A resource-frugal probabilistic dictionary and applications in (meta)genomics

Camille Marchet, Antoine Limasset, Lucie Bittner, Pierre Peterlongo

Prague Stringology Conference

August, 2016
Sequencing

>read1
ACGACGACGTAGACGACTAGC
AAAACTACGATCGACTAT

>read2
ACTACTACGATCGATGTCGC
GCTGCTCGCTCTCTCTCGCT

...
**Dataset comparison**

**Read set to references sequences**

Huge set of tool

---

**Read set to Read set**

Some tools but that index the dataset in a very roughly manner

<table>
<thead>
<tr>
<th>sequence</th>
<th>ATGGAAGTCGC</th>
<th>GAATC</th>
</tr>
</thead>
<tbody>
<tr>
<td>7mers</td>
<td>ATGGAAG</td>
<td>TGGAA</td>
</tr>
<tr>
<td></td>
<td>TGGAAGT</td>
<td>GGAAG</td>
</tr>
<tr>
<td></td>
<td>GAAGTCG</td>
<td>GAAGT</td>
</tr>
<tr>
<td></td>
<td>AAGTCG</td>
<td>AAGTC</td>
</tr>
<tr>
<td></td>
<td>AGTCG</td>
<td>AGTCG</td>
</tr>
<tr>
<td></td>
<td>GTCGC</td>
<td>GTCGC</td>
</tr>
<tr>
<td></td>
<td>TCGCG</td>
<td>TCGCG</td>
</tr>
<tr>
<td></td>
<td>CGCGG</td>
<td>CGCGG</td>
</tr>
<tr>
<td></td>
<td>GCGGAAT</td>
<td>GCGGA</td>
</tr>
<tr>
<td></td>
<td>CGGAATC</td>
<td>CGGA</td>
</tr>
</tbody>
</table>
Our proposal

The idea

- Filter erroneous kmers
- Index those $k$-mers using an adequate hash function (MPHF)
- Associate information to the $k$-mers according to the task
Hash functions

Classical hashing

Perfect hashing
(no collisions)

Minimal perfect hashing
(no collisions, \( |image| = |input| \))

a, b, c, d, e hashable elements (e.g. strings, integers, etc.)

\[ \begin{array}{c|c}
\text{hash function} & \text{image } [0;m] \text{ of hash function} \\
\end{array} \]

(e.g. indices of buckets in a hash table)
BBhash: our MPHF library

Pros

- Memory efficient (3 or 4 bits by key)
- Fast query (hundreds of ns)
- Fast to construct even for billions keys

Cons

- Static
- No membership operation
- A stranger key can be associated to a value
Quasi-dictionary

Can be reduced into

FP rate \( \approx \frac{1}{2^{\text{fingerprint size}}} \)
Biological question

How to detect similar reads inter set or intra set?
## State of the art

<table>
<thead>
<tr>
<th>Indexed Dataset</th>
<th>Time(s)</th>
<th>Memory(GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blast</td>
<td>Bowtie2</td>
</tr>
<tr>
<td>10K</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>100K</td>
<td>52</td>
<td>51</td>
</tr>
<tr>
<td>1M</td>
<td>795</td>
<td>10,644</td>
</tr>
<tr>
<td>10M</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>100M</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
What we want
Find reads of $A$ that share more than $t$ $k$-mers with a read of $B$. 
SRC Linker

Structure

SRC Linker

Kmer → MPHF

... fingerprint
List of read containing the Kmer
...
...
## Comparison

<table>
<thead>
<tr>
<th>Indexed Dataset</th>
<th>Blast</th>
<th>Bowtie2</th>
<th>BWA</th>
<th>starcode</th>
<th>SRC_linker</th>
<th>Time(s)</th>
<th>Memory(GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0.7</td>
<td>0.04</td>
</tr>
<tr>
<td>100K</td>
<td>52</td>
<td>51</td>
<td>106</td>
<td>29</td>
<td>5</td>
<td>18.5</td>
<td>0.49</td>
</tr>
<tr>
<td>1M</td>
<td>795</td>
<td>10,644</td>
<td>3,155</td>
<td>1,103</td>
<td>45</td>
<td>24.5</td>
<td>3.4</td>
</tr>
<tr>
<td>10M</td>
<td>x</td>
<td>x</td>
<td>62,912</td>
<td>131,139</td>
<td>587</td>
<td>x</td>
<td>5.9</td>
</tr>
<tr>
<td>100M</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>14,748</td>
<td>x</td>
</tr>
</tbody>
</table>
| Full            | x     | x       | x     | x        | x          | 40,828           | x          | 110.84
Memory usage

