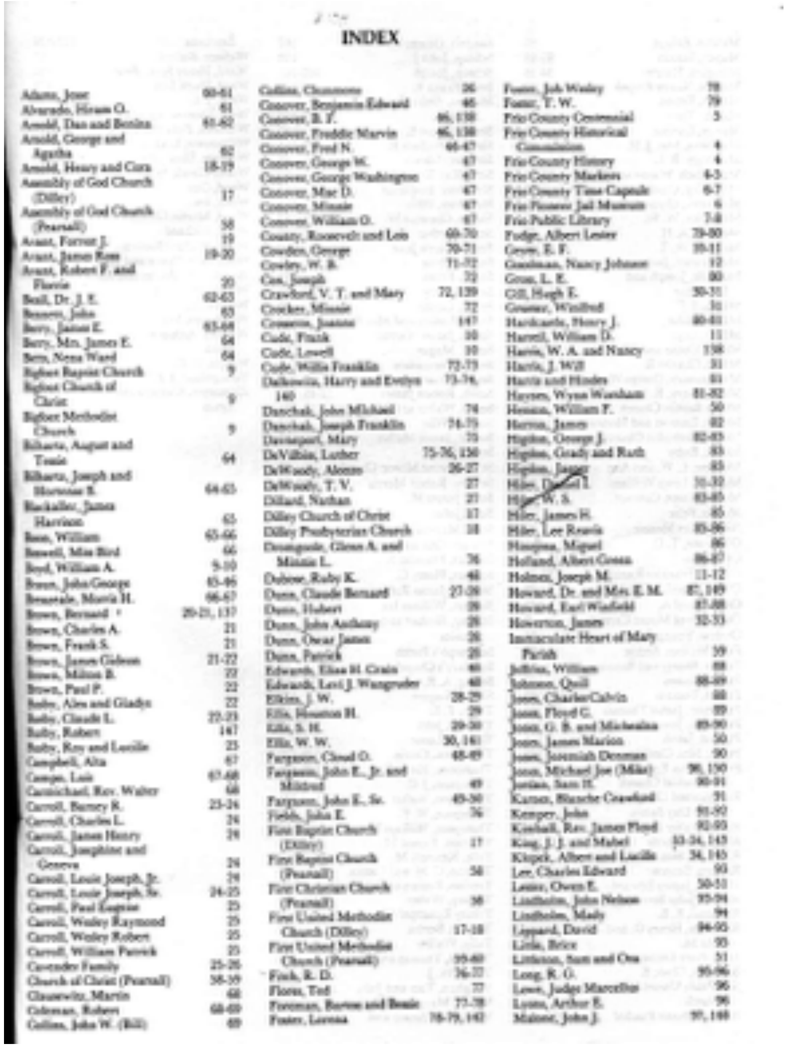
Naïve approach



```
              1234567891234
Searchstring: AATGAGACATGAA
              TTGT
              ✖✖✖✖
```

At most n-k comparisons, with n is the length of the search string, and k is the query length (read length).
This is not feasible for short read mapping.

Indexing speeds up searches



INDEX

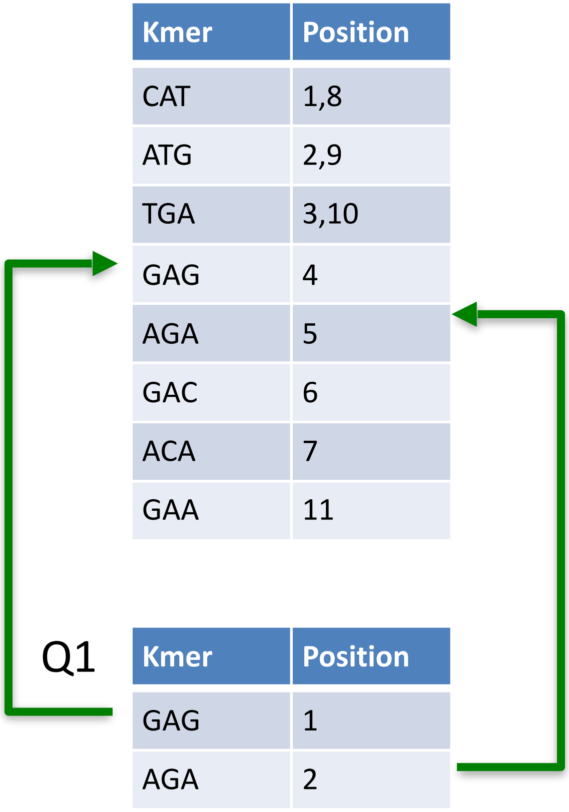
Adams, Jose 60-61
Alvarado, Hiram O. 61
Arnold, Dan and Benita 61-62
Arnold, George and Agatha 62
Arnold, Henry and Cora 18-19
Assembly of God Church (Dilley) 17
Assembly of God Church (Pearsall) 58
Avant, Forrest J. 19
Avant, James Ross 19-20
Avant, Robert F. and Flavia 20
Beall, Dr. J. E. 62-63
Bennett, John 63
Berry, James E. 63-64
Berry, Mrs. James E. 64
Betts, Nena Ward 64
Bigfoot Baptist Church 9
Bigfoot Church of Christ 9
Bigfoot Methodist Church 9
Bilharts, August and Tonie 64
Bilharts, Joseph and Hortense B. 64-65
Blackaller, James Harrison 65
Boon, William 65-66
Boswell, Miss Bird 66
Boyd, William A. 9-10
Braun, John George 65-66
Brazeale, Morris H. 66-67
Brown, Bernard 20-21, 137
Brown, Charles A. 21
Brown, Frank S. 21
Brown, James Gideon 21-22
Brown, Milton B. 22
Brown, Paul P. 22
Busby, Alex and Gladys 22
Busby, Claude L. 22-23
Busby, Robert 147
Busby, Roy and Lucille 23
Campbell, Alta 67
Campos, Luis 67-68
Carmichael, Rev. Walter 68
Carroll, Barney R. 23-24
Carroll, Charles L. 24
Carroll, James Henry 24
Carroll, Josephine and Geneva 24
Carroll, Louis Joseph, Jr. 24
Carroll, Louis Joseph, Sr. 24-25
Carroll, Paul Eugene 25
Carroll, Wesley Raymond 25
Carroll, Wesley Robert 25
Carroll, William Patrick 25
Cavender Family 25-26
Church of Christ (Pearsall) 58-59
Clausewitz, Martin 68
Coleman, Robert 68-69
Collins, John W. (Bill) 69
Collins, Chummons 26
Conover, Benjamin Edward 46
Conover, B. F. 46, 138
Conover, Freddie Marvin 46, 138
Conover, Fred N. 46-47
Conover, George W. 47
Conover, George Washington 47
Conover, Mae D. 47
Conover, Minnie 47
Conover, William O. 47
County, Roosevelt and Lois 69-70
Cowden, George 70-71
Cowley, W. B. 71-72
Cox, Joseph 72
Crawford, V. T. and Mary 72, 139
Crocker, Minnie 72
Crossetti, Joanne 147
Cude, Frank 10
Cude, Lowell 10
Cude, Willie Franklin 72-73
Dalkowitz, Harry and Evelyn 73-74, 140
Danchak, John Michael 74
Danchak, Joseph Franklin 74-75
Davenport, Mary 75
DeVilbiss, Luther 75-76, 150
DeWoody, Alonzo 26-27
DeWoody, T. V. 27
Dillard, Nathan 21
Dilley Church of Christ 17
Dilley Presbyterian Church 18
Dosangado, Glenn A. and Minnie L. 76
Dubose, Ruby K. 48
Dunn, Claude Bernard 27-28
Dunn, Hubert 28
Dunn, John Anthony 28
Dunn, Oscar James 28
Dunn, Patrick 28
Edwards, Eliza H. Crain 48
Edwards, Levi J. Wangruder 48
Elkins, J. W. 28-29
Ellis, Houston H. 29
Ellis, S. H. 29-30
Ellis, W. W. 30, 161
Farguson, Cloud O. 48-49
Farguson, John E., Jr. and Mildred 49
Farguson, John E., Sr. 49-50
Fields, John E. 76
First Baptist Church (Dilley) 17
First Baptist Church (Pearsall) 58
First Christian Church (Pearsall) 58
First United Methodist Church (Dilley) 17-18
First United Methodist Church (Pearsall) 59-60
Finch, R. D. 76-77
Flores, Ted 77
Foreman, Barton and Bessie 77-78
Foster, Lorena 78-79, 142
Foster, Job Wesley 78
Foster, T. W. 79
Frio County Centennial 3
Frio County Historical Commission 4
Frio County History 4
Frio County Markers 4-5
Frio County Time Capsule 6-7
Frio Pioneer Jail Museum 6
Frio Public Library 7-8
Fudge, Albert Lester 79-80
Geyer, E. F. 10-11
Goodman, Nancy Johnson 12
Gross, L. E. 80
Gill, Hugh E. 30-31
Graeser, Winifred 31
Hardcastle, Henry J. 80-81
Harrell, William D. 11
Harris, W. A. and Nancy 138
Harris, J. Will 31
Harris and Hinden 81
Haynes, Wynn Wareham 81-82
Henson, William F. 50
Herron, James 82
Higdon, George J. 82-83
Higdon, Grady and Ruth 83
Higdon, Jasper 83
Hiler, Daniel I. 31-32
Hiler, W. S. 83-85
Hiler, James H. 85
Hiler, Lee Rearis 85-86
Hinojosa, Miguel 86
Holland, Albert Green 86-87
Holmes, Joseph M. 11-12
Howard, Dr. and Mrs. E. M. 87, 149
Howard, Earl Winfield 87-88
Howerton, James 32-33
Immaculate Heart of Mary Parish 59
Jeffries, William 88
Johnson, Quill 88-89
Jones, Charles Calvin 88
Jones, Floyd C. 89
Jones, G. B. and Michealina 89-90
Jones, James Marion 50
Jones, Jeremiah Denman 90
Jones, Michael Joe (Mike) 90, 150
Jordan, Sam H. 90-91
Karnes, Blanche Crawford 91
Kemper, John 91-92
Kimball, Rev. James Floyd 92-93
King, J. J. and Mabel 33-34, 143
Klepek, Albert and Lucille 34, 145
Lee, Charles Edward 93
Lester, Owen E. 50-51
Lindholm, John Nelson 93-94
Lindholm, Mady 94
Lippard, David 94-95
Little, Brice 95
Littleton, Sam and Ora 51
Long, R. O. 95-96
Lowe, Judge Marcellus 96
Lyons, Arthur E. 96
Malone, John J. 97, 148

