Linear-Time BWT Construction in Small Space

[Munro, Navarro & Nekrich, SODA 2017]

Motivation

[Munro et al., SODA 2017]

Motivation

[Munro et al., SODA 2017]

- $BWT[i] = T[\mathbf{SA}[i]-1]$ well-known formula for $O(n)$-time BWT

Motivation

[Munro et al., SODA 2017]

- $BWT[i] = T[\mathbf{SA}[i]-1]$ well-known formula for $O(n)$-time BWT
- BWT can be stored in $\mathbf{O(n \lg \sigma)}$ bits (uncompressed)

Motivation

[Munro et al., SODA 2017]

- $BWT[i] = T[\mathbf{SA}[i]-1]$ well-known formula for $O(n)$-time BWT
- BWT can be stored in $\mathbf{O(n \lg \sigma)}$ bits (uncompressed)
- But: algorithms using $\mathbf{SA}$ require $\mathbf{O(n \lg n)}$ bits of working space!
  - … or they no longer run in (deterministic) time $O(n)$

Motivation

[Munro et al., SODA 2017]

- $BWT[i] = T[\mathbf{SA}[i]-1]$ well-known formula for $O(n)$-time BWT
- BWT can be stored in $\mathbf{O(n \lg \sigma)}$ bits (uncompressed)
- But: algorithms using $\mathbf{SA}$ require $\mathbf{O(n \lg n)}$ bits of working space!
  - … or they no longer run in (deterministic) time $O(n)$
- New $O(n)$-time algorithm using only $\mathbf{O(n \lg \sigma)}$ bits of space

Motivation

[Munro et al., SODA 2017]

- $BWT[i] = T[\mathbf{SA}[i]-1]$ well-known formula for $O(n)$-time BWT
- BWT can be stored in $\mathbf{O(n \lg \sigma)}$ bits (uncompressed)
- But: algorithms using $\mathbf{SA}$ require $\mathbf{O(n \lg n)}$ bits of working space!
  - … or they no longer run in (deterministic) time $O(n)$
- New $O(n)$-time algorithm using only $\mathbf{O(n \lg \sigma)}$ bits of space
- Sorting and batch operations – room for **efficient parallelization**!
  - [Fuentes-Sepúlveda et al., Theor. Comput. Sci. 812, 2020]

Running Example

$n = 30, \sigma = 5$

```
    12345678901234567890123456789 0
T = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

Running Example

$n = 30, \sigma = 5$

```
          123456789012345678901234567890
T = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

Rotations:

T_1 = $GATCAATGAGGTGGACACCAGAGGCGGGG

Running Example

$n = 30, \sigma = 5$

```
          123456789012345678901234567890
T = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

Rotations:

T_1 = $GATCAATGAGGTGGACACCAGAGGCGGGG

T_2 = G$GATCAATGAGGTGGACACCAGAGGCGGG

⋮

Metasymbols

$$n = 30,\ \sigma = 5$$

$$\Delta := \lceil \lg_\sigma(n) \rceil = 3$$

Metasymbols

$n = 30, \sigma = 5$

$\Delta := \lceil \lg_\sigma(n) \rceil = 3$

	1	2	3	4	5	6	7	8	9	0
T_1 =	$GA	TCA	ATG	AGG	TGG	ACA	CCA	GAG	GCG	GGG
	1	2	3	4	5	6	7	8	9	0
T_2 =	G$G	ATC	AAT	GAG	GTG	GAC	ACC	AGA	GGC	GGG

- Combine Δ-grams of T_1 and T_2 into metasymbols

Metasymbols

$$n = 30,\ \sigma = 5$$

$$\Delta := \lceil \lg_\sigma(n) \rceil = 3$$

	1	2	3	4	5	6	7	8	9	0
T_1 =	$GA	TCA	ATG	AGG	TGG	ACA	CCA	GAG	GCG	GGG

	1	2	3	4	5	6	7	8	9	0
T_2 =	G$G	ATC	AAT	GAG	GTG	GAC	ACC	AGA	GGC	GGG

- Combine Δ-grams of T_1 and T_2 into metasymbols
- Determine their lex. ranking in time $O(n)$ (e.g., using radix sort)

Metasymbols

$n = 30, \sigma = 5$

$\Delta := \lceil \lg_\sigma(n) \rceil = 3$

	1	2	3	4	5	6	7	8	9	0
T_1 =	$GA	TCA	ATG	AGG	TGG	ACA	CCA	GAG	GCG	GGG

	1	2	3	4	5	6	7	8	9	0
T_2 =	G$G	ATC	AAT	GAG	GTG	GAC	ACC	AGA	GGC	GGG

- Combine Δ-grams of T_1 and T_2 into metasymbols
- Determine their lex. ranking in time $O(n)$ (e.g., using radix sort)
- Compute $SA(T_1T_2)$ in time $O(n/\Delta)$, space $O((n/\Delta)\lg(n/\Delta)) = \boldsymbol{O(n \lg \sigma)}$

Metasymbols

$n = 30$, $\sigma = 5$

$\Delta := \lceil \lg_\sigma(n) \rceil = 3$

	1	2	3	4	5	6	7	8	9	0
T_1 =	$GA	TCA	ATG	AGG	TGG	ACA	CCA	GAG	GCG	GGG

	1	2	3	4	5	6	7	8	9	0
T_2 =	G$G	ATC	AAT	GAG	GTG	GAC	ACC	AGA	GGC	GGG

$SA(T_1T_2)$ = 1, 13, 6, 17, 18, 4, 12, 3, 7, 11, 16, 8, 14, 9, 19, 20, 10, 15, 2, 5

Sequential Algorithm

- $SA(T_1T_2)$ equivalent to suffix array w.r.t. suffixes $i\Delta$ and $i\Delta$-1 of T

Sequential Algorithm

- $SA(T_1T_2)$ equivalent to suffix array w.r.t. suffixes $i\Delta$ and $i\Delta$-1 of T

