DNA Sequencing Data Compression

Michael Chung
Problem

- DNA sequencing per dollar is increasing faster than storage capacity per dollar.

Stein (2010)
Data

- 3 billion base pairs in human genome
- Genomes are 99.9% similar
  - 95% SNPs
Data

- Image
- Base Sequences
  - Quality Scores
- Alignment
- Variation Detection
Data

- Image
  - Base Sequences
    - Quality Scores
  - Alignment
  - Variation Detection
Data

Image

Base Sequences
Quality Scores

Alignment

Variation
Detection

A: 87.5%
Data

- Image
- Base Sequences Quality Scores
- Alignment
- Variation Detection

A: 87.5%
T: 75%
A: 12.5%
Data

Image

Base Sequences
Quality Scores

Alignment

Variation
Detection

A: 87.5%
T: 75%
A: 12.5%
Data

![Diagram showing data processing steps: Image, Base Sequences, Quality Scores, Alignment, Variation Detection, with corresponding percentages A: 87.5%, T: 75%, G: 62.5%, A: 12.5%, T: 12.5%]
Data

Image

Base Sequences
Quality Scores

Alignment

Variation
Detection

A: 87.5%
T: 75%  A: 12.5%
G: 62.5%  T: 12.5%
Data

- Image
- Base Sequences
- Quality Scores
- Alignment
- Variation Detection

Percentages:
- A: 87.5%
- T: 75%
- A: 12.5%
- G: 62.5%
- T: 12.5%
- A: 37.5%
- G: 25%
- T: 12.5%
Data

Image

Base Sequences
Quality Scores

Alignment

Variation Detection

Automated image recognition
Further error is accumulated
Data

Some reads cannot be aligned
As a result, structural data can be lost
Data

Variation data may be:
- incorrect
- suboptimal for compression
- suboptimal for analysis
Compression Tools

• Fixed Codes
  – VINT
  – Golomb / Golomb-Rice
  – Elias Codes
  – MOV Codes

• Variable Codes
  – Huffman Codes
Variable Integers (VINT)

- Standard integer data type uses 4 bytes

- Example: Binary code for 5:

```
0000000000000000000000000000000000000000000000000000101
```
Variable Integers (VINT)

• Standard integer data type uses 4 bytes

• Example: Binary code for 5:

  \[000000000000000000000000000000000000000000001011\]

• Small integers don’t need full 4 bytes
• Last bit of each byte used as a flag:
  – 0: VINT continues to next byte
  – 1: End of VINT
Golomb / Golomb-Rice

- Encode an integer with two bit strings
  - Preamble
  - Mantissa

- Choose parameter $m$ to encode integer $j$:
Example
- Encode 61 with $m = 8$:

\[
\left\lfloor \frac{61}{8} \right\rfloor = 7 \text{ bits}
\]

\[
61 \mod 8 = 5
\]
Delta Encoding

• Store relative integers, not absolute integers

Instead of 2, 4, 6, 8, 10
Store 2, 2, 2, 2, 2
Elias Codes

• Binary code for 5:

\[
\begin{array}{cccccccccccccccccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{array}
\]

Waste of space

• Can we compress the leading zeroes?
Elias Codes

To encode integer $j$:

\[ n = \lfloor \log j \rfloor \]

- $n$ bits
- $n + 1$ bits
Monotone Value Coding (MOV Coding)

- Encode $j_1, j_2, \ldots, j_k$
- Encode $j_1$ with Elias Gamma encoding
- Preamble: $(\log j_i - \log j_{i-1})$ 0-bits
- Mantissa: binary code for $j_i$ starting with most significant 1 bit
Monotone Value Coding (MOV Coding)

2: 0 10
3: 11
4: 0 100
5: 101
6: 110
7: 111
8: 0 1000
Huffman Encoding

I will not pass notes in class.
I will not pass notes in class.
I will not pass notes in class.
I will not pass notes in class.
I will not pass notes in class.
Huffman Encoding

Let 0 = I will not pass notes in class.