Indexing speeds up searches

```
Searchstring: CATGAGACATGAA

    Query1: GAGA
    Query2: CATG
    Query3: ATGT
```

1) Decide on a word length *k*, e.g., k=3
2) Build hash table from search string, storing every word occurring in S together with its start position.
3) Process query and search for each word occurring in Q1 whether it is in the hash table.
4) Repeat for Q2.



Kmer	Position
CAT	1,8
ATG	2,9
TGA	3,10
GAG	4
AGA	5
GAC	6
ACA	7
GAA	11

Q1

Kmer	Position
GAG	1
AGA	2

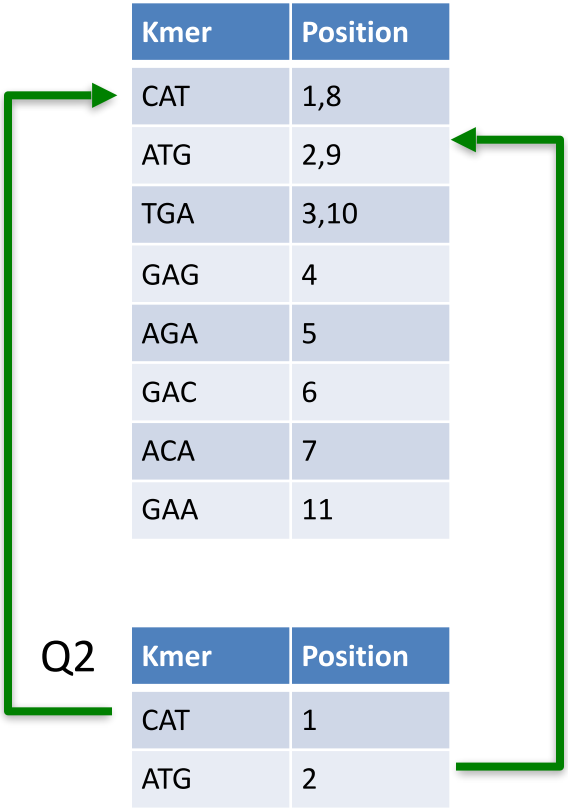
Two lookups are sufficient to find *Q1* in *S*

Main differences between mapping approaches

1) How does the mapper decide when the query maps multiple times to the sequence?

Mapping position of Q2: 1-4 **AND** 8-11

Relevant e.g. for RNA seq quantifying gene expression and for Chip-Seq as you must not consider a read more than once.



Kmer	Position
CAT	1,8
ATG	2,9
TGA	3,10
GAG	4
AGA	5
GAC	6
ACA	7
GAA	11

Q2

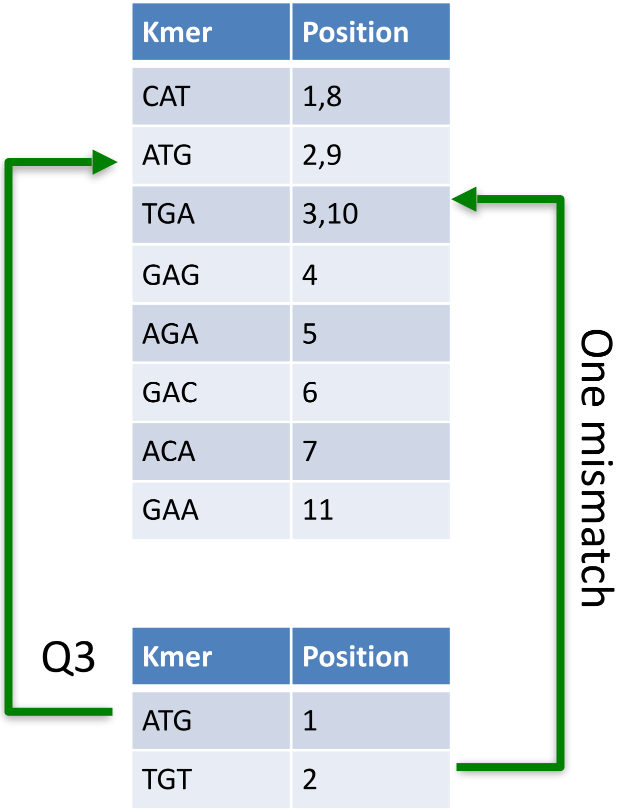
Kmer	Position
CAT	1
ATG	2

Main differences between mapping approaches

2) How does the mapper deal with queries that ‘almost’ match the reference?