```
          12345678901234567890123456789 0
T = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

Sequential Algorithm

- $SA(T_1T_2)$ equivalent to suffix array w.r.t. suffixes $i\Delta$ and $i\Delta$-1 of T

```
                1234567890123456789012345678 90
T = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

  - Append symbols at $i\Delta$-1 and $i\Delta$-2 to initially empty string $\boldsymbol{B}$
    - … via one scan of $SA(T_1T_2)$
    - In $\boldsymbol{B}$, we will construct the BWT of T step by step

Example: Sequential Algorithm

$SA(T_1T_2)$ = **1**,13,6,17,18,4,12,3,7,11,16,8,14,9,19,20,10,15,2,5

```
      1   2   3   4   5   6   7   8   9   0
T1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG
      1   2   3   4   5   6   7   8   9   0
T2 = G$G ATC AAT GAG GTG GAC ACC AGA GGC GGG

     12345678901234567890123456789 0
T  = GATCAATGAGGTGGACACCAGAGGCGGGG $
```

Example: Sequential Algorithm

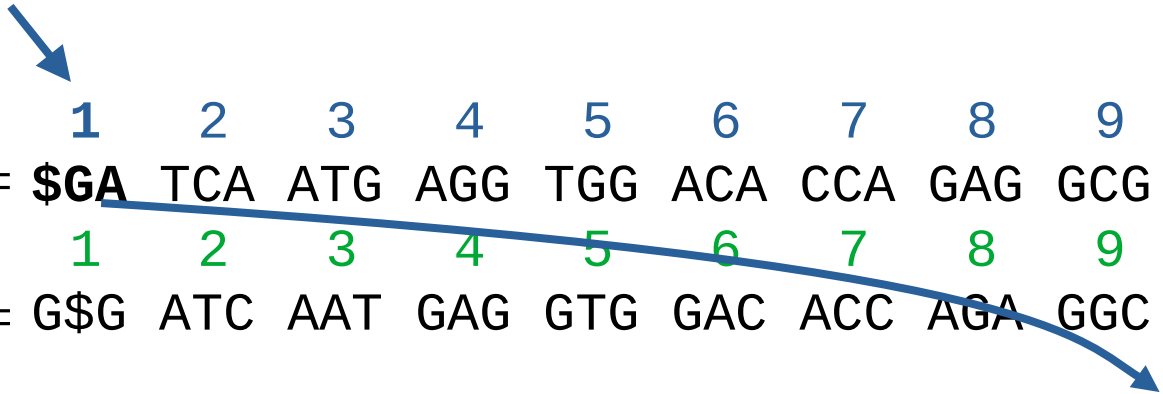
$SA(T_1T_2)$ = **1**,13,6,17,18,4,12,3,7,11,16,8,14,9,19,20,10,15,2,5

```
          1   2   3   4   5   6   7   8   9   0
T1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG
          1   2   3   4   5   6   7   8   9   0
T2 = G$G ATC AAT GAG GTG GAC ACC AGA GGC GGG