Bottleneck
More than 100GB for indexing a complete dataset.
Values » indexing structure
Disk version is less expensive but slow

<table>
<thead>
<tr>
<th></th>
<th>Indexation Time (s)</th>
<th>Query Time (s)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One thread 20 threads</td>
<td>One thread 20 threads</td>
<td></td>
</tr>
<tr>
<td>RAM Full</td>
<td>18,067 1,768</td>
<td>17,558 992</td>
<td>110</td>
</tr>
<tr>
<td>Disk Full</td>
<td>106,766 28,471</td>
<td>24,873 1,736</td>
<td>19</td>
</tr>
</tbody>
</table>
What we want

Find reads of $A$ that share $t$ kmers with the reads of $B$ and estimate its coverage in $B$
SRC counter

Structure

SRC Counter

Kmer → MPHF →

<table>
<thead>
<tr>
<th>fingerprint</th>
<th>Kmer count</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Indexed Dataset (nb solid k-mers)</th>
<th>k-mer count time (s)</th>
<th>Construc. time (s)</th>
<th>Memory (GB)</th>
<th>Query Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>QD</td>
<td>Hash</td>
<td>QD</td>
</tr>
<tr>
<td>1M (64,321,167)</td>
<td>2</td>
<td>1</td>
<td>106</td>
<td>0.25</td>
</tr>
<tr>
<td>10M (621,663,812)</td>
<td>15</td>
<td>7</td>
<td>1091</td>
<td>1.80</td>
</tr>
<tr>
<td>50M (2,812,637,134)</td>
<td>72</td>
<td>77</td>
<td>5027</td>
<td>8.00</td>
</tr>
<tr>
<td>100M (5,191,190,377)</td>
<td>196</td>
<td>220</td>
<td>9335</td>
<td>14.71</td>
</tr>
<tr>
<td>Full (8,783,654,120)</td>
<td>486</td>
<td>532</td>
<td>X</td>
<td>24.83</td>
</tr>
</tbody>
</table>
Warning

We still have to assess the qualitative aspect of our methods!

Similar in k-mer content but not in the alignment sense
Conclusion

Take home message
We have entered a new world in which we can index billions of objects.
Many highly consuming applications (for instance bioinformatics) could benefit of such structures in a straightforward way.
Bonus slide BBhash

\[ F_0 \xrightarrow{h_0} A_0 \]

\[
\begin{array}{c}
| \quad \quad \quad \quad \quad |
\end{array}
\]

\[
\begin{array}{c|c}
1 & k_6 \\
0 & k_4, k_2 \\
0 & 0 \\
1 & k_3 \\
0 & k_1, k_5 \\
\end{array}
\]

\[ k_1 k_2 k_3 \\
k_4 k_5 k_6 \\
F_0 \]

\[ h_0(\cdot) \]

\[ k_1 k_2 \]

\[ h_1(\cdot) \]

\[ k_2 k_4 \]

\[ h_2(\cdot) \]

\[ A_0 \xrightarrow{h_1} A_1 \]

\[ A_1 \xrightarrow{h_2} A_2 \]

\[ A_0 \]

\[ A_1 \]

\[ A_2 \]

\[ \begin{array}{c|c|c|c|c|c|c|c}
1 & (1) & 0 & 0 & 0 & 1 & (2) & 0 \\
1 & (3) & 0 & 1 & (4) & 0 & 1 & (5) \\
1 & (6) & \\
\hline
\end{array} \]

\[ A \] (rank of ‘1’s are indicated in parenthesis)