Indexing de Bruijn graph with minimizers

Antoine Limasset

Bonsai Team, CRIStAL, Université de Lille, CNRS

November, 2018
Data Deluge

NovaSeq: 1TB/day
Decreasing cost

![Cost per Genome Graph](image)

$100M$ $10M$ $1M$ $100K$ $10K$ $1K$


100$ Human genome incoming

Moore’s Law

 NIH National Human Genome Research Institute

[genome.gov/sequencingcosts]
Omni-Genomic?

Can biologists sequence the genomes of all the plants and the animals in the world, including this greater bird of paradise in Indonesia? TIM LAMAN/NATIONAL GEOGRAPHIC CREATIVE

Biologists propose to sequence the DNA of all life on Earth

By Elizabeth Pennisi | Feb. 24, 2017, 1:15 PM
Can we work with it?

Kmer/word associative indexing

- CATGCTAGCATAACG -> Found at position 987,654
- AAGTTACGTACGAT -> Present in dataset "Nadine12"
- TTCGATTCCGTTGGG -> Seen 666 times

Fundamental problem

- Sequence similarity (BLAST)
- Overlap detection (Minimap)
- Genome comparison (Mummer)
- Variant calling (Cortex)
- Quantification (Kallisto)
- Assembly (SPAdes)
- ...
Genome Size

<table>
<thead>
<tr>
<th>Species</th>
<th>T2 phage</th>
<th>Escherichia coli</th>
<th>Drosophila melanogaster</th>
<th>Homo sapiens</th>
<th>Paris japonica</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genome Size</td>
<td>170,000 bp</td>
<td>4.6 million bp</td>
<td>130 million bp</td>
<td>3.2 billion bp</td>
<td>150 billion bp</td>
</tr>
<tr>
<td>Common Name</td>
<td>Virus</td>
<td>Bacteria</td>
<td>Fruit fly</td>
<td>Human</td>
<td>Canopy Plant</td>
</tr>
</tbody>
</table>

From http://ib.bioninja.com.au

- Panggenome
- Meta-Genome
- Environmental meta-genome

**Scaling problem**

How to index $10^{10}$, $10^{11}$, $10^{12}$ kmers?
Hash functions

\[ a, b, c, d, e : \text{hashable elements (e.g. strings, integers, etc..)} \]

\[ \rightarrow : \text{hash function} \]

: image \([0;m]\) of hash function
(e.g. indices of buckets in a hash table)

Classical hashing

| Classical hashing | Perfect hashing (no collisions) | Minimal perfect hashing (no collisions, \(|image| = |input|\) |
|-------------------|--------------------------------|---------------------------------|
| a \rightarrow a    | a \rightarrow a               | a \rightarrow a               |
| b \rightarrow e    | b \rightarrow c              | b \rightarrow b               |
| c \rightarrow b,c  | c \rightarrow e              | c \rightarrow e               |
| d \rightarrow d    | d \rightarrow b              | d \rightarrow d               |
| e \rightarrow d    | e \rightarrow d              | e \rightarrow d               |

Constructing a perfect hash means finding the function \(\rightarrow\). Note, storing the function occupies much less space than input elements.
### BBhash\(^1\) library

<table>
<thead>
<tr>
<th>Method</th>
<th>Query time (ns)</th>
<th>MPHF size (bits/key)</th>
<th>Const. time(s)</th>
<th>Const. memory (bits/key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBhash</td>
<td>216</td>
<td>3.7</td>
<td>35</td>
<td>4.3</td>
</tr>
<tr>
<td>EMPHF</td>
<td>246</td>
<td>2.9</td>
<td>2,642</td>
<td>247.1</td>
</tr>
<tr>
<td>EMPHF HEM</td>
<td>581</td>
<td>3.5</td>
<td>489</td>
<td>258.4</td>
</tr>
<tr>
<td>CHD</td>
<td>1037</td>
<td>2.6</td>
<td>1,146</td>
<td>176.0</td>
</tr>
<tr>
<td>Sux4J</td>
<td>252</td>
<td>3.3</td>
<td>1,418</td>
<td>18.10</td>
</tr>
</tbody>
</table>