     12345678901234567890123456789 0
T  = GATCAATGAGGTGGACACCAGAGGCGGGG $
```

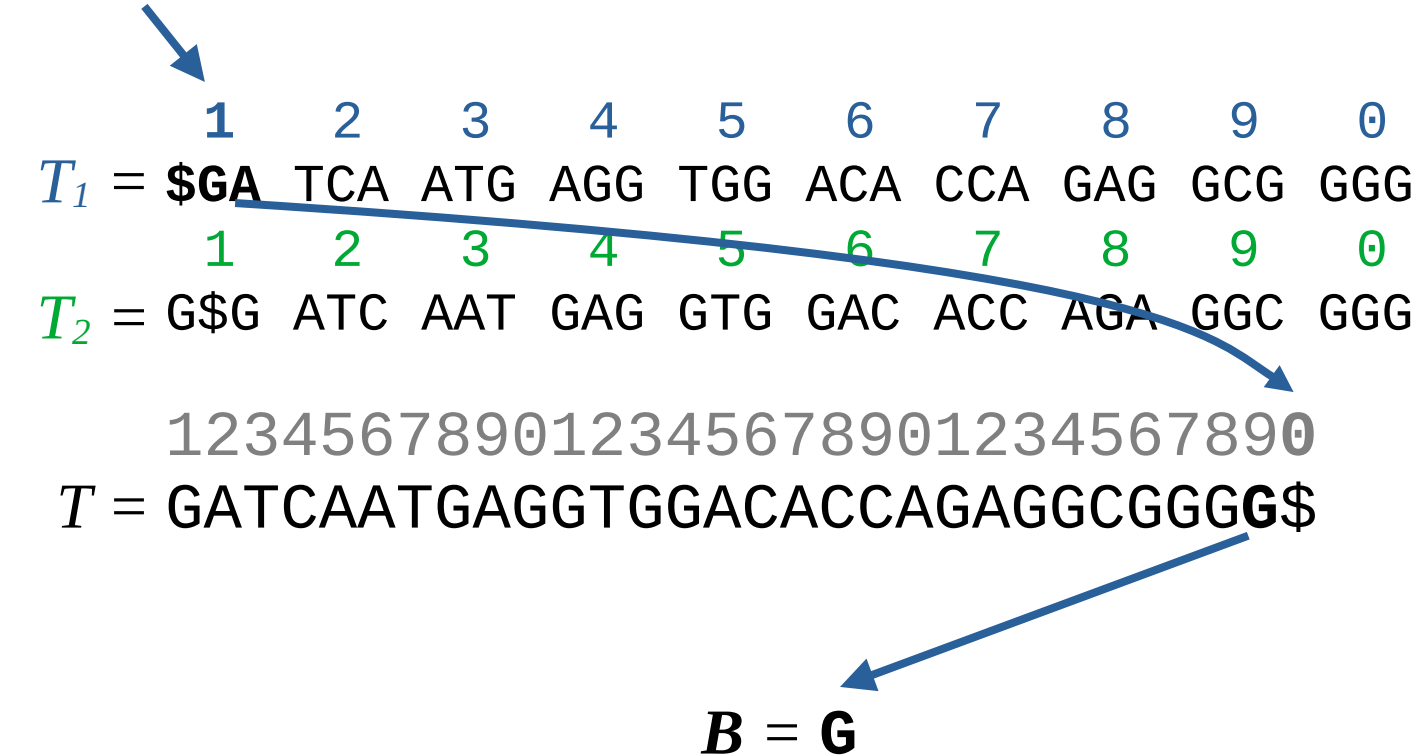
Example: Sequential Algorithm

$SA(T_1T_2)$ = **1**,13,6,17,18,4,12,3,7,11,16,8,14,9,19,20,10,15,2,5



Example: Sequential Algorithm

$SA(T_1T_2)$ = **1**,13,6,17,18,4,12,3,7,11,16,8,14,9,19,20,10,15,2,5



Example: Sequential Algorithm

$SA(T_1T_2)$ = 1, **13**, 6, 17, 18, 4, 12, 3, 7, 11, 16, 8, 14, 9, 19, 20, 10, 15, 2, 5

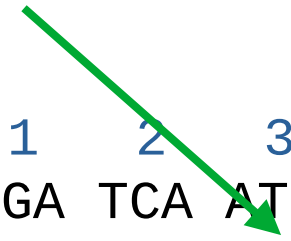
	1	2	3	4	5	6	7	8	9	0
T_1 =	$GA	TCA	ATG	AGG	TGG	ACA	CCA	GAG	GCG	GGG
	1	2	**3**	4	5	6	7	8	9	0
T_2 =	G$G	ATC	**AAT**	GAG	GTG	GAC	ACC	AGA	GGC	GGG

	1234**5**67890123456789012345678 90
T =	GAT**C**AATGAGGTGGACACCAGAGGCGGGG$

$\boldsymbol{B}$ = G**C**

Example: Sequential Algorithm

$SA(T_1T_2)$ = 1, **13**, 6, 17, 18, 4, 12, 3, 7, 11, 16, 8, 14, 9, 19, 20, 10, 15, 2, 5



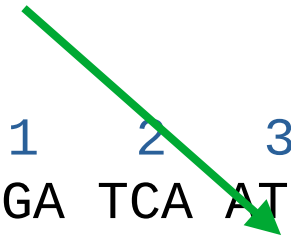
```
        1   2   3   4   5   6   7   8   9   0
T1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG
        1   2   3   4   5   6   7   8   9   0
T2 = G$G ATC AAT GAG GTG GAC ACC AGA GGC GGG

     12345678901234567890123456789 0
T  = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

