Sequence Mapping/Alignment

BCB 504: Applied Bioinformatics

Matt Settles

University of Idaho
Bioinformatics and Computational Biology Program

April 4, 2012

Outline

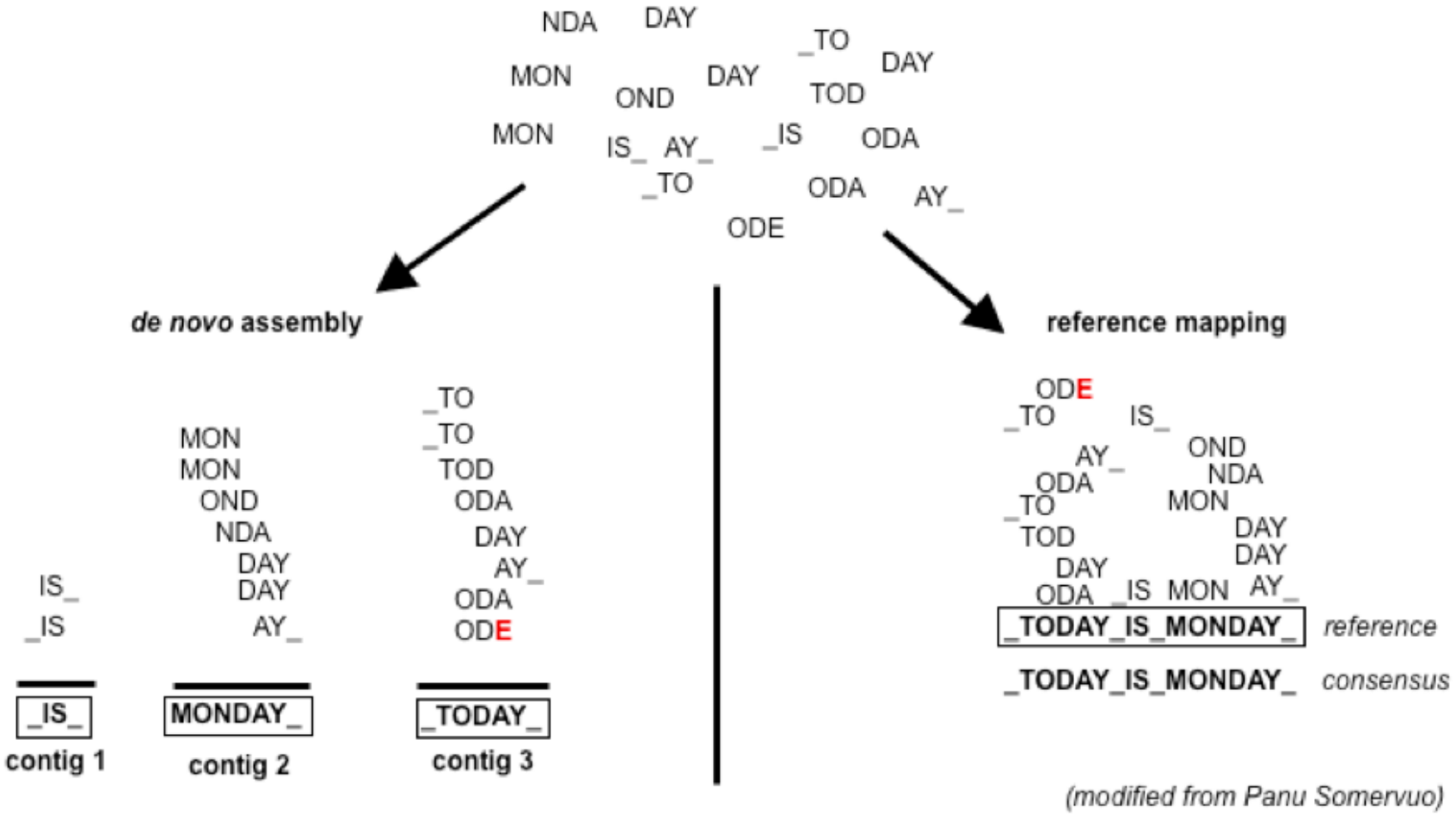
Mapping/Alignment

Given sequence data,

Assembly seeks to put together the puzzle without knowing what the picture is

Mapping tries to put together the puzzle pieces directly onto an image of the picture

In mapping the question is more, given a small chunk of sequence, where in the genome did this piece most likely come from.