Q3 matches the reference with one mismatch

Relevant for sensitivity and specificity of the mapping. Allowing more mismatches increases sensitivity (consider sequencing error and genetic diversity) but decreases specificity (more false positives).



Kmer	Position
CAT	1,8
ATG	2,9
TGA	3,10
GAG	4
AGA	5
GAC	6
ACA	7
GAA	11

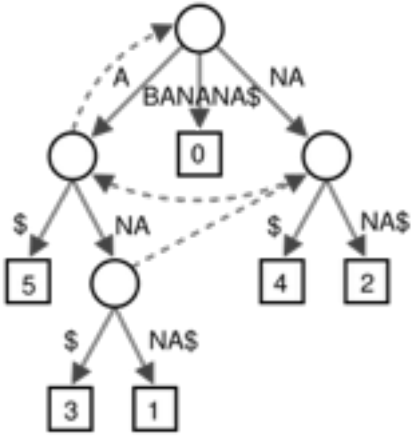
Kmer	Position
ATG	1
TGT	2

The 2nd lookup indicates that *Q3* is ‘almost’ in *S*

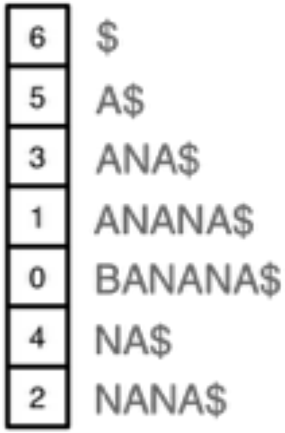
Main differences between mapping approaches

3) What kind of index does the mapper use?

Relevant for speed and memory footprint of the mapper

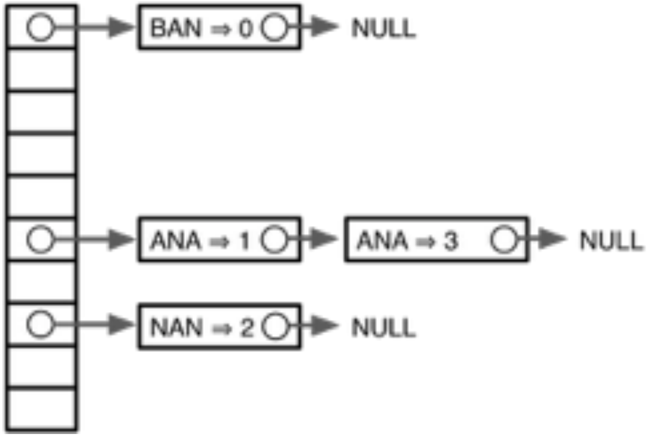


Suffix tree



6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

Suffix array



Seed hash tables

Many variants, incl. spaced seeds

$BANANA
A$BANAN
ANA$BAN
ANANA$B
BANANA$
NA$BANA
NANA$BA

Burrows Wheeler Transformation

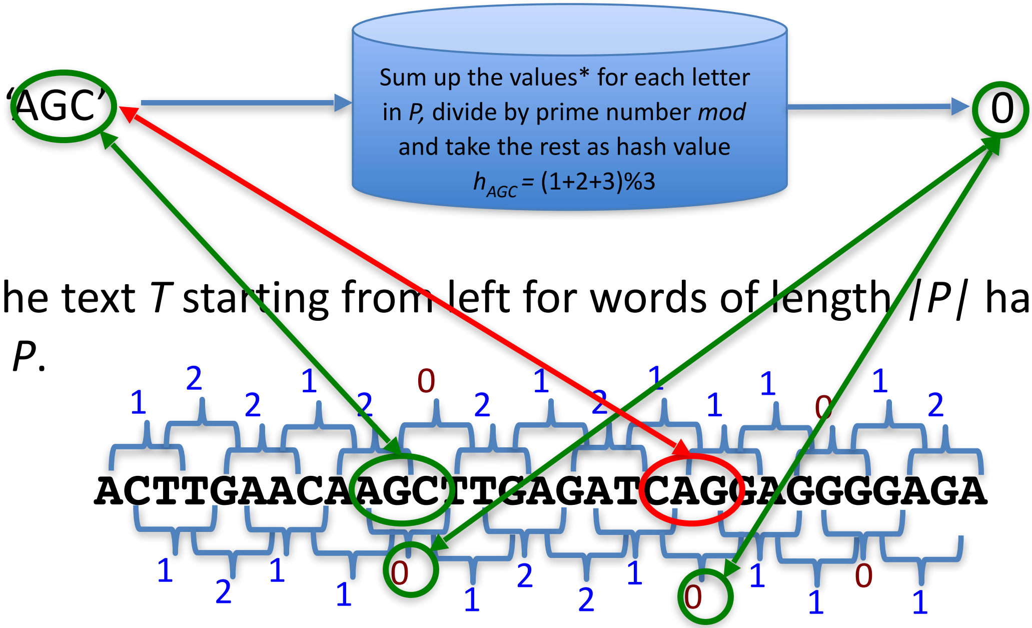
The “traditional way”: Hash tables

- Used by MAQ, Eland, SOAP, SHRiMP, ZOOM, partially by Mosaik

The use of ‘hashing’ in exact pattern search (Rabin-Karp; O(n+m))

Approach:

1) Use a ‘hash-function’ to transform pattern *P* into a numerical hash value h_P.

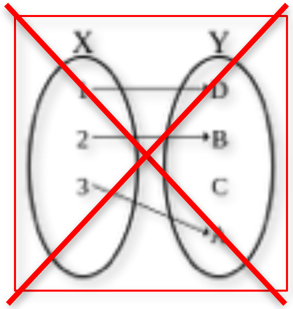


‘AGC’

Sum up the values* for each letter in *P*, divide by prime number *mod* and take the rest as hash value $h_{AGC} = (1+2+3)\%3$

0

1) Search the text *T* starting from left for words of length *|P|* having the same hash value as *P*.



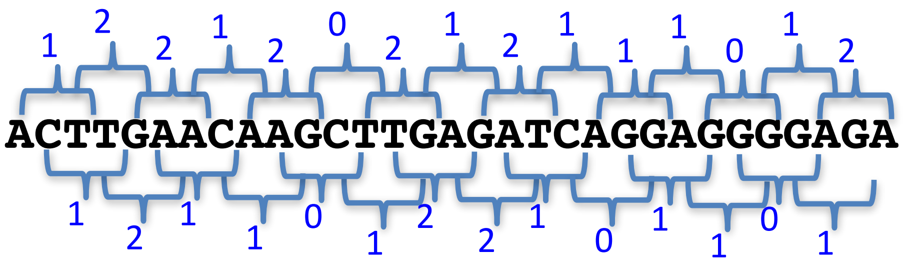
1) Given a word *K* with $h_k = h_P$ was found, perform an exact string comparison to verify that *K* == *P.* (Note, the projection of words with length *|P|* in the space of hash values is **not** injective (linkseindeutig!).