00000
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td></td>
</tr>
<tr>
<td>CTCT</td>
<td></td>
</tr>
<tr>
<td>CTCT</td>
<td></td>
</tr>
<tr>
<td>CTCT</td>
<td></td>
</tr>
<tr>
<td>AGCT</td>
<td></td>
</tr>
<tr>
<td>AGCC</td>
<td></td>
</tr>
<tr>
<td>AGCC</td>
<td></td>
</tr>
<tr>
<td>AGCC</td>
<td></td>
</tr>
<tr>
<td>TGAA</td>
<td></td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
</tbody>
</table>
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
</tbody>
</table>
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
</tbody>
</table>
Huffman Encoding

• Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
</tbody>
</table>

CTCT  CTCT  CTCT  CTCT
CTCT  CTCT  CTCT  CTCT
AGCT  AGCC  AGCC  AGCC
TGAA  CATC  CATC  CATC
CATC  CATC  CATC  CATC
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
</tr>
<tr>
<td>CATC</td>
<td></td>
</tr>
</tbody>
</table>
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
</tr>
</tbody>
</table>

Example sequence: CTCT CTCT CTCT CTCT CTCT CTCT AGCT AGCC AGCC AGCC TGAA CATC CATC CATC CATC
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
</tr>
</tbody>
</table>

CTCT  CTCT  CTCT  CTCT  CTCT  CTCT  AGCT  AGCC  AGCC  AGCC  TGAA  CATC  CATC  CATC  CATC  CATC
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
</tr>
</tbody>
</table>
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram showing the Huffman encoding process with a frequency table and a corresponding binary tree.
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
</tr>
</tbody>
</table>
Huffman Encoding

• Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
<th>Huffman Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
<td>01</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
<td>001</td>
</tr>
</tbody>
</table>
Huffman Encoding

- Huffman encoding assigns shorter tags to more frequently encountered data.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Occurrences</th>
<th>Huffman Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCT</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>CATC</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>AGCC</td>
<td>3</td>
<td>01</td>
</tr>
<tr>
<td>AGCT</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>TGAA</td>
<td>1</td>
<td>001</td>
</tr>
</tbody>
</table>

101000101 ↔ CTCT CTCT TGAA AGCC
Papers

• 2008 Human genomes as email attachments
  Scott Christley, Yiming Lu, Chen Li, Xiaohui Xie

• 2010 Data structures and compression algorithms for high-throughput sequencing technologies
  Kenny Daily, Paul Rigor, Scott Christley, Xiaohui Xie, Pierre Baldi

• 2011 Efficient Storage of high throughput DNA sequencing data using reference-based compression
  Markus His-Yang Fritz, Rasko Leinonen, Guy Cochrane, Ewan Birney
Papers

- **2008** Human genomes as email attachments
  
  *Scott Christley, Yiming Lu, Chen Li, Xiaohui Xie*

- **2010** Data structures and compression algorithms for high-throughput sequencing technologies
  
  *Kenny Daily, Paul Rigor, Scott Christley, Xiaohui Xie, Pierre Baldi*

- **2011** Efficient Storage of high throughput DNA sequencing data using reference-based compression
  
  *Markus His-Yang Fritz, Rasko Leinonen, Guy Cochrane, Ewan Birney*
Summary

[x 1 → 247 MB]

[x 46 → 11.4 GB]
Summary

• Genomes are 99.9% similar
  – Store only the differences

x 1  ➔  247 MB

x 46  ➔  11.4 GB

x 46  ➔  4 MB
Assumptions

Reference

A  G

C  T

correct variation data

quality scores
A: 85%
G: 72%
C: 65%
A: 43%

unaligned reads
?
?
?

→

Trash can
Method
Method

Reference:

Starting from the 7th position, G is replaced with C.
Starting from the 17th position, A is replaced with G.
Starting from the 21st position, C is replaced with A.

