Algorithms for analysis of deep sequencing data

Main Source:
4761 Computational Genomics
Itsik Pe'er, Dept of CS, Columbia Univ, Spring 2010

Also:
M. Brudno: Introduction to High Throughput Sequencing
Outline

• Brief intro to deep sequencing technologies
  - The numbers
  - Technology
• Mapping
  - basic techniques
  - MAQ algorithm
  - Bowtie algorithm
• Assembly
  - de Bruijn graph algorithms
History of DNA Sequencing

- 1870: Miescher: Discovers DNA
- 1940: Avery: Proposes DNA as ‘Genetic Material’
- 1953: Holley: Sequences Yeast tRNA\textsuperscript{Ala}
- 1965: Wu: Sequences λ Cohesive End DNA
- 1970: Sanger: Dideoxy Chain Termination
- 1977: Gilbert: Chemical Degradation
- 1980: Messing: M13 Cloning
- 1990:
  - Cycle Sequencing
  - Improved Sequencing Enzymes
  - Improved Fluorescent Detection Schemes
- 2002:
  - Next Generation Sequencing
  - Improved enzymes and chemistry
  - New image processing
- 2009:

Efficiency (bp/person/year):
- 1870: 1 bp/person/year
- 1940: 15 bp/person/year
- 1953: 150 bp/person/year
- 1965: 1,500 bp/person/year
- 1970: 15,000 bp/person/year
- 1980: 25,000 bp/person/year
- 1986: 50,000 bp/person/year
- 1990: 200,000 bp/person/year
- 2002: 50,000,000 bp/person/year
- 2009: 100,000,000,000 bp/person/year

Adapted from M Brudno; Adapted from E Green, NIH; Adapted from Messing & Llaca, PNAS (1998)
The race for quick and cheap sequencing

• 2003: Human genome project completed – cost $3 billion

• Challenges:
  • 2003: J. Craig Venter foundation announced a prize of $500,000 to be awarded to a group that could sequence a human genome for $1000.
  • 2005–2006: • NIH provides ~$32M in grants for developing new sequencing technologies. • X Prize foundation creates the Archon X Prize for Genomics: $10M to a group that builds a device and uses it to sequence 100 human genomes within 10 days with an accuracy of <1 error in every 100K bases, for no more than $10,000 per genome.

• Numerous new sequencing technologies were developed in response to these challenges: 454 Life Sciences, Solexa (Illumina), SOLiD, Helicos, Pacific Biosciences, Ion Torrent...

• Sep.–Oct. 2010: • Illumina offers human genome sequencing for $10,000.
  • 23andme offers genotyping for $100.
600B bases per day!!

January 2014
How do we read DNA?
How do we read DNA?

- We replicate it
How do we read DNA?

- We replicate it
- We shred it
Reading short DNA

• Use replication machinery with colored bases
• Take pictures of massively parallel reaction
  10 million reads of 30 bp per day & $1000
Reading short DNA

• Use replication machinery with colored bases
• Take pictures of massively parallel reaction

100 million reads of 100 bp per day & $1000
Example

The genome is:
TTATGGTCGGTGAGTGTGACTGGTGTTGTTGCTAA

The reads are:
GGTCGGTGAG
TGAGTGTGAC
TGGTGGTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAAA
Example

The genome is:

TTATGGTCGGTGAGTGTGACTGGTGTTGTTGCTAA

GGTCGGTGAG

The reads are:

GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAA
Example

The genome is:
```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
  ||||||||||||
  TGAGTGTGAC
```

The reads are:
```
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAAA
```
Example

The genome is:
TTATGGTCCGGTGAGTGTGAC

The reads are:
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAA
Example

The genome is:

TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA

The reads are:

GGTCGGTGAG
TGAGTGTTGAC
TGTTGTTGTC
TGACTGGGGTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAAAA
Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTTGTCCTAA
```

The reads are:

```
GGTCGGTGAG
TGAGTGTGAC
TGAGTGTGAC
TGACTGGTTT
AATGGTCG
GGTCGGTGAG
TGACTGGTTT
AATGGTCG
AAAAAAA
```
The genome is:

```
TTATGGTCGCGTGAGTGTGACTGGTGTTGTCTAA
GGTCGCGTGAG
TGAGTGACTGGTGTTGTCTAA
```

```
AATGGTCGGT
```

```
GAGTGACT
```

```
AAAAAAAAAAAA
```
NGS algorithmics

1. Mapping and the MAQ algorithm
Short read Mapping Problem

• Align reads to genome

• Input:
  Many reads: $l$-long strings $S_1, \ldots, S_m$
  Approximate reference genome: string $R$

• Output:
  positions $x_1, \ldots, x_m$ in $R$ where the reads match

• Complications:
  - Errors, Differences, Repetitive regions

$R$: the genome
$m$ reads of length $l$ each
Parameters

• Assume:
  - $m$: $10^7 - 10^8$ reads of length $l$: 100-200 per run
  - Possibly paired-ends
  - Genome $R$ of length $3 \times 10^9$
  - 2-8 Gb memory
  - May also want to analyze multiple runs together, resulting in $m > 10^9$

$R$: the genome $m$ reads of length $l$ each
Solutions

• Naïve :
  - For each $S_i$
    - For each position $p = l, \ldots, |R|$
      - Try matching $S_i$ to the substring $R[p-l+1, \ldots, p]$

• Complexity:
  $O(lm|R|)$ exact or inexact matching

R: the genome $m$ reads of length $l$ each
Solutions (2)

• Slightly less Naïve:
  - For each $S_i$
    - Match $S_i$ to $R$ using KMP

• Complexity:
  $O(m(l + |R|)) = O(ml + m|R|)$ exact matching

$R$: the genome $m$ reads of length $l$ each
Solutions (3)

- **Suffix tree approach:**
  - Build suffix tree for $R$
  - For each $S_i$
    - Find matches of $S_i$ to $R$ by tree traversal from the root

- **Time complexity:** $O(lm + |R|)$ exact matching

- **Space Complexity:** $O(|R|\log|R|)$ including leaf labels $\text{vs} \ |R|\log|\Sigma|$ for the text

- Can store Human Genome text in 750M bytes (6G bits), but need $\sim64G$ bytes for the tree
  - large constants, hard to implement

$R$: the genome $m$ reads of length $l$ each
Solutions (4)

• Preprocessing:
  - Create hash-table $H$
  - For each position $p = l, \ldots, |R|$
    - Hash (key=$R[p-l+1, \ldots, p]$, value=$p$) in $H$

• For each $S_i$ report $H(S_i)$

• Time complexity:
  $O(lm + |R|l)$

• Problems:
  Only exact matching, memory $O(|R|l)$
Improvements

- Pack strings into bit vectors
  - Need only $2l$ bits for each read
  - Pick hash of size that fits in memory

- Partition the genome to $|R|/|Mem|$ chunks

TGAGTGAC

11100010111011100001
Mapping Issues

Goal: Mapping, Alignment, and Calling

• Problems:
  - $>10^8$ reads to analyze
  - Repetitive regions: non-unique mapping
  - Mutations/errors: imperfect match, incorrect mapping, diploid samples

• Solutions (to be expanded later):
  - Hash while allowing mismatches
  - Map as best as you can + report probability of error
  - Combine data from multiple reads for calling
Outline of approach

1. Hash the reads, then scan the reference
2. Identify the reference position with the lowest number of mismatches
3. Assign each alignment probability it is not true
4. Use mapped reads to call nucleotide with variant scores
Read alignment with mismatches

1. **Indexing**: Hash the first $n$ bits iteratively into 2 masked tables.

2. Check reference against hash

3. If hit:
   - score it (#mismatches)
   - keep the best

6 templates find 100% of 2mm and 57% 3mm.

20 templates find 100% of 3mm and 64% 4mm.

*Gapped Seeds* - Similar ideas improve accuracy of homology searches like BLAST
1. Index the first 28 bases of each read with templates 1,2; generate H1, H2
2. Scan the reference: \( \forall \) position and orientation, hash with template 1,2; if hit - extend and score the complete read; keep per read 2 best scored hits and no. of 0/1/2 mismatch hits only
3. Repeat with templates 3,4, then 5,6

~Complexity: Time: 1. \( O(ml) \) 2. \( O(|R|l) \)
Space: 1. \( O(ml) \) 2. \( O(ml + |R|) \) total, but only \( O(ml) \) in cache

\( R \): the genome
\( m \) reads of length \( l \) each
Mapping qualities

In the sequencing process, every read base is assigned an error probability. Want to use these probs to assign error prob for the mapping of the read

\[ Q_s = -10 \log_{10} \Pr(\text{read is wrongly mapped}) \]

“Phred-scaled quality”

e.g. \( Q_s = 30 \): prob of incorrect mapping of the read is \( 10^{-3} \)

Assume sequencing errors are independent along read

\( p(z|x,u) \): Pr. read \( z \) coming from position \( u \) in reference \( x \)

If \( z \) mapped to \( u \) has 2 mismatches with Phred base qualities \( 10, 20 \) then

\[ p(z|x,u) = 10^{-\frac{(20 + 10)}{10}} = 0.001 \]
Mapping qualities (2)