*JUST AN EXAMPLE: *v*(A)=1, *v*(C)=2, *v*(G)=3, *v*(T)=4; *mod*=3

‘hashing’ in combination with hash tables help to reduce the average time complexity of the pattern search to O(1), i.e. constant in time*

Idea: Speed up pattern search by creating look-up tables storing the hash values

1) Search the text *T* starting from left for words of length *k* and compute their hash value h_k



1) Store the hash values together with the starting point of the corresponding words in a hash table. Note, a hash table is nothing but a special way of indexing your data, just like a phone book. This will provide direct access to your potential matches in the pattern search once you know the hash value of *P*.

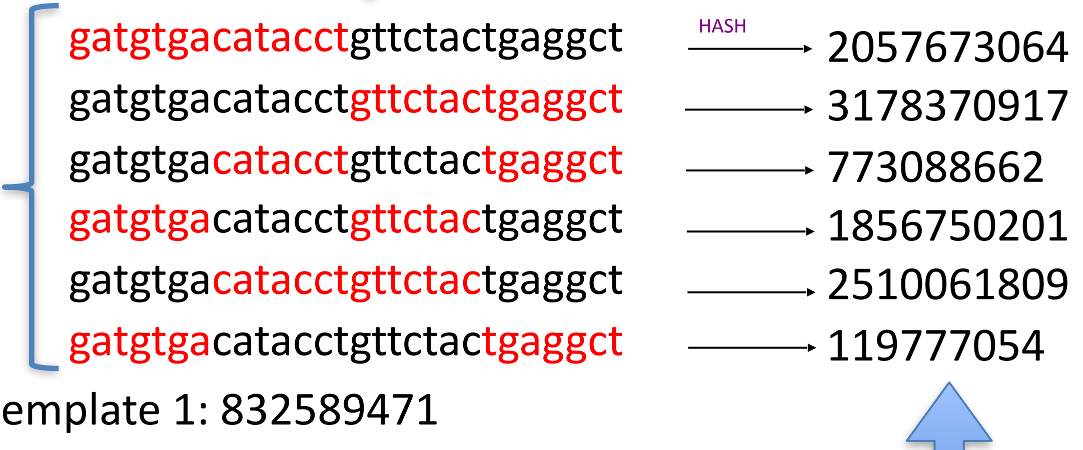
Hash value	Position in string
0	10,11,20,25,26
1	1,2,6,7,8,12,15,18,19,21,22,23,24,27,28
2	3,4,5,9,13,14,16,17,29

*note that this ignores the time and space you need to populate your hash table!

MAQ* uses **spaced seeds** for mapping

Example: read1 - gatgtgacatacctgttctactgaggct

Build **six** hash tables (templates) for the reads (only first 28 bp are considered) **only** from the colored nucleotides



6 'Templates'

	HASH
gatgtgacatacctgttctactgaggct	2057673064
gatgtgacatacctgttctactgaggct	3178370917
gatgtgacatacctgttctactgaggct	773088662
gatgtgacatacctgttctactgaggct	1856750201
gatgtgacatacctgttctactgaggct	2510061809
gatgtgacatacctgttctactgaggct	119777054

hash value for template 1: 832589471

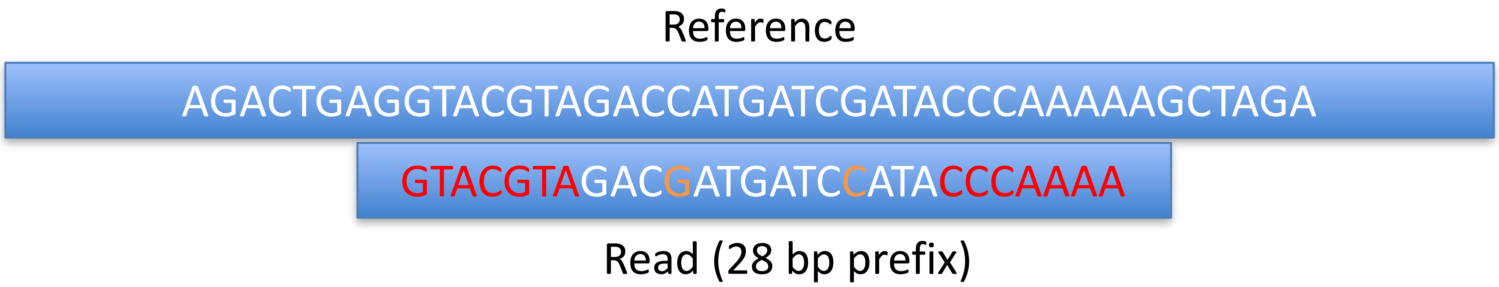
GENOME

```
ttcagttatccagaaaatgctattttcccaaaatgaaatctaaaatagtaactcaagtg
aaacattgtcaggtgtgtaaggaaggaaaatatgaagggctacccacaaacccacaaac
aacggaaactccaattcctaagtttccgggacacactattcatatagatataatttcta
cagacaaaaacgggtacttacggcaattcacaaattttcaaaacttgcgaaagcaaaaa
taaaaaaaattcaaaatcaatagaagacataagaaaacctttacatgacatcttatttt
attttggagtaccaaaatacgttgtaatggatatagaaaaatcctttaattccgcaaca
acagcctttatgatgaaagaccagctgggcatacaaattttcaaagcatccccttataa
aagttctgtaaacggacaaatagaacggtttcattctatcctcgctgaaattaaaagat
gtttaaaaactaaacaggtacaccgaacatttgaagaacagttcaattcagctgtctag
gaatataactacacaattcaccctgtatcaaaaatacaaaacaaaataacgcccttaaa
aatattttggcagaaatataaccactgatccaggaaaatatgacgaaagcagataagg
caacatcgaaaacctataatcaaaacaggcaacaggcttaaaaaccataacgaaaaaag
caataagatcaaagattatgagccaggacaaacagtttttataaagcaaatcacaaggc
caggttctaagctgtacaagaaagaaacattaaaagaaaacagacaaacatttgttatc
acagaagcaggaagatgtgacatacctgttctactgaaggct
```

compute hash values for spaced seeds in reference (on both strands and for one of the six templates) and perform lookups in hash tables of the reads

*Li et al *Genome Res. 2008. 18: 1851-1858*

Why does MAQ* use **spaced seeds** for mapping?



Reference

AGACTGAGGTACGTAGACCATGATCGATACCCAAAAAGCTAGA

GTACGTAGACGATGATCCATACCCAAAA

Read (28 bp prefix)

MAQ uses seed pairs as it allows per default up to 2 mismatches between seed and reference. As each 28 mere is represented by 4 non-overlapping seeds, we are guaranteed that we **always have at least 2 seeds** that must result in a perfect match to the reference.

MAQ: An overview

1. At the alignment stage, MAQ first searches for the ungapped match with lowest mismatch score, defined as the sum of qualities at mismatching bases.

2. MAQ only considers positions that have two or fewer mismatches in the first 28 bp (default parameters; speed-up).

3. Sequences that fail to reach a mismatch score threshold but whose mate pair is mapped are searched with a gapped alignment algorithm in the regions defined by the mate pair.