$\boldsymbol{B}$ = G**C**

Example: Sequential Algorithm

$SA(T_1T_2)$ = 1, **13**, 6, 17, 18, 4, 12, 3, 7, 11, 16, 8, 14, 9, 19, 20, 10, 15, 2, 5



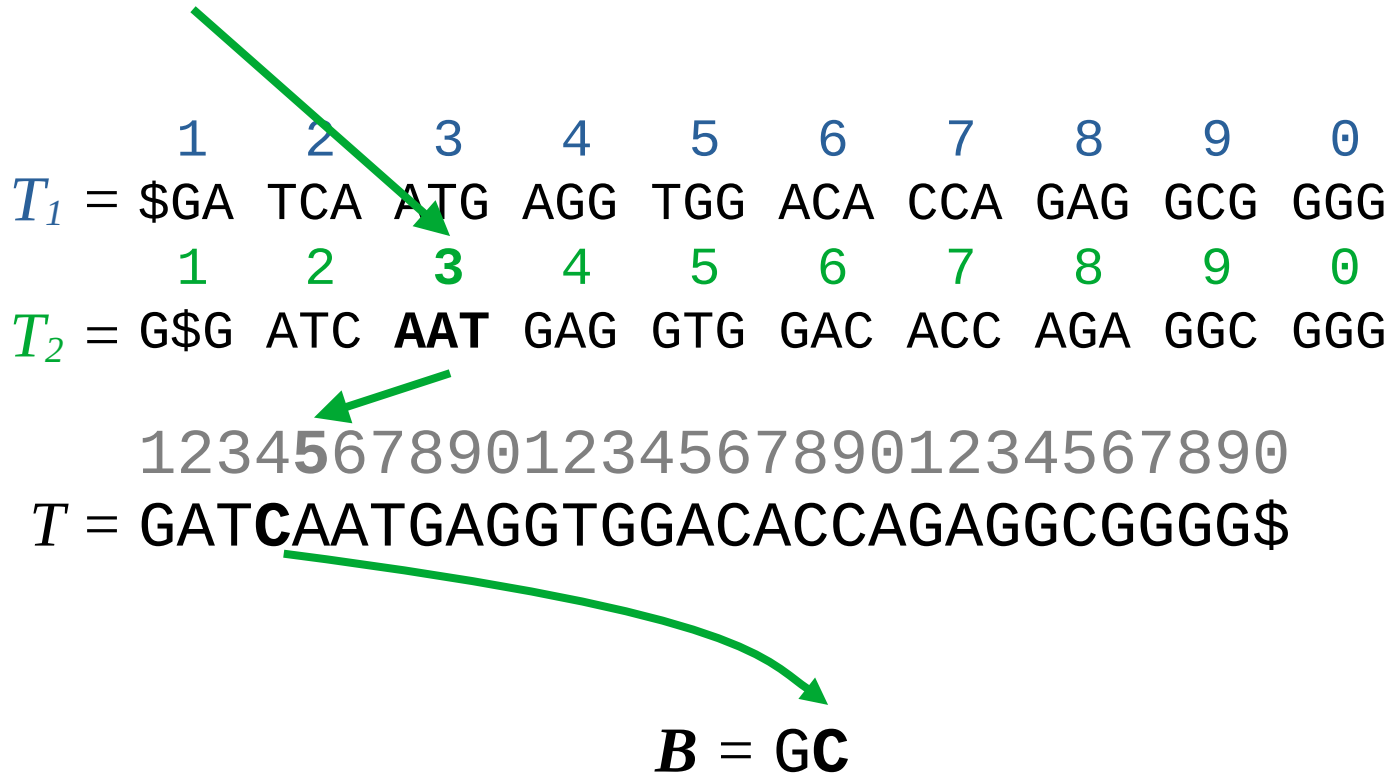
```
        1   2   3   4   5   6   7   8   9   0
T1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG
        1   2   3   4   5   6   7   8   9   0
T2 = G$G ATC AAT GAG GTG GAC ACC AGA GGC GGG
```

```
1234567890123456789012345678 90
T = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

$\boldsymbol{B}$ = G**C**

Example: Sequential Algorithm

$SA(T_1T_2)$ = 1, **13**, 6, 17, 18, 4, 12, 3, 7, 11, 16, 8, 14, 9, 19, 20, 10, 15, 2, 5



Example: Sequential Algorithm

$SA(T_1T_2) = 1,13,6,17,18,4,12,3,7,11,16,8,14,9,19,20,10,15,2,5$

```
      1   2   3   4   5   6   7   8   9   0
T1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG
      1   2   3   4   5   6   7   8   9   0
T2 = G$G ATC AAT GAG GTG GAC ACC AGA GGC GGG