NDA
DAY
_TO
DAY
MON
DAY
OND
TOD
MON
IS_
AY_
_IS
ODA
_TO
ODA
AY_
ODE
de novo assembly
reference mapping
IS_
_IS
MON
MON
OND
NDA
DAY
DAY
AY_
_TO
_TO
TOD
ODA
DAY
AY_
ODA
ODE
_IS_
MONDAY_
TODAY
contig 1
contig 2
contig 3
ODE
_TO
IS_
AY_
OND
ODA
NDA
_TO
MON
TOD
DAY
DAY
DAY
ODA
_IS
MON
AY_
_TODAY_IS_MONDAY_
reference
_TODAY_IS_MONDAY_
consensus
(modified from Panu Somervuo)

Basic Local Alignment Search Tool (BLAST)

Some say the first bioinformatics tool, developed at NIH and published in 1990.
Problem:

- Exact algorithms like Smith-Waterman and Needleman-Wunsch (dynamic programming) are slow, when the search space becomes large.
- With the advent of automated DNA sequencing technology, the database of possible matches was becoming increasingly larger.

the BLAST algorithm emphasizes speed over sensitivity, and does not guarantee an optimal alignment.
BLAST is a few to many algorithm - performs gapped alignment

BLAST Like Alignment Tool (BLAT)

Blat (Jim Kent, UCSC, 2002) was designed to solve the problem of performing comparisons between large genomes and was one of the first algorithms to efficiently search many query sequences against a large database (a genome). Blat also performs a gapped-alignment for searching RNA sequences against a genome and handling splice junctions.

gapped-alignment alignment allowing for insertions and deletions greater than a few base pairs. Gapped alignment are less efficient, but more accurate.

BLAT is a many to many algorithm - performs gapped alignments

Improving Algorithms

Many additional algorithms have been developed since BLAST and BLAT, mainly improving on either speed or accuracy, or both.