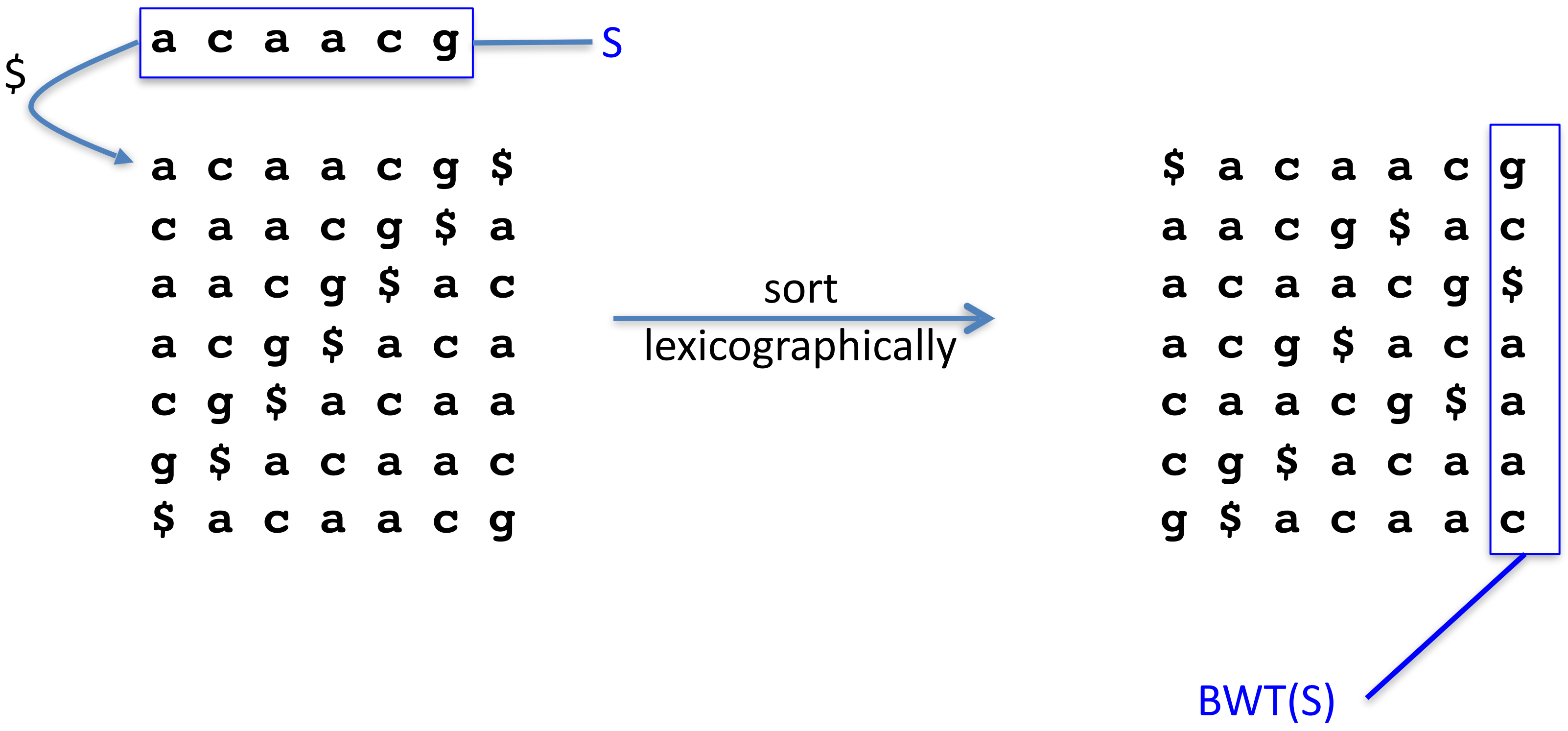
4. To evaluate the reliability of alignments, MAQ assigns each individual alignment a phred-scaled quality score (capped at 99), which measures the probability that the true alignment is not the one found by MAQ.

5. MAQ always reports a single alignment, and if a read can be aligned equally well to multiple positions, MAQ will randomly pick one position and give it a mapping quality zero. Because their mapping score is set to zero, reads that are mapped equally well to multiple positions will not contribute to variant calling.

The “new way”: Burrows-Wheeler Transform

- Invented by David Wheeler in 1983 (bell labs), pub. 1994
- Used in data compression (bzip2)
- Used in 2003 on the human genome to define exact word matches (originally for microarray probe design)
- First used for short read alignment by bowtie, now adopted by bwa (maq author) and SOAP2

The Burrows-Wheeler Transform



```
a c a a c g
```

S

$

```
a c a a c g $
c a a c g $ a
a a c g $ a c
a c g $ a c a
c g $ a c a a
g $ a c a a c
$ a c a a c g
```

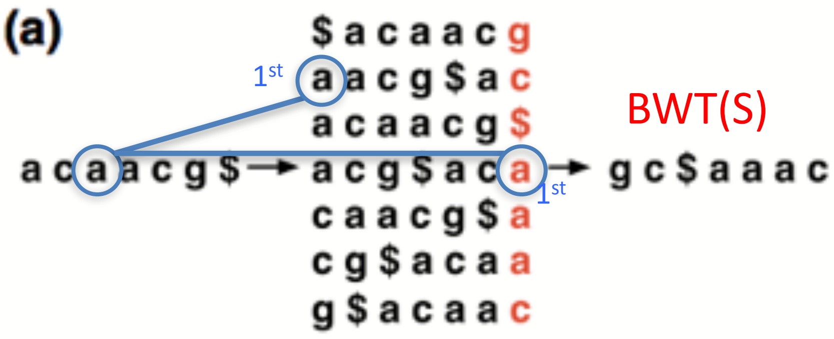
sort

lexicographically

```
$ a c a a c g
a a c g $ a c
a c a a c g $
a c g $ a c a
c a a c g $ a
c g $ a c a a
g $ a c a a c
```

BWT(S)

Burrows Wheeler Transform (BWT)

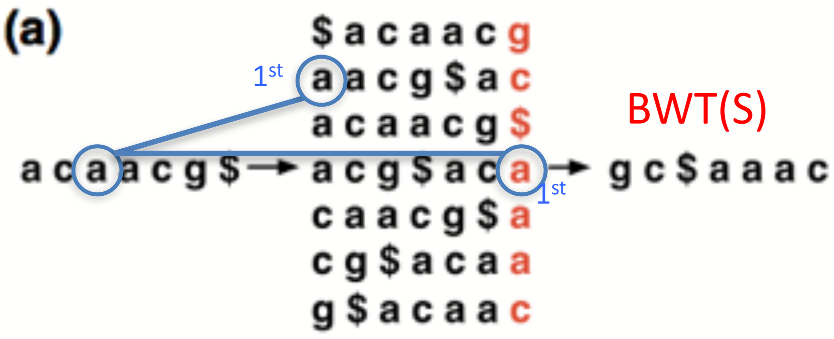


Generate matrix by

1. Appending a *$* to the end of the string *S* that should be indexed. *$* should have 2 properties
   1. it must not occur in the string
   2. it should be lexicographically smaller than any character in S
2. generate all cyclic permutations of S
3. sort the resulting matrix lexicographically (the line beginning with the $ is the first to occur in the matrix.