\[ x = \text{reference seq}; \ z = \text{read seq}; \ u = \text{mapped position}; \text{ Assume uniform prior } p(u|x) \]

\[
p_s(u \mid x, z) = \frac{p(u, x, z)}{p(x, z)} = \frac{p(z \mid x, u)p(x \mid u)p(u)}{p(x, z)} = \frac{p(z \mid x, u)p(u)}{\sum_{v=1}^{\lvert R \rvert - l + 1} p(z \mid x, v)p(v)} = \frac{p(z \mid x, u)}{\sum_{v=1}^{\lvert R \rvert - l + 1} p(z \mid x, v)}
\]

\[ p(z \mid x, u) = \text{product of mismatch error probabilities} \]
\[ \Sigma - \text{estimated from best hit, 2^{nd} best, all other hits} \]

\[
Q_s(u \mid x, z) = -10 \log_{10} (1 - p_s(u \mid x, z))
\]
Paired-end reads

- Index “mate” reads $s_1, s_2$ simultaneously
- Hash reference
- For mates that hit uniquely & consistently (right orientation, distance): $Q_p = Q_{s1} + Q_{s2}$
- For mates that hit in multiple places: $(Q_{s1}, Q_{s2})$
- For mapped $s_1$, but no hit to $s_2$, find gapped alignment (Smith-Waterman) to detect short indels
Summary

- Indexing speeds up alignment in DB-search; mapping
- Storage and cache considerations are paramount!

Source:
Li, Ruan and Durbin
Mapping short DNA sequencing reads and calling variants using mapping quality scores, Genome Research, 2008
More on the NGS technology
Illumina NGS technique

Figure 2: Prepare Genomic DNA Sample
Randomly fragment genomic DNA and ligate adapters to both ends of the fragments.

Figure 3: Attach DNA to Surface
Bind single-stranded fragments randomly to the inside surface of the flow cell channels.

VIDEO: http://www.illumina.com/techniques/sequencing/dna-sequencing.html
Figure 4: Bridge Amplification

Add unlabeled nucleotides and enzyme to initiate solid-phase bridge amplification.

Figure 5: Fragments Become Double Stranded

The enzyme incorporates nucleotides to build double-stranded bridges on the solid-phase substrate.
Figure 6: Denature the Double-Stranded Molecules

Denaturation leaves single-stranded templates anchored to the substrate.

Figure 7: Complete Amplification

Several million dense clusters of double-stranded DNA are generated in each channel of the flow cell.
**Figure 8: Determine First Base**

The first sequencing cycle begins by adding four labeled reversible terminators, primers, and DNA polymerase.

---

**Figure 9: Image First Base**

After laser excitation, the emitted fluorescence from each cluster is captured and the first base is identified.
Figure 10: Determine Second Base

The next cycle repeats the incorporation of four labeled reversible terminators, primers, and DNA polymerase.

Figure 11: Image Second Chemistry Cycle

After laser excitation, the image is captured as before, and the identity of the second base is recorded.
**Figure 12: Sequencing Over Multiple Chemistry Cycles**

The sequencing cycles are repeated to determine the sequence of bases in a fragment, one base at a time.

**Figure 13: Align Data**

The data are aligned and compared to a reference, and sequencing differences are identified.
Base calling from raw data

The identity of each base of a cluster is read off from sequential images.
Using 3D data

- Use Hi-C to create contact maps between chromosomal segments (Lieberman-Aiden et al. Science 09)
- Correct for statistical bias (Yaffe & Tanay Nat Gen 11)
- Are functional groups closer in space?
Sequence census methods

• High throughput DNA sequencers are also broadly utilized as bean counters for “sequence census” methods.

• The vast majority of sequencing experiments are used *-seq protocols:

Assays include: ChIP-Seq, RNA-Seq, methyl-Seq, GRO-Seq, Clip-Seq, BS-Seq, FRT-Seq, TraDI-Seq, Hi-C, ...
Overview of Selected HTS Applications Publication date of a representative article describing a method versus the number of citations that the article received. Methods are colored by category, and the size of the data point is proportional to publication rate (citations/months). The inset indicates the color key as well the proportion of methods in each group. For clarity, seq has been omitted from the labels.

Reuter et al. High-Throughput Sequencing Technologies Mol Cell 15
NGS algorithmics
2. Efficient indexing of sequences

Bowtie and the Burrows-Wheeler Transform
Parameters

• Assume:
  - $m$: $10^7 - 10^8$ reads of length $l$: 100-200 per run
  - Possibly paired-ends
  - Genome $R$ of length $3 \times 10^9$
  - 2-8 Gb memory
  - May also want to analyze multiple runs together, resulting in $m > 10^9$

$R$: the genome
$m$ reads of length $l$ each
Mapping Reads

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
```

