RLPBWT con bitvectors

Algorithm 1 Algoritmo per la costruzione di una colonna della RLPBWT con bitvectors

```
function BUILD($col,\ pref,\ div$)
    $c \leftarrow 0,\ u \leftarrow 0,\ v \leftarrow 0,\ u' \leftarrow 0,\ v' \leftarrow 0,\ curr_{lcs} \leftarrow 0,\ tmp_{thr} \leftarrow 0,\ tmp_{beg} \leftarrow 0$
    $start \leftarrow \top,\ beg_{run} \leftarrow \top,\ push_{zero} \leftarrow \bot,\ push_{one} \leftarrow \bot$
    for $every\ k \in [0,\ height)$ do
        if $k = 0 \land col[pref[k]] = 1$ then
            $start \leftarrow \bot$
        if $col[k] = 0$ then
            $c \leftarrow c + 1$
    $runs \leftarrow [0..0]$    ▷ sparse bitvector for runs of length $height + 1$
    $thrs \leftarrow [0..0]$    ▷ sparse bitvector for thresholds of length $height$
    $zeros \leftarrow [0..0]$    ▷ sparse bitvector for zeros of length $c$
    $ones \leftarrow [0..0]$    ▷ sparse bitvector for ones of length $height - c$
    $samples_{beg} \leftarrow [],\ samples_{end} \leftarrow []$    ▷ couple of vectors for samples of length $r$
    if $start$ then
        $push_{one} \leftarrow \top$
    else
        $push_{zero} \leftarrow \top$
    for $every\ k \in [0,\ height)$ do
        if $beg_{run}$ then
            $u \leftarrow u',\ v \leftarrow v',\ tmp_{beg} \leftarrow pref[k]$
            $beg_{run} \leftarrow \bot$
        if $col[pref[k]] = 1$ then
            $v' \leftarrow v' + 1$
        else
            $u' \leftarrow u' + 1$
        if $k = 0 \lor col[pref[k]] \neq col[pref[k-1]]$ then
            $curr_{lcs} \leftarrow div[i],\ tmp_{thr} \leftarrow i$
        if $div[i] < curr_{lcs}$ then
            $curr_{lcs} \leftarrow div[i],\ tmp_{thr} \leftarrow i$
        if $k = height - 1 \lor col[pref[k]] \neq col[pref[k+1]]$ then
            $runs[k] \leftarrow 1,\ thrs[tmp_{thr}] \leftarrow 1$
            $push(samples_{beg}, tmp_{beg})$
            $push(samples_{end}, pref[k])$
            if $push_{one}$ then
                if $v \neq 0$ then
                    $ones[k-1] = 1$
                $swap(push_{zero},\ push_{one})$
            else
                if $u \neq 0$ then
                    $zeros[k-1] = 1$
                $swap(push_{zero},\ push_{one})$
            $beg_{run} \leftarrow \top$
    if $|zeros| \neq 0$ then
        $zeros[|zeros| - 1] \leftarrow 1$
    if $|ones| \neq 0$ then
        $ones[|ones| - 1] \leftarrow 1$
    build rank/select for the four bitvectors
    return ($start,\ c,\ runs,\ zeros,\ ones,\ samples_{beg},\ samples_{end},\ div$)
```

Algorithm 2 Algoritmo per estrazione simbolo da una run in una colonna

```
function GET_SYMBOL($s,\ r$)                ▷ $s = \top$ iff column start with 0, $r$ run index
    if $s$ then
        if $r \bmod 2 = 0$ then return 0 else return 1
    else
        if $r \bmod 2 = 0$ then return 1 else return 0
```

Algorithm 3 Algoritmo per uvtrick

```
function UVTRICK($k,\ i$)                   ▷ $k$ is column index, $i$ row index
    if $i = 0$ then
        return $(0,\ 0)$
    $run \leftarrow rank_h^k(i)$
    if $run = 0$ then
        if $start^k$ then
            return $(index,\ 0)$
        else
            return $(0,\ index)$
    else if $run = 1$ then
        if $start^k$ then
            return $(select_h^k(run) + 1,\ i - (select_h^k(run) + 1))$
        else
            return $(i - (select_h^k(run) + 1),\ select_h^k(run) + 1)$
    else
        if $run \bmod 2 = 0$ then
            $pre_u \leftarrow select_u^k(\frac{run}{2}) + 1$
            $pre_v \leftarrow select_v^k(\frac{run}{2}) + 1$
            $offset \leftarrow i - (select_h^k(run) + 1)$
            if $start^k$ then
                return $(pre_u + offset,\ pre_v)$
            else
                return $(pre_u,\ pre_v + offset)$
        else
            $run_u \leftarrow (\frac{run}{2}) + 1$
            $run_v \leftarrow \frac{run}{2}$
            if $\neg start^k$ then
                $swap(run_u, run_v)$
            $pre_u \leftarrow select_u^k(run_u) + 1$
            $pre_v \leftarrow select_v^k(run_v) + 1$
            $offset \leftarrow i - (select_h^k(run) + 1)$
            if $start^k$ then
                return $(pre_u,\ pre_v + offset)$
            else
                return $(pre_u + offset,\ pre_v)$
```