The matrix has the property of last first (LF) mapping: The ith occurrence of character X in the last column corresponds to the same text character as the ith occurrence of X in the first column

Burrows Wheeler Transform (BWT)

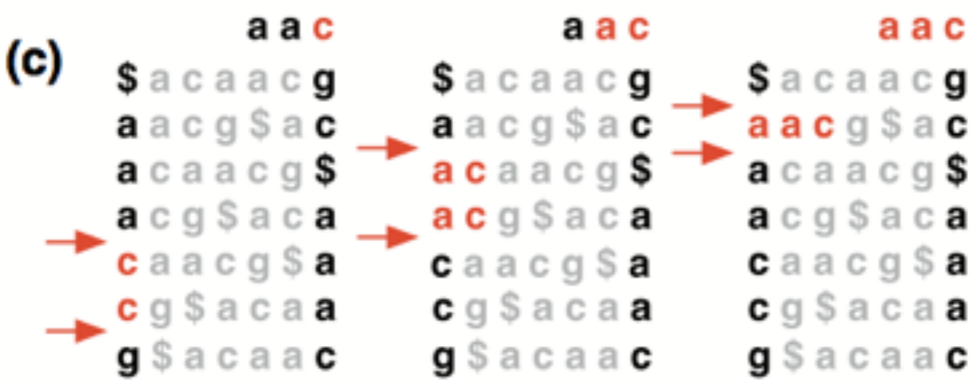


```
         $acaacg
         aacg$ac
         acaacg$
acaacg$→ acg$aca → gc$aaac
         caacg$a
         cg$acaa
         g$acaac
```

The matrix has the property of last first (LF) mapping: This can be used to reconstruct the original text from BWT(S) using the UNPERMUTE algorithm.

Burrows Wheeler Transform (BWT)

BWT(S)



(c)

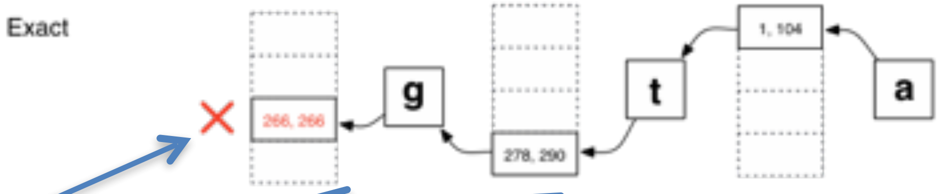
```
      aac        aac        aac
  $acaacg    $acaacg    $acaacg
  aacg$ac    aacg$ac    aacg$ac
  acaacg$    acaacg$    acaacg$
  acg$aca    acg$aca    acg$aca
  caacg$a    caacg$a    caacg$a
  cg$acaa    cg$acaa    cg$acaa
  g$acaac    g$acaac    g$acaac
```

The matrix has the property of last first (LF) mapping: This can be used to search for a text within BWT(S) using the EXACTMATCH algorithm.

Key aspects:

1) Matrix is sorted lexicographically. Thus, rows beginning with a given sequence appear **consecutively**.
2) EXACTMATCH algorithm calculates the range of matrix rows beginning with **successively longer** suffixes of the query.
3) At each step, the size of the **range either shrinks or remains the same**.
4) When the algorithm completes, rows beginning with S_0 (the entire query) correspond to **exact occurrences of the query** in the text.

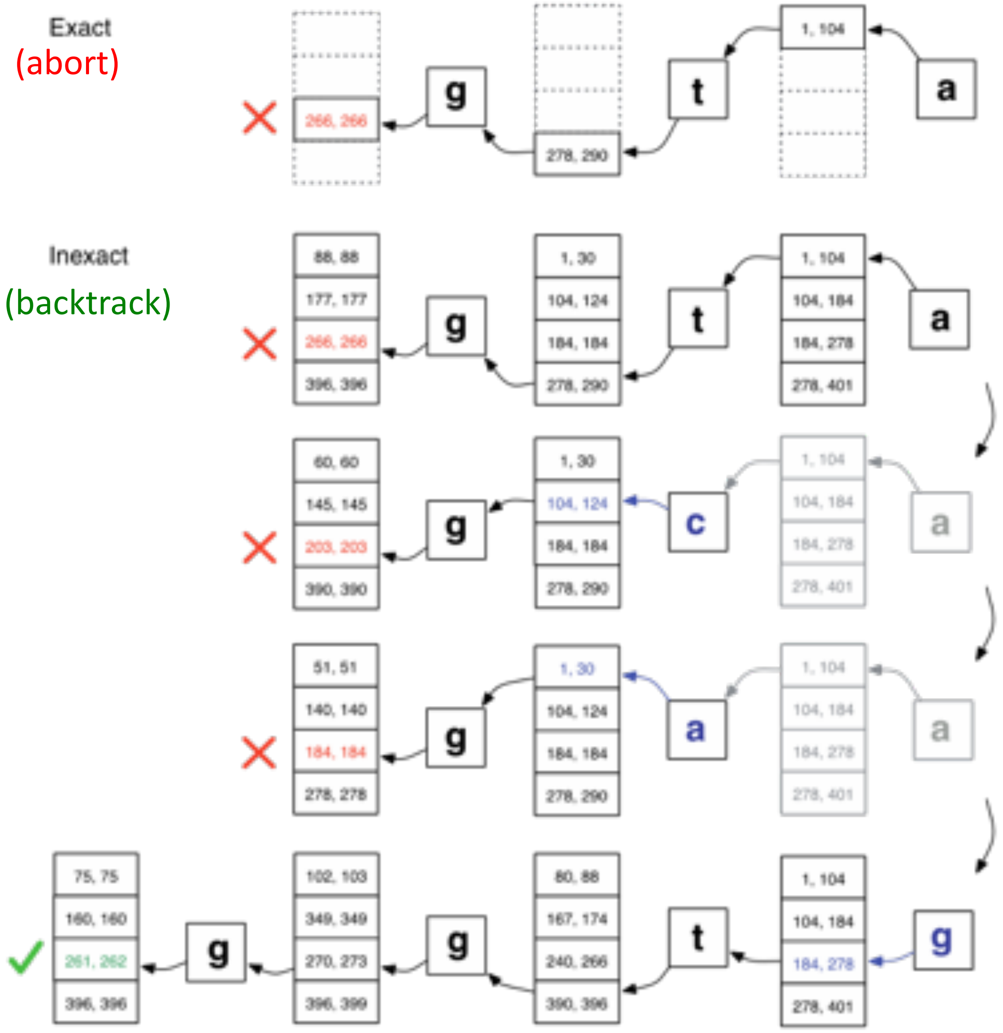
How to cope with mismatches?
Query GGTA



Empty range:
Abort or backtrack.

Ranges of the matrix rows beginning with the suffix observed to that point.

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



1. The search proceeds similarly to EXACTMATCH
2. If range becomes empty (suffix does not occur in text), the algorithm backtracks and selects an already-matched query position and substitutes a different base there. The EXACTMATCH algorithm resumes from this modified position.
3. The algorithm allows only substitutions that yield a modified suffix that occurs at least once in the text. If there are multiple candidate substitution positions, then the algorithm greedily selects a position with a minimal quality value.
4. Because search is greedy, the first valid alignment is not necessarily the best (Number of mismatches and quality). Bowtie has parameters to cope with this (--best or –all (all alignments).
5. Excessive Backtracking should be avoided. Note, we start from the low quality end...

