Additional File 1: Proof and queries algorithms to manuscript ”Querying huge read sets in main memory: a versatile data structure”

Nicolas Philippe, Mikaël Salson, Thierry Lecroq, Martine Léonard, Thérèse Commes, Eric Rivals

November 25, 2010

Additional table: elements of $G_{kSA}$ and their rank.

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{kSA}[i]$</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>11</td>
<td>4</td>
<td>14</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>rank</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 1: $G_{kSA}$ with values of identical rank sorted by increasing values. Compared to Figure 1, values having rank 0 are now ordered and appear as 0, 3, 10 instead of 0, 10, 3.

Proof of Theorem 1

Proof of Theorem 1. We want to prove that Algorithm 1 correctly computes the arrays $GkIFA$ and $GkCFA$.

Initialization on lines 2 to 4. $GkSA[0]$ is the smallest $P$-suffix in the lexicographic order among the $P$-suffixes. Therefore, its lexicographic rank is initialized to 0 (line 4), which is recorded in variable $t$ (line 2), and until now there is only one $P_k$-factor lexicographically ranked 0 (line 3).

Main loop on lines 5 to 12. The array $GkSA$, which stores the $P$-suffixes sorted in lexicographic order, is scanned from position 1 to position $\hat{q} - 1$. The rank of the previous $P_k$-factor is stored in $t$ when entering the loop. Line 8 determines whether the current $P_k$-factor differs from the previous one. If it is so, $t$ is incremented by since the current $P_k$-factor is the next one in lexicographic order. Moreover, its counter of occurrences, $GkCFA[t]$, is initialized to zero (line 10). Hence, by induction, we know that $t$ is the lexicographic rank of the
current $P_k$-factor after line 10. In which case, it is recorded in entry $j$ of $GkIFA$ (line 11). Thus, $GkIFA$ is correctly computed.

Moreover, each time a $P_k$-factor having rank $t$ is encountered, its counter, $GkCFA[t]$, is incremented by one (line 12). Hence, at the end of the algorithm $GkCFA[t]$ correctly stores the number of occurrences of $P_k$-factor having rank $t$, as expected from its definition. This concludes the proof.

\[ \square \]

## Algorithms for queries Q2, Q5-Q7

For each of these queries the input consists in a $k$-mer denoted $f$, which is known to occur at position $j$ in $C_R$.

---

**Algorithm 1:** Q2 ($\#Ind_k(f)$)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>2</td>
<td>$t \leftarrow GkIFA[j]$;</td>
</tr>
<tr>
<td>3</td>
<td>$\ell_f \leftarrow GkCFPS[t - 1]$;</td>
</tr>
<tr>
<td>4</td>
<td>$u_f \leftarrow GkCFPS[t]$;</td>
</tr>
<tr>
<td>5</td>
<td>$prev \leftarrow -1$;</td>
</tr>
<tr>
<td>6</td>
<td>CIndk $\leftarrow 0$;</td>
</tr>
<tr>
<td>7</td>
<td><strong>foreach</strong> $i \in [\ell_f, u_f]$ <strong>do</strong></td>
</tr>
<tr>
<td>8</td>
<td>readIndex $\leftarrow \lfloor g^{-1}(GkSA[i])/m \rfloor$;</td>
</tr>
<tr>
<td>9</td>
<td><strong>if</strong> readIndex $\neq prev$ <strong>then</strong></td>
</tr>
<tr>
<td>10</td>
<td>CIndk $\leftarrow$ CIndk + 1;</td>
</tr>
<tr>
<td>11</td>
<td>prev $\leftarrow$ readIndex;</td>
</tr>
<tr>
<td>12</td>
<td><strong>return</strong> (CIndk);</td>
</tr>
</tbody>
</table>

---
Algorithm 2: $Q5 \ (UInd_k(f))$

Data: $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j \ldots j+k-1] = f$

Result: The set $UInd_k(f)$, subset of $Ind_k(f)$ where $f$ occurs only once

1 begin
2 $UInd_k \leftarrow$ empty set;
3 $t \leftarrow GkIFA[j]$;
4 $\ell_f \leftarrow GkCFPS[t-1]$;
5 $u_f \leftarrow GkCFPS[t]$;
6 prev $\leftarrow [g^{-1}(GkSA[\ell_f])/m]$;
7 count $\leftarrow 1$;
8 foreach $i \in [\ell_f, u_f]$ do
9   readIndex $\leftarrow [g^{-1}(GkSA[i])/m]$;
10   if readIndex $\neq$ prev then
11      if count $= 1$ then
12         Add prev to $UInd_k$;
13         count $\leftarrow 1$;
14         prev $\leftarrow$ readIndex;
15      else
16         count $\leftarrow$ count + 1;
17     if count $= 1$ then
18        Add prev to $UInd_k$;
19 return $(UInd_k)$;
Algorithm 3: Q6 (#UInd_k(f))

Data: $f \in \Sigma^k$, $j \in P_{pos}$ such that $C_R[j \ldots j + k - 1] = f$

Result: $\#UInd_k(f)$, the cardinality of $UInd_k(f)$

begin

1: $t \leftarrow GkIFA[j]$;
2: $\ell_f \leftarrow GkCFPS[t - 1]$;
3: $u_f \leftarrow GkCFPS[t]$;
4: prev $\leftarrow \lfloor g^{-1}(GkSA[\ell_f])/m \rfloor$;
5: CUIndk $\leftarrow 0$;
6: count $\leftarrow 1$;

foreach $i \in [\ell_f, u_f[$ do

9: readIndex $\leftarrow \lfloor g^{-1}(GkSA[i])/m \rfloor$;

if readIndex $\neq$ prev then

12: if count $= 1$ then
13: CUIndk $\leftarrow$ CUIndk + 1;
14: count $\leftarrow 1$;
15: prev $\leftarrow$ readIndex;

else

16: count $\leftarrow$ count + 1;

if count $= 1$ then

18: CUIndk $\leftarrow$ CUIndk + 1;

return (CUIndk);
Algorithm 4: Q7 (UPosk(f))

**Data:** $f \in \Sigma^k$, $j \in P_{pos}$ such that $C_R[j \ldots j+k-1] = f$

**Result:** The set $UPos_k(f)$, subset of $Pos_k(f)$ where $f$ occurs only once

```
begin
UPos_k ← empty set;
t ← GkIFA[j];
ℓ_f ← GkCFPS[t - 1];
u_f ← GkCFPS[t];
prev ← ⌊g^-1(GkSA[ℓ_f])/m⌋;
posPrev ← g^-1(GkSA[ℓ_f]) mod m;
foreach i ∈ |ℓ_f, u_f| do
    readIndex ← ⌊g^-1(GkSA[i])/m⌋;
posInRead ← g^-1(GkSA[i]) mod m;
    if readIndex ≠ prev then
        Add the pair (prev, posPrev) to UPos_k;
        prev ← readIndex;
posPrev ← posInRead;
return (UPos_k);
```