Algorithm 4 Algoritmo per lf-mapping

```
function LF($k$, $i$, $s$)                ▷ $k$ is column index, $i$ row index, $s$ symbol
    $c \leftarrow rlpbwt[k].c$
    $(u, v) \leftarrow uvtrick(k,\ i)$
    if $s = 0$ then
        return $u$
    else
        return $c + v$
```

Algorithm 5 Algoritmo per lf-mapping inverso

```
function REVERSE_LF($k$, $i$)                ▷ $k$ is column index, $i$ row index
    if $k = 0$ then                          ▷ by design
        return 0
    $k \leftarrow k - 1$
    $c \leftarrow rlpbwt[k].c$
    if $i < c$ then
        if $start^k$ then
            $run \leftarrow rank_u^k(i) \cdot 2$
        else
            $run \leftarrow rank_u^k(i) \cdot 2 + 1$
        $i_{run} \leftarrow 0$
        if $run \neq 0$ then
            $i_{run} \leftarrow select_h^k(run) + 1$
        $(prev_0,\ \_) \leftarrow uvtrick(k,\ i_{run})$
        return $i_{run} + (i - prev_0)$
    else
        if $start^k$ then
            $run \leftarrow rank_v^k(i) \cdot 2 + 1$
        else
            $run \leftarrow rank_v^k(i) \cdot 2$
        $i_{run} \leftarrow 0$
        if $run \neq 0$ then
            $i_{run} \leftarrow select_h^k(run) + 1$
        $(\_,\ prev_1) \leftarrow uvtrick(k,\ i_{run})$
        return $i_{run} + (i - (c + prev_1))$
```

Algorithm 6 Algoritmo per match con aplotipo esterno con panel $width \times height$

```
function EXTERNAL_MATCHES($z$)                    ▷ assuming $|z| = rlpbwt.width$
    $f \leftarrow 0,\ f_{run} \leftarrow 0,\ f' \leftarrow 0$
    $g \leftarrow 0,\ g_{run} \leftarrow 0,\ g' \leftarrow 0$
    $e \leftarrow 0,\ l \leftarrow 0$
    for $every\ k \in [0,\ |z|)$ do
        $f_{run} \leftarrow rank_h^k(f),\ g_{run} \leftarrow rank_h^k(g)$
        $f' \leftarrow lf(k,\ f,\ z[k]),\ g' \leftarrow lf(k,\ g,\ z[k])$
        $l \leftarrow g - f$
        if $f' < g'$ then
            $f \leftarrow f',\ g \leftarrow g'$
        else
            if $k \neq 0$ then
                report $matches\ in\ [e,\ k-1]\ with\ l$ haplotypes
            if $f' = |lcp^{k+1}|$ then
                $e \leftarrow k+1$
            else
                $e \leftarrow lcp^{k+1}[f']$
            if $(z[e] = 0 \wedge f' > 0) \vee f' = height$ then
                $f' \leftarrow g' - 1$
                if $e \geq 1$ then
                    $f_{rev} \leftarrow f',\ k' \leftarrow k+1$
                    while $k' \neq e-1$ do
                        $f_{rev} \leftarrow reverse\_lf(k',\ f_{rev}),\ k' \leftarrow k'-1$
                    $run \leftarrow rank_h^{k'}(f_{rev}),\ symb \leftarrow get\_symbol(start^{k'}, run)$
                    while $e > 0 \wedge z[e-1] = symb$ do
                        $f_{rev} \leftarrow reverse\_lf(e,\ f_{rev})$
                        $run \leftarrow rank_h^{e-1}(f_{rev})$
                        $symb \leftarrow get\_symbol(start^{e-1}, run)$
                while $f' > 0 \wedge (k+1) - lcp^{k+1}[f] \leq e$ do $e \leftarrow e-1$
                $f \leftarrow f',\ g \leftarrow g'$
            else
                $g' \leftarrow f' - 1$
                if $e \geq 1$ then
                    $f_{rev} \leftarrow f',\ k' \leftarrow k+1$
                    while $k' \neq e-1$ do
                        $f_{rev} \leftarrow reverse\_lf(k',\ f_{rev}),\ k' \leftarrow k'-1$
                    $run \leftarrow rank_h^{k'}(f_{rev}),\ symb \leftarrow get\_symbol(start^{k'}, run)$
                    while $e > 0 \wedge z[e-1] = symb$ do
                        $f_{rev} \leftarrow reverse\_lf(e,\ f_{rev})$
                        $run \leftarrow rank_h^{e-1}(f_{rev})$
                        $symb \leftarrow get\_symbol(start^{e-1}, run)$
                while $e < height \wedge (k+1) - lcp^{k+1}[e] \leq e$ do $e \leftarrow e+1$
                $f \leftarrow f',\ g \leftarrow g'$
    if $f < g$ then
        $l \leftarrow g - f$
        report $matches\ in\ [e,\ |z|-1]\ with\ l$ haplotypes
```