Achieved the construction of a trillion key MPHF

Alien problem

MPHF have undefined behavior on non-indexed keys (alien keys)

<table>
<thead>
<tr>
<th>BBhash accept aliens</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Key -&gt; [Value]</td>
</tr>
<tr>
<td>▶ CATGCTAGCATA CG   -&gt; Found at position 987,654</td>
</tr>
<tr>
<td>▶ AAGTTACGTACGAT    -&gt; Present in dataset &quot;Nadine12&quot;</td>
</tr>
<tr>
<td>▶ TTCGATTCCGTTGGG   -&gt; Seen 666 times</td>
</tr>
<tr>
<td>▶ TGTGTGTGTGTGTGTG  -&gt; Present in dataset &quot;Nadine12&quot;</td>
</tr>
</tbody>
</table>
Classic solution

Keep the original key

- Key -> [Key, Value]
- CGTCGTCGT -> [AAGTTACGTACGAT, Seen 666 times]
  Alien key detected!

Memory cost per key

- MPHF: half a byte
- 32mer: 4 bytes
- 64mer: 8 bytes

25 GB for a human genome 1.2 TB for P. Japonica
Quasi-dictionary

<table>
<thead>
<tr>
<th>fingerprint</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGT</td>
<td>[...]</td>
</tr>
</tbody>
</table>

**SRC-Linker\(^a\)**

- Key -> [Fingerprint, Value]

A fingerprint of $f$ bits mean a false positive rate of $\approx 1/2^f$


---

**Default value $f = 12$**

Represent $\approx 2$ bytes per kmer for a false positive rate of 0.02\%
Using a reference graph

**De Bruijn graph reference**

A compacted de Bruijn graph can store efficiently a set of kmer (May be as low as 2 3 bit per kmer)

CATGCATGACTGACTGCTGCATCGTAGCTCGATCGTCAGTC
Represent 30 11mer with 41 nucleotide (>3 bits per kmer)
Reference graph encoding

- Key -&gt; [Position in the graph, Value]

Achieved a rate memory usage of 12.5 GB for a human genome (35 bit per kmer)

---

**Partition**

**Pufferfish memory bottleneck**

Position field \(\approx \log_2(\text{genome\_Size})\)

Position field of partitioned graph \(\approx \log_2\left(\frac{\text{genome\_Size}}{\text{number\_partition}}\right)\)

**Using minimizer**

Partition the kmers according to their minimizer

Index each partition separately

**Various advantages**

- Parallel construction
- Cache coherence during query
Blight

Read file to index

> R1
AACTCATGCAAA
>R2
CATGCAACGTC
>R3
GCAAACGTCTGC
>R4
AAACGTCTGCCC
...

De Bruijn graph construction (BCALM2)

De Bruijn graph sequences

> Unitig_sequence_1
AACTCATGCAAA
CGTCTGCCC
...

Split according to minimizers

Index position in:

MPHF_AAA
> Sub_graph_AAA
ATGCAACGT
...

MPHF_AAC
> Sub_graph_AAC
AACTCAT
...

MPHF_CCC
> Sub_graph_CCC
CTGCTGCCC
...

Blight index

Kmer: CTGCCC
Minimizer: CCC
### Memory result

<table>
<thead>
<tr>
<th>Minimizer size</th>
<th>Graph sequences</th>
<th>Positions</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td>12</td>
<td>26</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>9</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>Pufferfish</td>
<td></td>
<td></td>
<td>35</td>
</tr>
</tbody>
</table>

Pufferfish used 12.5 GB for the human Genome  
Blight objective: 8 GB
Time result

Whole human genome

- Construction time: 3,064
- Query time 311
- Pufferfish construction time: 4,248
- Pufferfish query time: 1,331
Objectives

Efficient AND user-friendly library

- Single header to include
- Serialization (index saved on disk)
- Results on largest genome, pangenome, metaGenome

Optimization

- Direct spitted graph construction
- Successive positive query (50 sec on human genome)
- Specialized minimizer scheme
The end

THANK YOU

FOR YOUR ATTENTION