The reads are:

```
GGTCGGTGAG
TGAGTGTGAC
TGGTGTGGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAAA
```
Indexing Reads

$m l$-long reads

1. GGTCGGTGAG
2. TGAGTGTGAC
3. TGGTGTTGTC
4. TGACTGGTTT
5. AATGGTCGGT
6. GAGTGTGACT
7. AAAAAAAAAA

G-long genome

TTATGGTGCAGTGACTGGTGTTGTCTAA

Memory [bits]:
$m \times 2l$ (reads) + $2m \times (2l/2 + \log m)$ (index)
# Indexing the Genome

**G-long genome**
TTATGGTCGCTGAGTGTGACTGGTGTTGTTGTCTAA

**m l-long reads**

| 1. | GGTCGGGTGAG |
| 2. | TGAGTGTGAC |
| 3. | TGGTGTGGTC |
| 4. | TGAICTGGTGT |
| 5. | AATGGTCGCT |
| 6. | GAGTGACTG |
| 7. | AAAAAAAAAA |

| ACTGG 19 | GTGAC 16 | TGGTC 4 |
| AGTGT 13 | GTGAG 10 | TGGTTG 24 |
| ATGGT 3 | GTGTG 14 | TGTGA 15 |
| CGGTG 8 | GTGTT 23 | TTATG 1 |
| CTGGT 20 | GTTGT 25 | TTGTC 26 |

**Memory [bits]:**

\[
[2G \text{ (genome)}] + G \times (2l/2 + \log G) \text{ (index)}
\]
Efficient Indexing: Burrows-Wheeler Transform

the_next_text_that_i_index.
Burrows-Wheeler Transform: Cyclic shifts

the_next_text_that_i_index.
he_next_text_that_i_index.t
e_next_text_that_i_index.th
_next_text_that_i_index.the
ext_text_that_i_index.the_n
xt_text_that_i_index.the_ne
xt_text_that_i_index.the_nex
_text_that_i_index.the_next
text_that_i_index.the_next
Burrows-Wheeler Transform: Cyclic shifts
Burrows-Wheeler Transform: Cyclic shifts sorted

\[ \text{BWT}\left(\text{the\_next\_text\_that\_i\_index}\right) = \text{xtietthhdnttt}\--i-axx--.eee \]
Inverting BWT

- How many “e” do we have?
- Can you recover the 1st column?
Inverting BWT

- How many “xt” do we have?
- Can you recover the first two columns?
- Which of the x’s is followed by a “t”? 
- Which of the “xt”-s follows an “e”?
- Which “ext” is in “text”? 
One more example