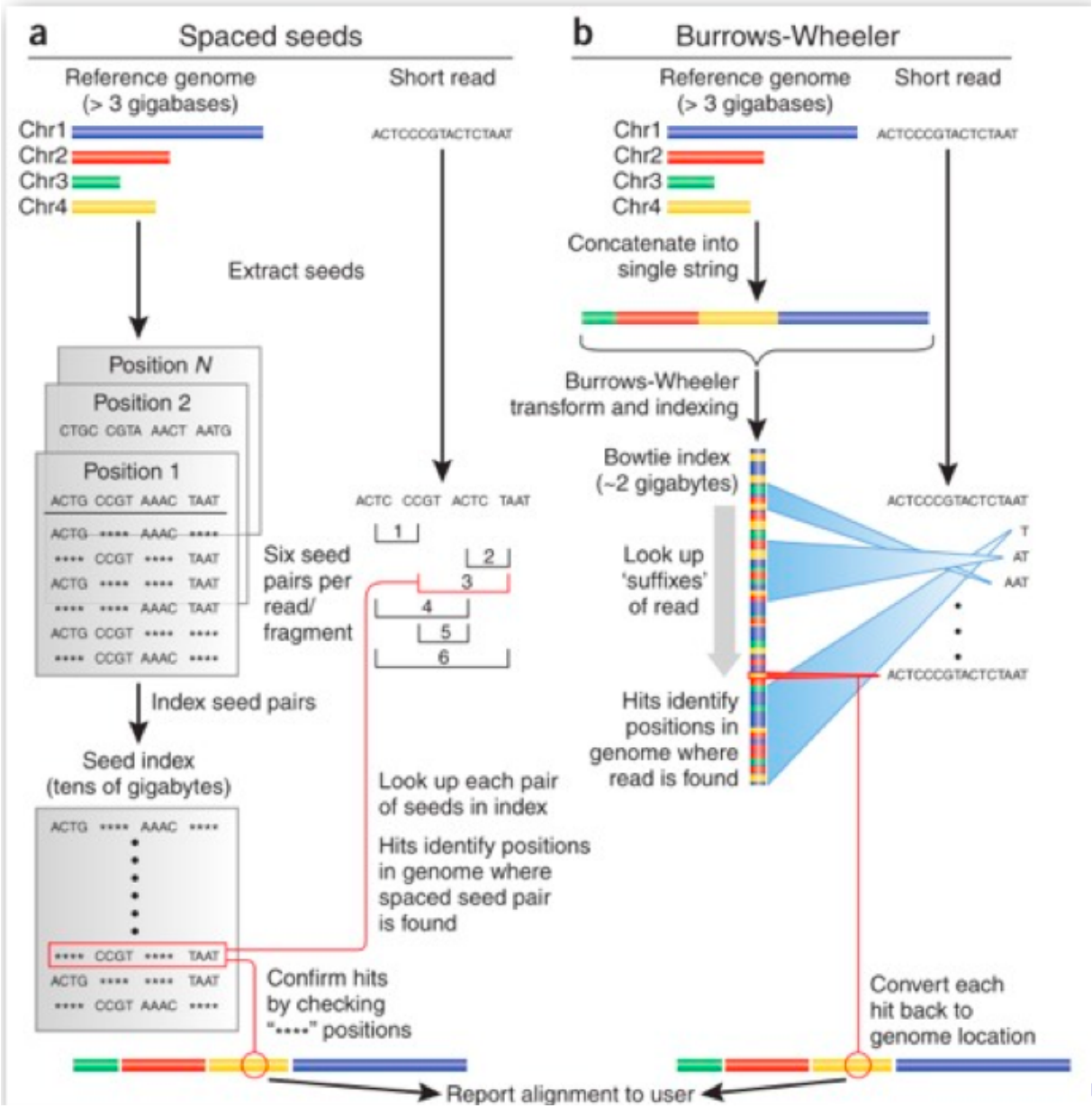
and then came Illumina data -
New Problem:

- We still have a large search space (aka genome)
- Very small pieces, many possible close matches
- Millions or tens of millions of query sequences

Types

Hash based First generation of alignment algorithms relied on hashes (Eland [Illumina], RMAP, MAQ, SHRiMP, SOAP)

Burrows-Wheeler Second generation algorithms with a reduced memory footprint (BWA, SOAP2, Bowtie)


a
Spaced seeds
Reference genome
(> 3 gigabases)
Short read
Chr1
Chr2
Chr3
Chr4
ACTCCCGTACTCTAAT
Extract seeds
Position N
Position 2
CTGC CGTA AACT AATG
Position 1
ACTG CCGT AAAC TAAT
ACTC CCGT ACTC TAAT
1
2
3
4
5
6
Six seed pairs per read/ fragment
Index seed pairs
Seed index
(tens of gigabytes)
Look up each pair of seeds in index
Hits identify positions in genome where spaced seed pair is found
Confirm hits by checking "****" positions
b
Burrows-Wheeler
Reference genome
(> 3 gigabases)
Short read
Concatenate into single string
Burrows-Wheeler transform and indexing
Bowtie index
(~2 gigabytes)
Look up 'suffixes' of read
Hits identify positions in genome where read is found
Convert each hit back to genome location
Report alignment to user

hash based example: MAQ

- Index reference genome (or sequence reads) => creates hash index
  - Big file: >50GB
  - takes a long time (hours or overnight), but only need to do once
- Divide each read into segments (seeds) and look up in table
  - Search stage finds regions in the genome that can potentially be homologous to the read.
  - Alignment stage verifies these regions to check if they are indeed homologous. More computationally intensive