The compressed sequence is 00101100...
Techniques

- SNP Mapping

<table>
<thead>
<tr>
<th>SNP</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A → G</td>
</tr>
<tr>
<td>0</td>
<td>C → G</td>
</tr>
<tr>
<td>1</td>
<td>A → T</td>
</tr>
<tr>
<td>1</td>
<td>T → C</td>
</tr>
<tr>
<td>0</td>
<td>T → G</td>
</tr>
<tr>
<td>0</td>
<td>A → C</td>
</tr>
</tbody>
</table>
Techniques

• SNP Mapping

• NCBI dbSNP
  – 11 million SNPs
  – 99% are bi-allele
Techniques

• VINT and Delta Positions
  – Use for indels (insertions and deletions)
Techniques

- **VINT and Delta Positions**
  - Use for indels (insertions and deletions)
Techniques

- **VINT and Delta Positions**
  - Use for indels (insertions and deletions)

```
0001010101 00001001

Position +10 relative to last position.
End of VINT
Insertion length 4
```

```
01 00 00 11
C   A   A   G
```

```
0001010101 00001001

Position +10 relative to last position.
End of VINT
Program length 4
```

```
0001010101 00001001

End of VINT
Deletion length 4
```
Techniques

- K-mer partitioning
Techniques

• K-mer partitioning

CTCTCTCTCTCTCTCTCTCTCT

\[ k = 4 \]

CTCT  CTCT  CTCT  CTCT  CTCT

Huffman Table

CTCT $\Rightarrow$ 01

0101010101

I WILL NOT DO ANYTHING BAD EVER AGAIN.
Techniques

• K-mer partitioning

CTCTCTCTCTCTCTCTCTCTCTCT

↓ k = 5

CTCTC TCTCT CTCTC TCTCT
Techniques

• K-mer partitioning

CTCTCTCTCTCTCTCTCTCTCTCTCT

k = 5

CTCTCTCTCTCTCTCTCTCTCTCTCT

CTCTC  TCTCT  CTCTC  TCTCT

CTCTC ⇒ 000
TCTCT ⇒ 001

000001000001

Huffman Table
Results

James Watson’s Genome

- 3.5 M differences
  - 3.3 M SNPs
  - 220 K indels

4 MB
Review

• Good general concepts used in later research
• Compare to reference genome
• Take advantage of:
  – Common SNPs
  – Position integer compression + delta encoding
  – Frequency analysis based compression (Huffman)
Papers

• 2008  Human genomes as email attachments
  Scott Christley, Yiming Lu, Chen Li, Xiaohui Xie

• 2010  Data structures and compression algorithms for high-throughput sequencing technologies
  Kenny Daily, Paul Rigor, Scott Christley, Xiaohui Xie, Pierre Baldi

• 2011  Efficient Storage of high throughput DNA sequencing data using reference-based compression
  Markus His-Yang Fritz, Rasko Leinonen, Guy Cochrane, Ewan Birney
Summary

- Compress position and length integers with various compression methods
- Fixed code compression works well
Assumptions
Method

Substitution triples

Example: (1000, 30, 3A)
Method

Substitution triples

Example: (1000, 30, 3A)
Method

Substitution triples

Example: (1000, 30, 3A)

Reference

4000

1000

5000

Reads

30 bases long
Method

Substitution triples

Example: (1000, 30, 3A)

Reference

4000

5000

Reads

A

30 bases long
# Results

## Compressed Data Size (Bytes)

<table>
<thead>
<tr>
<th>Compression Method</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenCompress</td>
<td>56,166,419</td>
<td>36,099,480</td>
<td>390,541,330</td>
</tr>
<tr>
<td>gzip</td>
<td>41,378,624</td>
<td>95,688,992</td>
<td>618,818,824</td>
</tr>
<tr>
<td>bzip2</td>
<td>42,233,336</td>
<td>94,030,320</td>
<td>955,061,616</td>
</tr>
<tr>
<td>7zip</td>
<td>30,651,664</td>
<td>83,319,584</td>
<td>411,811,520</td>
</tr>
</tbody>
</table>
Relative Elias Gamma (REG) indexed encoding performs best