Avoidance of excessive backtracking (assuming 1 mismatch)

Problem: The aligner spends most of its effort fruitlessly backtracking to positions close to the 3' end of the query (error prone).

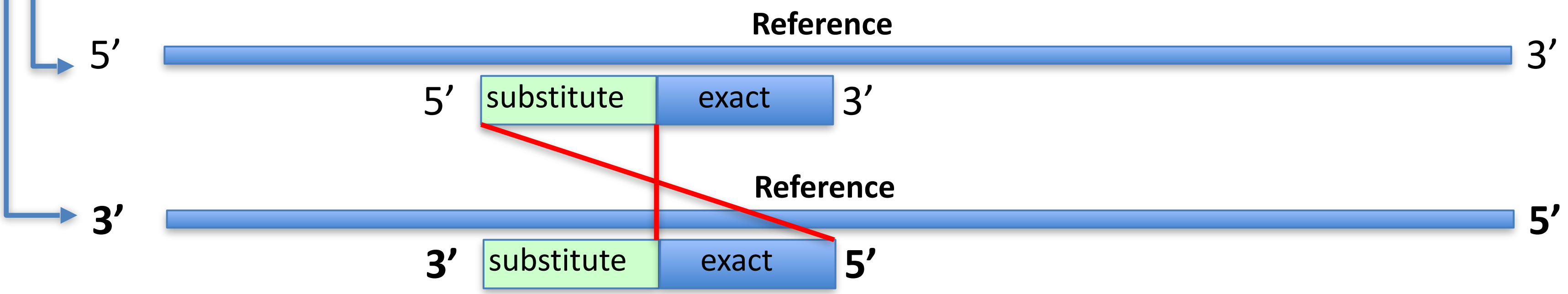
Solution Part1: double indexing (similar to MAQ), using two indices for the genome

- Index 1: BWT of the original genome
- Index 2: BWT of the genome with reversed character order (**not reverse complemented!**)

Solution Part2: The aligner is invoked twice

- **First round:** Index 1 is used, and the aligner is started with original read with the constraint that it must not substitute a position in the query's right half (3' end).
- **Second round:** Index 2 is used, and the aligner is started with the **reversed read**, again with the constraint that it must not substitute a position in the reversed query's right half (originally and still the 5' end).

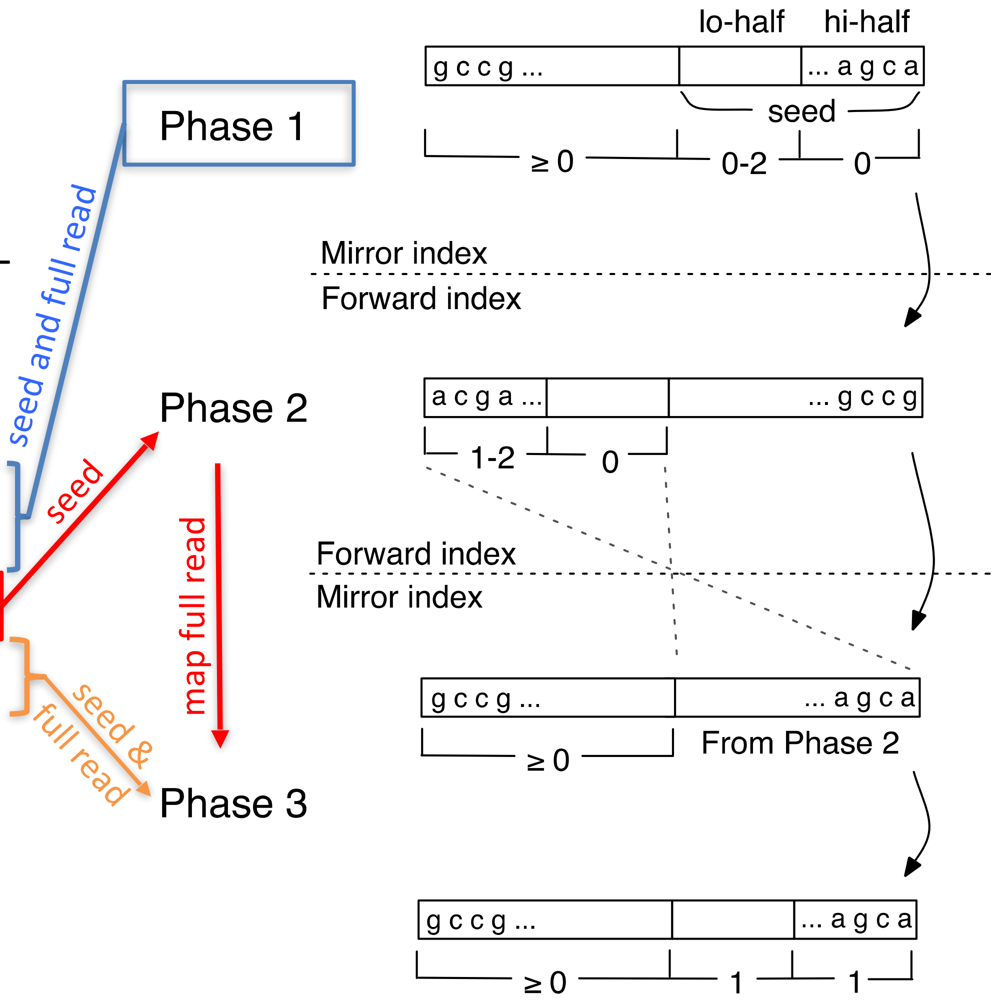
Solution Part3: set a hard upper limit of backtracks to be performed.



The three phases of Bowtie

In the case of 2 (or more mismatches):

- Bowtie uses the first 28 bp as seed
- The seed is split into a high quality 5’ half (hi-half) and a low quality 3’ half (lo-half)
- For up to 2 mismatches we have four scenarios:
    1. no mismatches in seed
    2. 1-2 mismatches only in the lo-half
    3. 1-2 mismatches only in the hi-half
    4. 1 mismatch each in hi- and low-half
- Any number of mismatches can occur in non-seed part (subject to other thresholds).