Algorithm 7 Algoritmo per match con matching-statistics (MS) e thresholds

```
function MATCHES_MS($z$)
    $ms_{row} \leftarrow [0..0],\ ms_{len} \leftarrow [0..0]$    ▷ ms vectors with row and len of length $|z|$
    $curr_{row} \leftarrow rlpbwt[0].samples_{end}[|rlpbwt[0].samples_{end}| - 1]$
    $curr_{index} \leftarrow curr_{row}$
    $curr_{run} \leftarrow rank_h^0(curr_{index})$
    $symb \leftarrow get\_symbol(start^0, curr_{run})$    ▷ **build matching statistics row**
    for every $k \in [0, |z|)$ do
        if $z[i] = symb$ then
            $ms_{row}[k] \leftarrow curr_{row}$
            if $k \neq |z| - 1$ then
                $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
        else
            $curr_{thr} \leftarrow rank_t^k(curr_{index}),\ single_{run} \leftarrow \bot$
            if $samples_{beg}^k[curr_{run}] = samples_{end}^k[curr_{run}]$ then
                $single_{run} \leftarrow \top$
            if $|samples_{beg}^k| = 1$ then
                $ms_{row}[k] \leftarrow rlpbwt.height$
                if $k \neq |z| - 1$ then
                    $curr_{row} \leftarrow rlpbwt[0].samples_{end}[|rlpbwt[0].samples_{end}| - 1]$
                    $curr_{index} \leftarrow curr_{row}$
                    $curr_{run} \leftarrow rank_h^{k+1}(curr_{index})$
                    $symb \leftarrow get\_symbol(start^{k+1}, curr_{run})$
            else if $(curr_{run} \neq 0 \wedge \neg single_{run} \wedge curr_{run} = curr_{thr}) \vee curr_{run} = |samples_{beg}^k| - 1$ then
                $curr_{index} \leftarrow select_h^k(curr_{run})$
                $curr_{row} \leftarrow samples_{end}^k[curr_{run} - 1]$
                $ms_{row}[k] \leftarrow curr_{row}$
                if $k \neq |z| - 1$ then
                    $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
            else
                $curr_{index} \leftarrow select_h^k(curr_{run} + 1) + 1$
                $curr_{row} \leftarrow samples_{beg}^k[curr_{run} + 1]$
                $ms_{row}[k] \leftarrow curr_{row}$
                if $k \neq |z| - 1$ then
                    $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
    $tmp_{index} \leftarrow 0$    ▷ **build matching statistics len**
    for every $k \in [0, |ms_{row}|)$ do
        if $ms_{row}[k] = rlpbwt.height$ then
            $ms_{len}[k] \leftarrow 0$
        else
            $tmp_{index} \leftarrow i$    ▷ $ra$ is a data structure for random access over the orginale panel
            while $tmp_{index} \geq 0 \wedge z[tmp_{index}] = ra(ms_{row}[k], tmp_{index})$ do
                $tmp_{index} \leftarrow tmp_{index} - 1$
            $ms_{len}[k] \leftarrow k - tmp_{index}$
    for every $k \in [0, |ms_{row}|)$ do    ▷ **build matching statistics matches**
        report match at $ms_{row}[k]$ with $ms_{len}[k]$ iff $ms_{len}[k] > ms_{len}[k + 1]$
        or $ms_{len}[k] = ms_{len}[k + j]$, $j \geq 1$ and $\nexists j$ such that $ms_{len}[k] < ms_{len}[k + j]$
```