Huffman coding can outperform when dataset is highly repetitive
Review

• **Pros**
  – Efficient encoding of integers
  – Can handle read-level compression

• **Cons**
  – Storage space used to represent data identical to reference
  – Does not handle quality scores
  – Assumes aligned data
Papers

• 2008  Human genomes as email attachments
  Scott Christley, Yiming Lu, Chen Li, Xiaohui Xie

• 2010  Data structures and compression algorithms for high-throughput sequencing technologies
  Kenny Daily, Paul Rigor, Scott Christley, Xiaohui Xie, Pierre Baldi

• 2011  Efficient Storage of high throughput DNA sequencing data using reference-based compression
  Markus His-Yang Fritz, Rasko Leinonen, Guy Cochrane, Ewan Birney
Summary

• Lossy compression using quality budget
Summary

- Lossy compression using quality budget

- Include extra sequences in reference
  - common unmapped reads
  - bacterial / viral sequences
Summary

• Lossy compression using quality budget

• Include extra sequences in reference
  – common unmapped reads
  – bacterial / viral sequences

• Support paired-end reads
Summary

• Lossy compression using quality budget

• Include extra sequences in reference
  – common unmapped reads
  – bacterial / viral sequences

• Support paired-end reads

• 0.02 – 0.66 bits / base pair
Assumptions
Method

- Relative Golomb encoding for positions
- Elias Gamma encoding for deletion lengths
- Huffman encoding for read lengths, quality scores
Method

Quality score budget

Reference

Reads
Method

Quality score budget

Reference

Reads

Keep these quality scores
Keep X% of all others
## Results

<table>
<thead>
<tr>
<th>Compression Method</th>
<th>NA12878 chrom20</th>
<th>Pseudomonas syringae pathovar syringae B728a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bzip2</td>
<td>1.09</td>
<td>0.99</td>
</tr>
<tr>
<td>BAM</td>
<td>17.48</td>
<td>7.02</td>
</tr>
<tr>
<td>Reference-based, 2% quality budget</td>
<td>0.59</td>
<td>0.47</td>
</tr>
<tr>
<td>Bzip2, reference-based, 2% quality budget</td>
<td>0.56</td>
<td>0.39</td>
</tr>
</tbody>
</table>
Results

• 0.02 – 0.66 bits / base pair
  – Compared to bzip2 (1 bit / base pair)

• 2 hours for compression
Review

• **Pros**
  - Stores quality scores
  - Handles some unaligned data
  - Allows for controlled loss of precision
  - Can handle read-level compression

• **Cons**
  - Storage space used to represent data identical to reference
  - Still cannot efficiently store all unaligned data
What is too expensive to compress?

- Image data?
- Quality scores?
- Unaligned reads?
Conclusions

• No single compression method is optimal for all data sets.

• Choose assumptions about data carefully.

• Must make decisions about which data to discard, depending on goals.

• Need a centralized database of genomic sequence data
  – Short Read Archive (SRA)
  – Things to consider
    • Storage
    • Speed
    • Accessibility
    • Annotation and mass data organization
    • Scalable, reliable, secure distributed system design
    • Business
References

• 2008  Human genomes as email attachments
  Scott Christley, Yiming Lu, Chen Li, Xiaohui Xie

• 2010  Data structures and compression algorithms for high-throughput sequencing technologies
  Kenny Daily, Paul Rigor, Scott Christley, Xiaohui Xie, Pierre Baldi

• 2010  The case for cloud computing in genome informatics
  Lincoln D. Stein

• 2011  Efficient Storage of high throughput DNA sequencing data using reference-based compression
  Markus His-Yang Fritz, Rasko Leinonen, Guy Cochrane, Ewan Birney