SAM/BAM format* & SAMtools

Header section

```
@HD   VN:1.0
@SQ   SN:chr20 AS:HG18 LN:62435964
@RG   ID:L1 PU:SC_1_10 LB:SC_1 SM:NA12891
@RG   ID:L2 PU:SC_2_12 LB:SC_2 SM:NA12891
```

position of alignment

Alignment section

Query Name

Ref sequence

query sequence (same strand as ref)

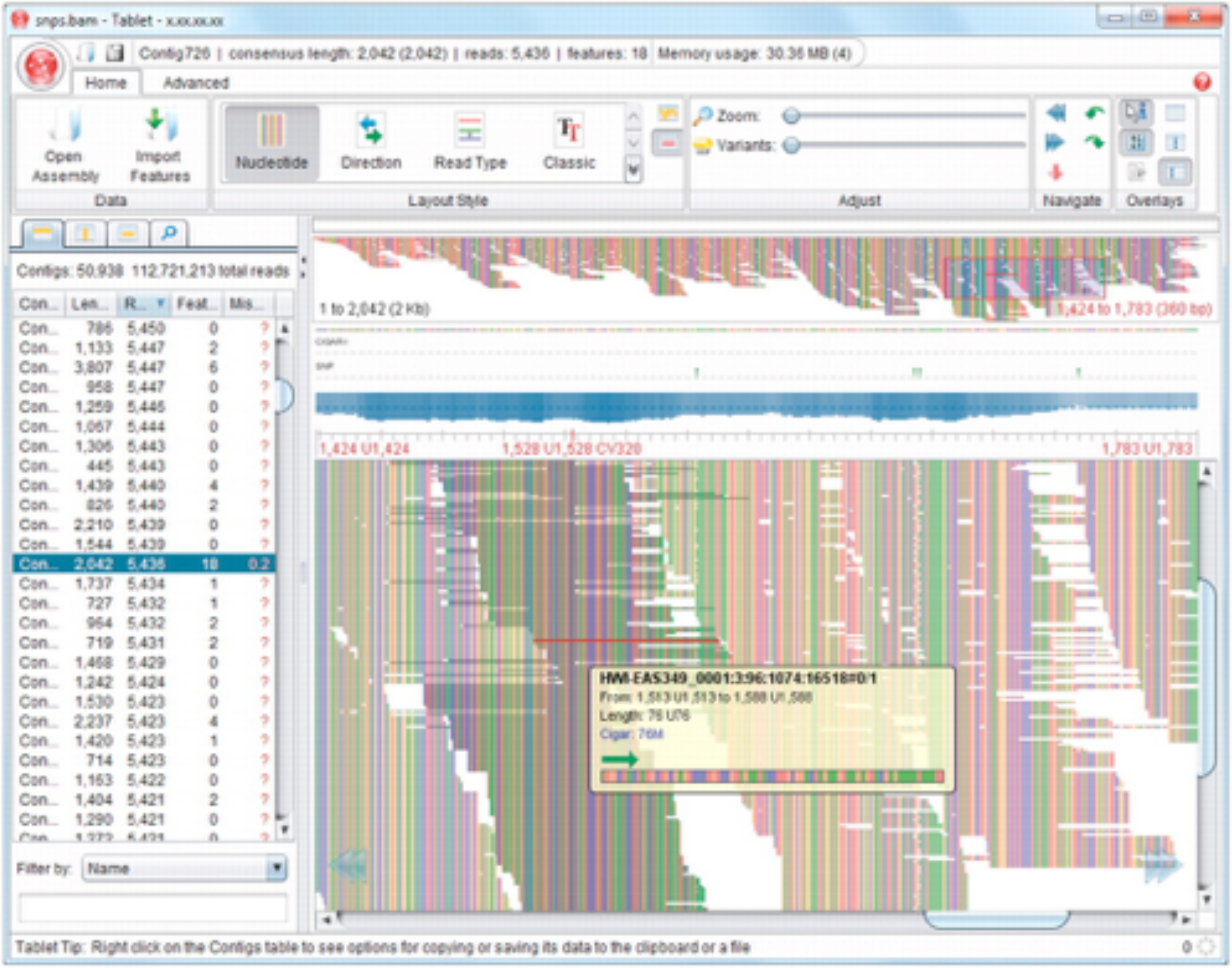
query quality

```
V00-HWI-EAS132:3:38:959:2035#0     147     chr1     28     255     36M     =     79       0     GATCTGATGGCAGAAAACCCCTCTCAGTCCGTCGTG     aaX`[\`Y^Y^]ZX``\EV_BBBBBBBBBBBBBBBB     NM:i:1
V00-HWI-EAS132:4:99:122:772#0      177     chr1     32     255     36M     =     9127     0     AAAGGATCTGATGGCAGAAAACCCCTCTCAGTCCGT     aaaaaa\OWaI_\WL\aa`Xa^]\ZUaa[XWT\^XR     NM:i:1
V00-HWI-EAS132:4:44:473:970#0      25      chr1     40     255     36M     *     0        0     GTCGTGGTGAAGGATCTGATGGCAGAAAACACCTCT     __YaZ`W[aZNUZ[U[_TL[KVVX^QURUTDRVZBB     NM:i:2
V00-HWI-EAS132:4:29:113:1934#0     99      chr1     44     255     36M     =     107      0     GGGTTTTCTGCCATCAGATCCTTTACCACGACAGAC     aaaQaa__``]\\_^``^a^`a`_^^^_XQ[ZS\XX     NM:i:1
```

- Sequence Alignment/Map & Binary Alignment/Map (Heng Li, et al)
- SAMtools - C or Java
  - convert formats
  - sort/merge alignments
  - generate per-position information
  - call SNPs/indel variants
  - show alignments in a text viewer
  - http://bioinformatics.oxfordjournals.org/cgi/reprint/btp352v1

*for details on the format see http://samtools.sourceforge.net/SAM1.pdf

Data visualization using Tablet



Contig726 | consensus length: 2,042 (2,042) | reads: 5,436 | features: 18 Memory usage: 30.36 MB (4)

Contigs: 50,938 112,721,213 total reads

Con...	Len...	R...	Feat...	Mis...
Con...	786	5,450	0	?
Con...	1,133	5,447	2	?
Con...	3,807	5,447	6	?
Con...	958	5,447	0	?
Con...	1,259	5,445	0	?
Con...	1,067	5,444	0	?
Con...	1,306	5,443	0	?
Con...	445	5,443	0	?
Con...	1,439	5,440	4	?
Con...	826	5,440	2	?
Con...	2,210	5,439	0	?
Con...	1,544	5,439	0	?
Con...	2,042	5,436	18	0.2
Con...	1,737	5,434	1	?
Con...	727	5,432	1	?
Con...	964	5,432	2	?
Con...	719	5,431	2	?
Con...	1,468	5,429	0	?
Con...	1,242	5,424	0	?
Con...	1,530	5,423	0	?
Con...	2,237	5,423	4	?
Con...	1,420	5,423	1	?
Con...	714	5,423	0	?
Con...	1,163	5,422	0	?
Con...	1,404	5,421	2	?
Con...	1,290	5,421	0	?
Con...	1,272	5,421	0	?

HWI-EAS349_0001:3:96:1074:16518#0/1
From: 1,513 U1,513 to 1,588 U1,588
Length: 76 U76
Cigar: 76M

Tablet Tip: Right click on the Contigs table to see options for copying or saving its data to the clipboard or a file

Milne et al. Brief Bioinform (2013) 14 (2): 193-202.

Mapping: Watch out for the following…

- Up to 2 mismatches in first 28 bases reported by default (Maq, Bowtie, BWA). Maq additionally reports some 3-mismatches.
- If >=1 exact matches exist for a read, Bowtie will report one of them. If the best match is inexact, bowtie will not always return the highest quality alignment (unless run with --best).
- Bowtie, BWA, and SOAP2 all randomly place repeat reads.
- Although in theory BWA works with arbitrarily long reads,its performance is degraded on long reads (probably with any BWT aligner). Consider using Bowtie2.
- Non-ACTG bases in the genome are converted to random bases in BWA
- How does the mapper deal with paired end reads? Will it report a mapping success only when both reads map in the correct distance (specify insert size) and orientation?