<table>
<thead>
<tr>
<th>Cyclic Rotations</th>
<th>$F(M(\text{&quot;panamabanananas$&quot;}))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>panamabanananas$</td>
<td>$p a n a m a b a n a n a s$</td>
</tr>
<tr>
<td>$p$panamabanananas</td>
<td>$a b a n a n a s$</td>
</tr>
<tr>
<td>s$panamabanaana</td>
<td>$a m a b a n a n a s$</td>
</tr>
<tr>
<td>as$panamabanan</td>
<td>$a n a m a b a n a n a s$</td>
</tr>
<tr>
<td>nas$panamabana</td>
<td>$a n a n a s$</td>
</tr>
<tr>
<td>anas$panamaban</td>
<td>$a n a s$</td>
</tr>
<tr>
<td>nanas$panamaba</td>
<td>$a s$</td>
</tr>
<tr>
<td>anananas$panamab</td>
<td>$b a n a n a s$</td>
</tr>
<tr>
<td>bananas$panama</td>
<td>$m a b a n a n a s$</td>
</tr>
<tr>
<td>abananas$panam</td>
<td>$n a m a b a n a n a s$</td>
</tr>
<tr>
<td>mabanananas$pana</td>
<td>$n a n a s$</td>
</tr>
<tr>
<td>amabanananas$pan</td>
<td>$n a s$</td>
</tr>
<tr>
<td>namabanananas$pa</td>
<td>$p a n a m a b a n a n a s$</td>
</tr>
<tr>
<td>anamabanananas$p</td>
<td>$s$ $p a n a m a b a n a n a$</td>
</tr>
</tbody>
</table>
Key lemmas

M - the cyclic shifts matrix
L - last col.
F - first col

• In the i-th row of M, L(i) precedes F(i) in the original text: T=....L(i)F(i)....

• The i-th occurrence of char x in L corresponds to the same text character as the i-th occurrence of x in F

“last-first (LF) mapping”
Last-First mapping

$  p  a  n  a  m  a  b  a  n  a  n  a  s  
\text{a}_1  b  a  n  a  n  a  s  $  p  a  n  a  m  
\text{a}_2  m  a  b  a  n  a  n  a  s  $  p  a  n  
\text{a}_3  n  a  m  a  b  a  n  a  n  a  s  $  p  
\text{a}_4  n  a  n  a  s  $  p  a  n  a  m  a  b  
\text{a}_5  n  a  s  $  p  a  n  a  m  a  b  a  n  
\text{a}_6  s  $  p  a  n  a  m  a  b  a  n  a  n  
\text{b}  a  n  a  n  a  s  $  p  a  n  a  m  \text{a}_1  
\text{m}  a  b  a  n  a  n  a  s  $  p  a  n  \text{a}_2  
\text{n}  a  m  a  b  a  n  a  n  a  s  $  p  a  n  \text{a}_3  
n  a  n  a  s  $  p  a  n  a  m  a  b  \text{a}_4  
n  a  s  $  p  a  n  a  m  a  b  a  n  \text{a}_5  
p  a  n  a  m  a  b  a  n  a  n  a  s  $  
s  $  p  a  n  a  m  a  b  a  n  a  n  a  \text{a}_6
Inverting BWT
Out of all $x$-s in the last col, those in the interval of $t$ are $2^{nd}$ and $3^{rd}$ → should appear $2^{nd}$ and $3^{rd}$ in the interval of $x$ (if at all)
Searching the Index

- Search last nucleotide and expand backwards
- Maintain interval of possible matches
Exact or Inexact Match Using BWT

Search for ggta
Matching Using BWT: Details

- To compute intervals fast:
  - Slice blocks of logG rows
  - Maintain #a’s, c’s, g’s, t’s up to each block

- Position ≠ Position after BWT along sequence
Suffix Array Representation

1. the_next_text_that_i_index.
2. the_next_text_that_i_index.the
3. the_next_text_that_i_index.the_next_text_that_i_index.th
4. the_next_text_that_i_index.the_next_text_that_i_index.the
5. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
6. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
7. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
8. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
9. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
10. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
11. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
12. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
13. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
14. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
15. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
16. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
17. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
18. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
19. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
20. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
21. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
22. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
23. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
24. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
25. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
26. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.the
27. the_next_text_that_i_index.the_next_text_that_i_index.the_next_text_that_i_index.
Another viewpoint on finding the position