Algorithm 8 Algoritmo per match con matching-statistics (MS) e LCE

```
function MATCHES_MS_LCE($z$)
    $ms_{row} \leftarrow [0..0],\ ms_{len} \leftarrow [0..0]$    ▷ ms vectors with row and len of length $|z|$
    $curr_{row} \leftarrow rlpbwt[0].samples_{end}[|rlpbwt[0].samples_{end}| - 1]$
    $curr_{index} \leftarrow curr_{row},\ curr_{run} \leftarrow rank_h^0(curr_{index})$
    $symb \leftarrow get\_symbol(start^0, curr_{run})$    ▷ **build matching statistics row**
    for $every\ k \in [0, |z|)$ do
        if $z[i] = symb$ then
            $ms_{row}[k] \leftarrow curr_{row}$
            if $k = 0$ then
                $ms_{len}[k] \leftarrow 1$
            else
                $ms_{len}[k] \leftarrow ms_{len}[k-1] + 1$
            if $k \neq |z| - 1$ then
                $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
        else
            if $|samples_{beg}^k| = 1$ then
                $ms_{row}[k] \leftarrow rlpbwt.height$
                $ms_{len}[k] \leftarrow 0$
                if $k \neq |z| - 1$ then
                    $curr_{row} \leftarrow rlpbwt[0].samples_{end}[|rlpbwt[0].samples_{end}| - 1]$
                    $curr_{index} \leftarrow curr_{row}$
                    $curr_{run} \leftarrow rank_h^{k+1}(curr_{index})$
                    $symb \leftarrow get\_symbol(start^{k+1}, curr_{run})$
            else
                if $curr_{run} = |samples_{beg}^k| - 1$ then
                    $curr_{index} \leftarrow select_h^k(curr_{run}),\ prev_{pos} \leftarrow samples_{end}^k[curr_{run} - 1]$
                    $lce \leftarrow LCE(k, curr_{pos}, prev_{pos})$
                    $ms_{row}[k] \leftarrow prev_{pos},\ curr_{pos} \leftarrow prev_{pos}$
                    if $k = 0$ then
                        $ms_{len}[k] \leftarrow 1$
                    else
                        $ms_{len}[k] \leftarrow min(ms_{len}[k-1], lce_{len}) + 1$
                    if $k \neq |z| - 1$ then
                        $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
                else if $curr_{run} = 0$ then
                    $curr_{index} \leftarrow select_h^k(curr_{run} + 1) + 1,\ next_{pos} \leftarrow samples_{beg}^k[curr_{run} + 1]$
                    $lce \leftarrow LCE(k, curr_{pos}, next_{pos})$
                    $ms_{row}[k] \leftarrow next_{pos},\ curr_{pos} \leftarrow next_{pos}$
                    if $k = 0$ then
                        $ms_{len}[k] \leftarrow 1$
                    else
                        $ms_{len}[k] \leftarrow min(ms_{len}[k-1], lce_{len}) + 1$
                    if $k \neq |z| - 1$ then
                        $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
                else
                    $prev_{pos} \leftarrow samples_{end}^k[curr_{run} - 1],\ next_{pos} \leftarrow samples_{beg}^k[curr_{run} + 1]$
                    $lce \leftarrow \max_{len}(LCE(k, curr_{pos}, prev_{pos}), LCE(k, curr_{pos}, next_{pos}))$
                    $curr_{pos} \leftarrow lce_{pos}$
                    $ms_{row}[k] \leftarrow curr_{pos}$
                    if $k = 0$ then
                        $ms_{len}[k] \leftarrow 1$
                    else
                        $ms_{len}[k] \leftarrow min(ms_{len}[k-1], lce_{len}) + 1$
                    if $k \neq |z| - 1$ then
                        $(curr_{index},\ curr_{run},\ symb) \leftarrow UPDATE(k, curr_{index}, z)$
    for $every\ k \in [0, |ms_{row}|)$ do    ▷ **build matching statistics matches**
        $report\ match\ at\ ms_{row}[k]\ with\ ms_{len}[k]\ iff\ ms_{len}[k] > ms_{len}[k+1]$
        $or\ ms_{len}[k] = ms_{len}[k+j],\ j \geq 1$ and $\nexists j\ such\ that\ ms_{len}[k] < ms_{len}[k+j]$
```

Algorithm 9 Algorithm to update values for matching statistics of RLPBWT

```
function UPDATE($k, curr_{index}, z$)
    $curr_{index} \leftarrow lf(k, curr_{index}, z[k])$
    $curr_{run} \leftarrow rank_h^{k+1}(curr_{index})$
    $symb \leftarrow get\_symbol(start^{k+1}, curr_{run})$
    return ($curr_{index},\ curr_{run},\ symb$)
```