     12345678901234567890123456789
T  = GATCAATGAGGTGGACACCAGAGGCGGGG$
```

$\boldsymbol{B}$ = GCGCCGGAAGGATGACGGAG

Grand Strategy

Grand Strategy

- Build BWT of T in a total of Δ steps
  - Each step will take time $O(n/\Delta)$, so we need time $O(n)$ in total

Grand Strategy

- Build BWT of T in a total of Δ steps
  - Each step will take time $O(n/\Delta)$, so we need time $O(n)$ in total
- After step j, $\boldsymbol{B}$ contans the BWT of T with respect to suffixes at positions $i\Delta-j$, …, $i\Delta-1$
  - Let $T[k\ldots]$ be the $(i+1)$-th smallest suffix, then $B[k] := T[(k-1) \bmod n]$
  - $\boldsymbol{B}$ is a substring of the BWT of T

Grand Strategy

- Build BWT of T in a total of Δ steps
  - Each step will take time $O(n/\Delta)$, so we need time $O(n)$ in total
- After step j, $\boldsymbol{B}$ contans the BWT of T with respect to suffixes at positions $i\Delta$-j, …, $i\Delta$-1
  - Let $T[k\ldots]$ be the $(i+1)$-th smallest suffix, then $B[k] := T[(k-1) \bmod n]$
  - $\boldsymbol{B}$ is a substring of the BWT of T
- In each step, we insert n/Δ further characters into $\boldsymbol{B}$

Grand Strategy

- Build BWT of T in a total of Δ steps
  - Each step will take time $O(n/\Delta)$, so we need time $O(n)$ in total
- After step j, $\boldsymbol{B}$ contans the BWT of T with respect to suffixes at positions $i\Delta-j$, …, $i\Delta-1$
  - Let $T[k\ldots]$ be the $(i+1)$-th smallest suffix, then $B[k] := T[(k-1) \bmod n]$
  - $\boldsymbol{B}$ is a substring of the BWT of T
- In each step, we insert n/Δ further characters into $\boldsymbol{B}$
- Steps 1 and 2 were already combined

Preparation for Step $j+1$

Preparation for Step $j+1$

- While computing $\boldsymbol{B}$, also build two arrays for further steps:

Preparation for Step $j+1$

- While computing $\boldsymbol{B}$, also build two arrays for further steps:
- For every suffix of T_j (metacharacters), store the $\boldsymbol{B}$-position of its first character in array $\boldsymbol{W}$
  - Space $O((n/\Delta) \lg n) = \boldsymbol{O(n \lg \sigma)}$

Preparation for Step $j+1$

- While computing $\boldsymbol{B}$, also build two arrays for further steps:
- For every suffix of T_j (metacharacters), store the $\boldsymbol{B}$-position of its first character in array $\boldsymbol{W}$
  - Space $O((n/\Delta) \lg n) = \boldsymbol{O(n \lg \sigma)}$
- $\boldsymbol{C}$ array of $\boldsymbol{B}$ (for every character, # of lex. smaller characters)
  - Space $O(\sigma \lg n) = \boldsymbol{O(n \lg \sigma)}$ if $\sigma = o(n)$

Preparation for Step 3

- For every suffix of T_2 (metacharacters), store the $\boldsymbol{B}$-position of its first character in array $\boldsymbol{W}$

Preparation for Step 3

- For every suffix of T_2 (metacharacters), store the $\boldsymbol{B}$-position of its first character in array $\boldsymbol{W}$

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
$SA(T_1T_2) =$	1,	13,	6,	17,	18,	4,	12,	3,	7,	11,	16,	8,	14,	9,	19,	20,	10,	15,	2,	5

$\boldsymbol{B}$ = GCGCCGGAAGGATGACGGAG

Preparation for Step 3

- For every suffix of T_2 (metacharacters), store the $\boldsymbol{B}$-position of its first character in array $\boldsymbol{W}$

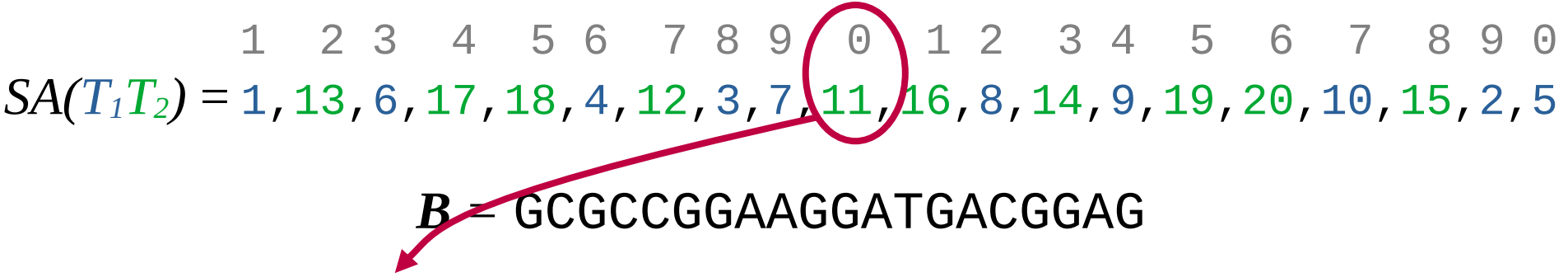
$$SA(T_1T_2) = \begin{array}{cccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ 1, & 13, & 6, & 17, & 18, & 4, & 12, & 3, & 7, & 11, & 16, & 8, & 14, & 9, & 19, & 20, & 10, & 15, & 2, & 5 \end{array}$$

$$\boldsymbol{B} = \texttt{GCGCCGGAAGGATGACGGAG}$$

$$\boldsymbol{W}[1]=10,\ \boldsymbol{W}[2]=7,\ \boldsymbol{W}[3]=2,\ \boldsymbol{W}[4]=13,\ \boldsymbol{W}[5]=18$$
$$\boldsymbol{W}[6]=11,\ \boldsymbol{W}[7]=4,\ \boldsymbol{W}[8]=5,\ \boldsymbol{W}[9]=15,\ \boldsymbol{W}[10]=16$$

Preparation for Step 3

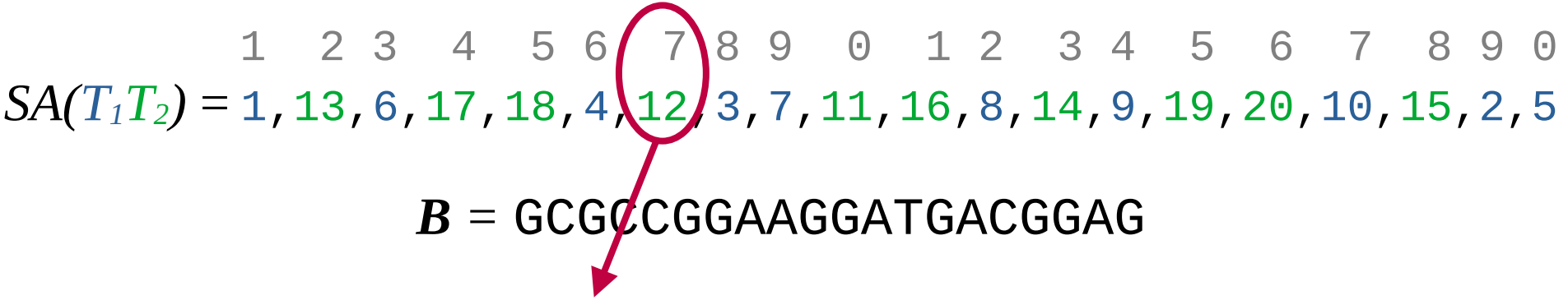
- For every suffix of T_2 (metacharacters), store the B-position of its first character in array W



$W[1]=10$, $W[2]=7$, $W[3]=2$, $W[4]=13$, $W[5]=18$
$W[6]=11$, $W[7]=4$, $W[8]=5$, $W[9]=15$, $W[10]=16$

Preparation for Step 3

- For every suffix of T_2 (metacharacters), store the $\boldsymbol{B}$-position of its first character in array $\boldsymbol{W}$



$\boldsymbol{W}[1]=10$, $\boldsymbol{W}[2]=7$, $\boldsymbol{W}[3]=2$, $\boldsymbol{W}[4]=13$, $\boldsymbol{W}[5]=18$
$\boldsymbol{W}[6]=11$, $\boldsymbol{W}[7]=4$, $\boldsymbol{W}[8]=5$, $\boldsymbol{W}[9]=15$, $\boldsymbol{W}[10]=16$

Preparation for Step 3

- For every suffix of T_2 (metacharacters), store the B-position of its first character in array W

B = GCGCCGGAAGGATGACGGAG