\[ \star = 11 \]
### Relationships between ST, suffix array and BWT

<table>
<thead>
<tr>
<th>Suffixes ID</th>
<th>Sorted Suffixes</th>
<th>Suffix Array</th>
<th>Sorted Rotations ((A_s \text{ matrix}))</th>
<th>BWT Output ((L))</th>
</tr>
</thead>
<tbody>
<tr>
<td>mississippi$</td>
<td>1 $</td>
<td>12 $mississippi$</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>ississippi$</td>
<td>2 i$</td>
<td>11 i$mississipp</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>ssissippi$</td>
<td>3 ippi$</td>
<td>8 ippi$mississ</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>ssissippi$</td>
<td>4 ississippi$</td>
<td>5 ississippi$miss</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>ississippi$</td>
<td>5 ississippi$</td>
<td>2 ississippi$m</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>ssissippi$</td>
<td>6 mississippi$</td>
<td>1 mississippi$</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>sissippi$</td>
<td>7 pi$</td>
<td>10 pi$mississip</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>ippi$</td>
<td>8 ppi$</td>
<td>9 ppi$mississi</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>ppi$</td>
<td>9 sissippi$</td>
<td>7 sissippi$missis</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>pi$</td>
<td>10 sissippi$</td>
<td>4 sissippi$mis</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>i$</td>
<td>11 sissippi$</td>
<td>6 sissippi$missi</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>12 sissippi$</td>
<td>3 sissippi$mi</td>
<td>i</td>
<td></td>
</tr>
</tbody>
</table>

(a)
Sources

• Langmead, Trapnell, Pop and Sazlberg., Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, Genome Biology '09

• Ferragina & Manzini, Opportunistic data structures with applications, FOCS ‘00

• Li & Durbin, Fast and accurate short read alignment with Burrows-Wheeler transform, Bioinformatics, '09

• Li et al., SOAP2: an improved ultrafast tool for short read alignment, Bioinformatics, ‘09

• Adjeroh, Bell, Mukherjee: The Burrows Wheeler Transform, Springer 2008
Burrows-Wheeler Transform
the not-so-gory details

Sources
Ferragina and Manzini, Opportunistic data structures with applications, FOCS 00
Langmead et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, Genome Biology 09
Suffix Arrays

- $T[1...u]$ text
- Suffix array $A$ of $T$: lexicographically ordered suffixes of $T$ represented by pointers to their starting points
- $T=acabc \rightarrow A=[3,1,4,5,2]$
- Requires $4u$ bytes in practice
- Can one compress suffix arrays?
Burrows-Wheeler Transform

Forward BWT: T→L

a. \( T \rightarrow T\# \) (\# unique smallest char)
b. Form conceptual matrix \( M \): all cyclic shifts of \( T\# \), sorted lexicographically
c. Transformed text \( L \): the last column of \( T \). \( L = BWT(T) \)

Notation: \( F \): first column in \( M \)

Close connection to suffix arrays!
Key lemmas

• In the i-th row of M, L(i) precedes F(i) in the original text: T=.....L(i)F(i)....

• The i-th occurrence of char X in L corresponds to the same text character as the i-th occurrence of X in F

“last-first (LF) mapping”
LF mapping
Backward BWT (1)
goal: \( L \rightarrow T \)

\( \Sigma \) - alphabet, lex ordered. \# - unique end char.

Compute the array \( C[1, \ldots |\Sigma|] \):

\( C(c) \) is the total no. of chars \{\#,1,.. c-1\} in \( T \)
\( \text{Occ}(c, r) \) = no. of occurrences of \( c \) in BWT up to but not including the element at index \( r \)

Alg Stepleft(\( r \)):

1. Return \( C[\text{BWT}[r]]+1+\text{Occ}[\text{BWT}[r], r] \)
Backward BWT (2)

goal: L → T

a. \( r \leftarrow 1; \ T \leftarrow " " \)
b. While BWT[\( r \)]\( \neq $ \) do
c. \( T \leftarrow \text{prepend } \text{BWT}[r] \text{ to } r \)
d. \( r \leftarrow \text{stepleft}[r] \)
e. Return T
Alg Exactmatch ( P[1,p] )
goal: Find the interval [sp, ep] of matrix rows that begin with the query P[1,p]