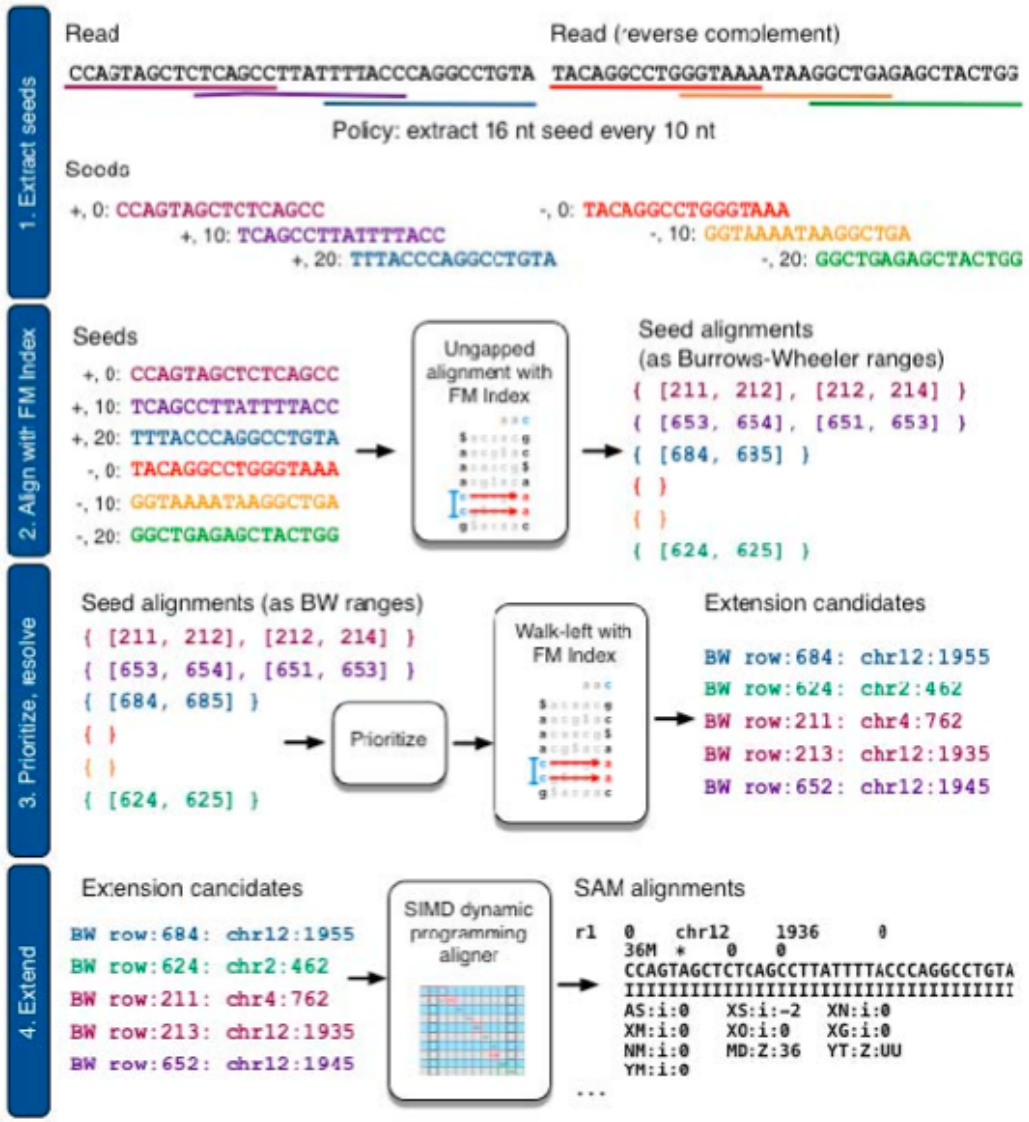
burrows-wheeler example: Bowtie2

Used in data compression (e.g. bzip) => index: much smaller than hash-based index (<2GB)