$W[1]=10$, $W[2]=7$, $W[3]=2$, $W[4]=13$, $W[5]=18$
$W[6]=11$, $W[7]=4$, $W[8]=5$, $W[9]=15$, $W[10]=16$

$C[\$]=0$, $C[A]=0$, $C[C]=5$, $C[G]=9$, $C[T]=19$

Step $j > 2$

Step $j > 2$

- After step j-1, $\boldsymbol{B}$ contans the BWT of T w.r.t. suffixes at positions $i\Delta$-j+1, …, $i\Delta$-1

Step $j > 2$

- After step j-1, $\boldsymbol{B}$ contans the BWT of T w.r.t. suffixes at positions $i\Delta$-j+1, …, $i\Delta$-1
- In step j, we insert suffixes $i\Delta$-j into $\boldsymbol{B}$
  - But where?

Step $j > 2$

Step $j > 2$

- Let S_{j-1} be the set of suffixes of T starting at positions $i\Delta-j+1$ to $i\Delta-2$
  - These have been inserted into $\boldsymbol{B}$ in steps 2 to $j-1$
  - $\boldsymbol{W}$ contains the positions of suffixes $i\Delta-j+1$ in $\boldsymbol{B}$

Step $j > 2$

- Let S_{j-1} be the set of suffixes of T starting at positions $i\Delta-j+1$ to $i\Delta-2$
  - These have been inserted into $\boldsymbol{B}$ in steps 2 to $j-1$
  - $\boldsymbol{W}$ contains the positions of suffixes $i\Delta-j+1$ in $\boldsymbol{B}$
- Let s be a suffix of T at position $i\Delta-j$ starting with character α

Step $j > 2$

- Let S_{j-1} be the set of suffixes of T starting at positions $i\Delta - j + 1$ to $i\Delta - 2$
  - These have been inserted into $\boldsymbol{B}$ in steps 2 to $j-1$
  - $\boldsymbol{W}$ contains the positions of suffixes $i\Delta - j + 1$ in $\boldsymbol{B}$
- Let s be a suffix of T at position $i\Delta - j$ starting with character α
- For every occurence of some $\alpha' < \alpha$, there is exactly one suffix $s' < s$ starting with α' in S_{j-1} (that's $C[\alpha]$ in total)
  - Because we begin with the Δ-th rotation of T and rotate to the right in each step

Step $j > 2$

- Let S_{j-1} be the set of suffixes of T starting at positions $i\Delta-j+1$ to $i\Delta-2$
  - These have been inserted into $\boldsymbol{B}$ in steps 2 to $j-1$
  - $\boldsymbol{W}$ contains the positions of suffixes $i\Delta-j+1$ in $\boldsymbol{B}$
- Let s be a suffix of T at position $i\Delta-j$ starting with character α
- For every occurence of some $\alpha' < \alpha$, there is exactly one suffix $s' < s$ starting with α' in S_{j-1} (that's $C[\alpha]$ in total)
  - Because we begin with the Δ-th rotation of T and rotate to the right in each step
- There are exactly $\text{rank}_{\alpha}(\boldsymbol{B}, \boldsymbol{W}[i])$ suffixes $s'' \leq s$ in S_{j-1} that start with α
  - Because the BWT is order-preserving

Step $j > 2$

- Let S_{j-1} be the set of suffixes of T starting at positions $i\Delta - j + 1$ to $i\Delta - 2$
  - These have been inserted into $\boldsymbol{B}$ in steps 2 to $j-1$
  - $\boldsymbol{W}$ contains the positions of suffixes $i\Delta - j + 1$ in $\boldsymbol{B}$
- Let s be a suffix of T at position $i\Delta - j$ starting with character α
- For every occurence of some $\alpha' < \alpha$, there is exactly one suffix $s' < s$ starting with α' in S_{j-1} (that's $C[\alpha]$ in total)
  - Because we begin with the Δ-th rotation of T and rotate to the right in each step
- There are exactly $\text{rank}_\alpha(\boldsymbol{B}, \boldsymbol{W}[i])$ suffixes $s'' \leq s$ in S_{j-1} that start with α
  - Because the BWT is order-preserving
- Thus, $C[\alpha] + \text{rank}_\alpha(\boldsymbol{B}, \boldsymbol{W}[i])$ suffixes from S_{j-1} precede s in $\boldsymbol{B}$

Step $j > 2$

Step $j > 2$

- … Thus, $C[\alpha] + \text{rank}_{\alpha}(\boldsymbol{B}, W[i])$ suffixes from S_{j-1} precede s in $\boldsymbol{B}$

Step $j > 2$

- … Thus, $C[\alpha] + \text{rank}_{\alpha}(B, W[i])$ suffixes from S_{j-1} precede s in B
- Consider $SA(T_1T_j)$ and let c_i be the # of T_1 suffixes appearing before suffix $i\Delta$-j in $SA(T_1T_j)$

Step $j > 2$

- … Thus, $C[\alpha] + \text{rank}_{\alpha}(\boldsymbol{B}, W[i])$ suffixes from S_{j-1} precede s in $\boldsymbol{B}$
- Consider $SA(T_1T_j)$ and let c_i be the # of T_1 suffixes appearing before suffix $i\Delta$-j in $SA(T_1T_j)$
- These also precede s in $\boldsymbol{B}$ and have not yet been considered

Step $j > 2$

- … Thus, $C[\alpha] + \text{rank}_{\alpha}(\boldsymbol{B}, W[i])$ suffixes from S_{j-1} precede s in $\boldsymbol{B}$
- Consider $SA(T_1T_j)$ and let c_i be the # of T_1 suffixes appearing before suffix $i\Delta\text{-}j$ in $SA(T_1T_j)$
- These also precede s in $\boldsymbol{B}$ and have not yet been considered
- In step j, we insert $T[i\Delta\text{-}j]$ at position $C[\alpha] + \text{rank}_{\alpha}(\boldsymbol{B}, W[i]) + c_i$

Example: Step 3