1. \( c \leftarrow P[p]; \) \( sp \leftarrow C[c]+1; \) \( ep \leftarrow C[c+1]+1; \) \( i \leftarrow p-1 \)
2. While \( sp<ep \) and \( i \geq 1 \) do
3. \( c \leftarrow P[i] \)
4. \( sp \leftarrow C[c] + \text{Occ}(c,sp) + 1 \)
5. \( ep \leftarrow C[c] + \text{Occ}(c,ep) + 1 \)
6. \( i \leftarrow i-1 \)
7. Return \( sp, ep \)
Computing $\text{Occ}(c,r)$

1. Precompute $\text{Occ}(c,r)$ naively for each $c,r$
   
   Time $O(|\Sigma|u)$, but space $O(|\Sigma|u \log u)$, for text of length $u$

2. Precompute $\text{Occ}(c,r)$ only for $r=j*p$
   
   When $\text{Occ}(c,r)$ is needed – if $r$ not available go back to the previous multiple of $p$ and add
   
   Time $O(|\Sigma|u)$ to preprocess, $O(p)$ to compute
   
   Space $O((|\Sigma|u \log u)/p)$
Computing the Text location of an exact match

- Problem: Exactmatch P gives row(s) in M that begin with the query - need to find their offset - location in the text

- Solution:
  - mark some rows with pre-calculated offsets.
  - In search, if row is not marked, do Stepleft k times until finding a marked row with offset o; report o+k

- Time/space tradeoff
Burrows & Wheeler

- **David Wheeler (1927-2004)**
  - Educated in Math, Cambridge
  - First CS PhD in the world (51)
  - Invented the subroutine
  - Cambridge prof. active in cryptography

- **Michael Burrows (~1963- )**
  - PhD Cambridge
  - Worked in DEC, Microsoft, now Google
  - Co-developed the AltaVista search engine

NGS algorithmics

3. Sequence assembly
What if we have no reference?

TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA

GGTCGGTGAG
TGAGTGTGAC
TGGTGGTTGTC
TGACTGGTT
AATGGTCGGT
GAGTGTGACT
Sequence Assembly

- **Input:**
  - GGTCGGTGAG
  - TGAGTGTGAC
  - TGGTGTTGTC
  - TGACTGGTTT
  - AATGGTCGGT
  - GAGTGTGACT
  - AAAAAAAAAA

- **Output:**
  - ATATGGTCGGTGAGTGTGACTGGTTGTTGCTAA
de Bruijn graph of order $k$

- Vertex = $k$-mer $(x_1,...,x_k)$ \( |V| = |\Sigma|^k \)

- Directed edge for overlap of $(k-1)$:
  \[
  (x_1,x_2,...,x_k) \rightarrow (x_2,...,x_k,x_{k+1})
  \]
  \( |E| = |\Sigma|^{k+1} \)

- \( \forall v \ d^+(v) = d^-(v) = |\Sigma| \)

We use the name “de Bruijn graph” also for subgraphs induced by a subset of the vertices
de Bruin Graph

Nodes: k-tuples
Idea: de Bruin Graph

Edges: (k+1)-tuples
Sequence ⇔ Path in Graph
Assembly Using de-Bruijn Graphs

Input:
GGTCGGTGAG
TGAGTGTGAC
TGGTGTGTC

1. Turn reads to paths
GGTC → GTCG → TCGG → CGGT → GGTG → GTGA → TGAG
TGAG → GAGT → AGTG → GTGT → TGTG → GTGA → TGAC
TGGT → GGTC → GTGT → TGTT → GTTG → TTGT → TGTC
Assembly Using de-Bruijn Graphs

1. Turn reads to paths
2. Merge paths

GGTC → GTCG → TCGG → CGGT → GGTG → GTGA → TGAG
TGAG → GAGT → AGTG → GTGT → TGTG → GTGA → TGAC

GGTC → GTCG → TCGG → CGGT → GGTG → GTGA → TGAG
GAGT → AGTG → GTGT → TGTG → GTGA → TGAC
Assembly Using de-Bruijn Graphs

1. Turn reads to paths
2. Merge paths
3. Resolve error “bubbles”
4. Resolve cycles (repeats)
Bibliography:

- Zerbino & Birney, Velvet: Algorithms for De Novo Short Read Assembly Using De Bruijn Graphs, Genome Research 2008