- Alignment speed: 30x faster than MAQ

Steps:

- Create BWT index of genome
- Align read 1 character at a time to BWT-transformed genome


1. Extract seeds
Read
CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA
Read (reverse complement)
TACAGGCCTGGGTAAAATAAGGCTGAGAGCTACTGG
Policy: extract 16 nt seed every 10 nt
Seeds
+, 0: CCAGTAGCTCTCAGCC
+, 10: TCAGCCTTATTTTACC
+, 20: TTTACCCAGGCCTGTA
-, 0: TACAGGCCTGGGTAAA
-, 10: GGTAAAATAAGGCTGA
-, 20: GGCTGAGAGCTACTGG
2. Align with FM Index
Seeds
+, 0: CCAGTAGCTCTCAGCC
+, 10: TCAGCCTTATTTTACC
+, 20: TTTACCCAGGCCTGTA
-, 0: TACAGGCCTGGGTAAA
-, 10: GGTAAAATAAGGCTGA
-, 20: GGCTGAGAGCTACTGG
Ungapped alignment with FM Index
Seed alignments (as Burrows-Wheeler ranges)
{ [211, 212], [212, 214] }
{ [653, 654], [651, 653] }
{ [684, 685] }
{ }
{ }
{ [624, 625] }
3. Prioritize, resolve
Seed alignments (as BW ranges)
{ [211, 212], [212, 214] }
{ [653, 654], [651, 653] }
{ [684, 685] }
{ }
{ }
{ [624, 625] }
Prioritize
Walk-left with FM Index
Extension candidates
BW row:684: chr12:1955
BW row:624: chr2:462
BW row:211: chr4:762
BW row:213: chr12:1935
BW row:652: chr12:1945
4. Extend
Extension candidates
BW row:684: chr12:1955
BW row:624: chr2:462
BW row:211: chr4:762
BW row:213: chr12:1935
BW row:652: chr12:1945
SIMD dynamic programming aligner
SAM alignments
r1 0 chr12 1936 0
36M * 0 0
CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
AS:i:0 XS:i:-2 XN:i:0
XM:i:0 XO:i:0 XG:i:0
NM:i:0 MD:Z:36 YT:Z:UU
YM:i:0
...

considerations

- placing reads in regions that do not exist in the reference genome
- sequencing errors and variations: alignment between read and true source in genome may have more differences than alignment with some other copy of repeat.
- What if many nucleotide differences with closest fully sequenced genome? (3% is a common alignment capability)
- placing reads in repetitive regions: Some algorithms only return 1 mapping; If multiple: map quality = 0
- algorithms that use paired-end information => might prefer correct distance over correct alignment.

SAM/BAM format

SAM (Sequence Alignment/Map) format = unified format for storing read alignments to a reference genome

BAM = binary version of SAM for fast querying

```
7172283    163    chr9    139389330    60    90M    =    139389482    242    TAGGAGG...    EHHHHHH...
7705896     83    chr9    139389513    60    90M    =    139389512    -91    GCTGGGG...    EBCHHFC...
7705896    163    chr9    139389512    60    90M    =    139389513     91    AGCTGGG...    HHHHHHH...
```

1	QNAME	query template name
2	FLAG	bitwise flag
3	RNAME	reference sequence name
4	POS	1-based leftmost mapping position
5	MAPQ	mapping quality
6	CIGAR	CIGAR string
7	RNEXT	reference name of mate
8	PNEXT	position of mate
9	TLEN	observed template length
10	SEQ	sequence
11	QUAL	ASCII of Phred-scaled base quality