```
                  12345678901234567890123456789 0
T  = GATCAATGAGGTGGACACCAGAGGCGGGG$

                   1   2   3   4   5   6   7   8   9   0
T1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG
                   1   2   3   4   5   6   7   8   9   0
T3 = GG$ GAT CAA TGA GGT GGA CAC CAG AGG CGG
```

$SA(T_1T_3) = 1,6,19,4,3,13,17,18,7,20,8,12,9,11,16,10,15,2,14,5$

```
                  12345678901234567890
B  = GCGCCGGAAGGATGACGGAG
```

Example: Step 3

1234567890123456789012345678 90

T = GATCAATGAGGTGGACACCAGAGGCGGGG$

1 2 3 4 5 6 7 8 9 0

T_1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG

1 2 3 4 5 6 7 8 9 0

T_3 = GG$ GAT CAA TGA GGT GGA CAC CAG AGG CGG

$SA(T_1T_3)$ = 1,6,19,4,3,13,17,18,7,20,8,12,9,11,16,10,15,2,14,5

123456789012345678 90

$\boldsymbol{B}$ = GCGCCGGAAGGATGACGGAG

- Insert G at position $C[G] + \text{rank}_G(\boldsymbol{B}, W[1]) + c_1 = 9 + 5 + 7 = \mathbf{21}$ ($W[1] = 10$)

Example: Step 3

1234567890123456789012345678 90

T = GATCAATGAGGTGGACACCAGAGGCGGGG$

1 2 3 4 5 6 7 8 9 0

T_1 = $GA TCA ATG AGG TGG ACA CCA GAG GCG GGG

1 2 3 4 5 6 7 8 9 0

T_3 = GG$ GAT CAA TGA GGT GGA CAC CAG AGG CGG

$SA(T_1T_3)$ = 1,6,19,4,3,13,17,18,7,20,8,12,9,11,16,10,15,2,14,5

12345678901234567890

$\boldsymbol{B}$ = GCGCCGGAAGGATGACGGAG

- Insert $ at position $C[G] + \text{rank}_G(\boldsymbol{B}, W[2]) + c_2 = 9 + 4 + 6 = 19$

Example: Step 3

1234567890123456789 0

B = GCGCCGGAAGGATGACGGAG

- Insert **G** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[1]) + c_1 = 9 + 5 + 7 = \mathbf{21}$
- Insert **$** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[2]) + c_2 = 9 + 4 + 6 = \mathbf{19}$
- Insert **T** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[3]) + c_3 = 5 + 1 + 4 = \mathbf{10}$
- Insert **A** at position $C[\mathbf{T}] + \text{rank}_{\mathbf{T}}(B, W[4]) + c_4 = 19 + 1 + 9 = \mathbf{29}$
- Insert **A** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[5]) + c_5 = 9 + 9 + 8 = \mathbf{26}$
- Insert **T** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[6]) + c_6 = 9 + 6 + 7 = \mathbf{22}$
- Insert **A** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[7]) + c_7 = 5 + 2 + 4 = \mathbf{11}$
- Insert **C** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[8]) + c_8 = 5 + 3 + 4 = \mathbf{12}$
- Insert **G** at position $C[\mathbf{A}] + \text{rank}_{\mathbf{A}}(B, W[9]) + c_9 = 0 + 4 + 2 = \mathbf{6}$
- Insert **G** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[10]) + c_{10} = 5 + 4 + 5 = \mathbf{14}$

Example: Step 3

```
1234567890123456789012345678 90
B = GCGCC-GGA---A-GGAT-G--AGC-GA-G
```

- Insert **G** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[1]) + c_1 = 9 + 5 + 7 = \mathbf{21}$
- Insert **$** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[2]) + c_2 = 9 + 4 + 6 = \mathbf{19}$
- Insert **T** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[3]) + c_3 = 5 + 1 + 4 = \mathbf{10}$
- Insert **A** at position $C[\mathbf{T}] + \text{rank}_{\mathbf{T}}(B, W[4]) + c_4 = 19 + 1 + 9 = \mathbf{29}$
- Insert **A** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[5]) + c_5 = 9 + 9 + 8 = \mathbf{26}$
- Insert **T** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[6]) + c_6 = 9 + 6 + 7 = \mathbf{22}$
- Insert **A** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[7]) + c_7 = 5 + 2 + 4 = \mathbf{11}$
- Insert **C** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[8]) + c_8 = 5 + 3 + 4 = \mathbf{12}$
- Insert **G** at position $C[\mathbf{A}] + \text{rank}_{\mathbf{A}}(B, W[9]) + c_9 = 0 + 4 + 2 = \mathbf{6}$
- Insert **G** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[10]) + c_{10} = 5 + 4 + 5 = \mathbf{14}$

Example: Step 3

1234567890123456789012345678 90

B = GCGCCGGGATACAGGGAT$GGTAGCAGAAG

- Insert G at position $C[G] + \text{rank}_G(B, W[1]) + c_1 = 9 + 5 + 7 = \mathbf{21}$
- Insert $ at position $C[G] + \text{rank}_G(B, W[2]) + c_2 = 9 + 4 + 6 = \mathbf{19}$
- Insert T at position $C[C] + \text{rank}_C(B, W[3]) + c_3 = 5 + 1 + 4 = \mathbf{10}$
- Insert A at position $C[T] + \text{rank}_T(B, W[4]) + c_4 = 19 + 1 + 9 = \mathbf{29}$
- Insert A at position $C[G] + \text{rank}_G(B, W[5]) + c_5 = 9 + 9 + 8 = \mathbf{26}$
- Insert T at position $C[G] + \text{rank}_G(B, W[6]) + c_6 = 9 + 6 + 7 = \mathbf{22}$
- Insert A at position $C[C] + \text{rank}_C(B, W[7]) + c_7 = 5 + 2 + 4 = \mathbf{11}$
- Insert C at position $C[C] + \text{rank}_C(B, W[8]) + c_8 = 5 + 3 + 4 = \mathbf{12}$
- Insert G at position $C[A] + \text{rank}_A(B, W[9]) + c_9 = 0 + 4 + 2 = \mathbf{6}$
- Insert G at position $C[C] + \text{rank}_C(B, W[10]) + c_{10} = 5 + 4 + 5 = \mathbf{14}$

Example: Step 3

```
12345678901234567890123456789 0
B = GCGCCGGGATACAGGGAT$GGTAGCAGAAG
```

- Insert **G** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[1]) + c_1 = 9 + 5 + 7 = \mathbf{21}$
- Insert **$** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[2]) + c_2 = 9 + 4 + 6 = \mathbf{19}$
- Insert **T** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[3]) + c_3 = 5 + 1 + 4 = \mathbf{10}$
- Insert **A** at position $C[\mathbf{T}] + \text{rank}_{\mathbf{T}}(B, W[4]) + c_4 = 19 + 1 + 9 = \mathbf{29}$
- Insert **A** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[5]) + c_5 = 9 + 9 + 8 = \mathbf{26}$
- Insert **T** at position $C[\mathbf{G}] + \text{rank}_{\mathbf{G}}(B, W[6]) + c_6 = 9 + 6 + 7 = \mathbf{22}$
- Insert **A** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[7]) + c_7 = 5 + 2 + 4 = \mathbf{11}$
- Insert **C** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[8]) + c_8 = 5 + 3 + 4 = \mathbf{12}$
- Insert **G** at position $C[\mathbf{A}] + \text{rank}_{\mathbf{A}}(B, W[9]) + c_9 = 0 + 4 + 2 = \mathbf{6}$
- Insert **G** at position $C[\mathbf{C}] + \text{rank}_{\mathbf{C}}(B, W[10]) + c_{10} = 5 + 4 + 5 = \mathbf{14}$

n/Δ rank queries must not exceed $O(1)$ time per query (amortized)!

Data Structure for Batched Rank Queries

```
          12345678901234567890
B = GCGCCGGAAGGATGACGGAG
```

Data Structure for Batched Rank Queries

$$\boldsymbol{B} = \begin{array}{|cccc|cccc|cccc|cccc|cccc|}\hline 1&2&3&4&5&6&7&8&9&0&1&2&3&4&5&6&7&8&9&0\\ \hline G&C&G&C&C&G&G&A&A&G&G&A&T&G&A&C&G&G&A&G\\ \hline\end{array}$$

- Divide $\boldsymbol{B}$ into chunks C_1, …, C_g of size σ

Data Structure for Batched Rank Queries

$\boldsymbol{B}$ =

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
G	C	G	C	C	G	G	A	A	G	G	A	T	G	A	C	G	G	A	G

M_A = 0010110101
M_C = 011010010
M_G = 01101101101011 1
M_T = 000010

- Divide $\boldsymbol{B}$ into chunks C_1, …, C_g of size σ
- For each character $\boldsymbol{c}$, store $M_c = 01^{d(1)}\ldots01^{d(g)}$ with select support
  - With $d(i)$ the # of occurences of $\boldsymbol{c}$ in C_i

Data Structure for Batched Rank Queries

$$\boldsymbol{B} = \begin{array}{|cccc|cccc|cccc|cccc|cccc|} \hline 1&2&3&4&5&6&7&8&9&0&1&2&3&4&5&6&7&8&9&0 \\ \hline G&C&G&C&C&G&G&A&A&G&G&A&T&G&A&C&G&G&A&G \\ \hline \end{array}$$

$M_A = 0010110101$
$M_C = 011010010$
$M_G = 01101101101011 1$
$M_T = 000010$

$R_1 = (C,2,1)(C,4,2)(G,1,1)(G,3,2)$
$R_2 = (A,4,1)(C,1,1)(G,2,1)(G,3,2)$
$R_3 = (A,1,1)(A,4,2)(G,2,1)(G,3,2)$
$R_4 = (A,3,1)(C,4,1)(G,2,1)(T,1,1)$
$R_5 = (A,3,1)(G,1,1)(G,2,1)(G,4,3)$

- Divide $\boldsymbol{B}$ into chunks $C_1, \ldots, C_g$ of size σ
- For each character c, store $M_c = 01^{d(1)}\ldots01^{d(g)}$ with select support
  - With $d(i)$ the # of occurences of c in C_i
- For each $C_i[j] = c$, store a triplet $(c, j, rank_c(C, j))$ in R_i
  - Sort triplets according by c and j