Karlsruhe Institute of Technology

# Distributed Suffix Sorting

## Fast and Lightweight Suffix Array Construction

Master Thesis · April 2, 2025
Manuel Haag

# Suffix Sorting

input text: `banana$`

| | |
|---|---|
| 0 | banana$ |
| 1 | anana$ |
| 2 | nana$ |
| 3 | ana$ |
| 4 | na$ |
| 5 | a$ |
| 6 | $ |

# Suffix Sorting

input text: banana$

| | |
|---|---|
| 0 | banana$ |
| 1 | anana$ |
| 2 | nana$ |
| 3 | ana$ |
| 4 | na$ |
| 5 | a$ |
| 6 | $ |


| | |
|---|---|
| 6 | $ |
| 5 | a$ |
| 3 | ana$ |
| 1 | anana$ |
| 0 | banana$ |
| 4 | na$ |
| 2 | nana$ |

# Suffix Sorting

Suffix Array (SA)

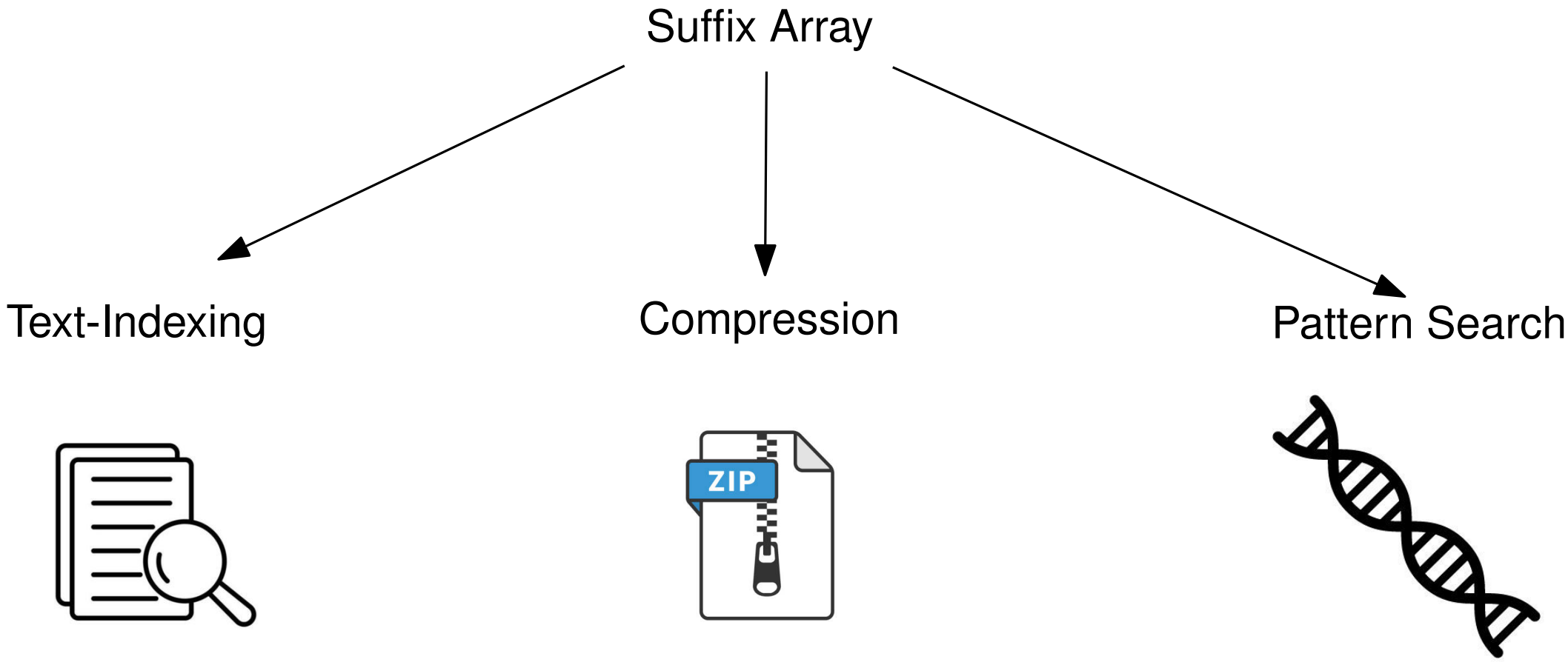
input text: `banana$`

| | |
|---|---|
| 0 | banana$ |
| 1 | anana$ |
| 2 | nana$ |
| 3 | ana$ |
| 4 | na$ |
| 5 | a$ |
| 6 | $ |

suffix sorting

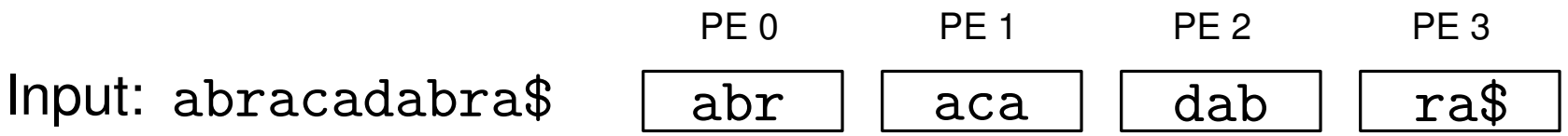
| | |
|---|---|
| 6 | $ |
| 5 | a$ |
| 3 | ana$ |
| 1 | anana$ |
| 0 | banana$ |
| 4 | na$ |
| 2 | nana$ |

# Applications of the Suffix Array



# Distributed Memory Model

- input is distributed over $p$ processing elements (PEs):




- PEs communicate via message passing over network:

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

**inducing** [Itoh & Tanaka 1999]

**recursion** [Kärkkäinen and Sanders 2003]

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

**inducing** [Itoh & Tanaka 1999]

**recursion** [Kärkkäinen and Sanders 2003]

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

- PSAC [Flick & Aluru 2015]
- dPD [Kurpicz & Fischer 2019]

**inducing** [Itoh & Tanaka 1999]

**recursion** [Kärkkäinen and Sanders 2003]

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

**inducing** [Itoh & Tanaka 1999]

$T \xrightarrow{\text{classify}} C_1, C_2 \xrightarrow{\text{sort}} SA_1 \xrightarrow{\text{infer}} SA$

$\mathcal{O}(n)$

**recursion** [Kärkkäinen and Sanders 2003]

- PSAC [Flick & Aluru 2015]
- dPD [Kurpicz & Fischer 2019]

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

- PSAC [Flick & Aluru 2015]
- dPD [Kurpicz & Fischer 2019]

**inducing** [Itoh & Tanaka 1999]

$T \xrightarrow{\text{classify}} C_1, C_2 \xrightarrow{\text{sort}} SA_1 \xrightarrow{\text{infer}} SA$

$\mathcal{O}(n)$

- dDivSuffSort [Kurpicz & Fischer 2019]

**recursion** [Kärkkäinen and Sanders 2003]

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

- PSAC [Flick & Aluru 2015]
- dPD [Kurpicz & Fischer 2019]

**inducing** [Itoh & Tanaka 1999]

$T \xrightarrow{\text{classify}} C_1, C_2 \xrightarrow{\text{sort}} SA_1 \xrightarrow{\text{infer}} SA$

$\mathcal{O}(n)$

- dDivSuffSort [Kurpicz & Fischer 2019]

**recursion** [Kärkkäinen and Sanders 2003]

$T \xrightarrow{\text{classify}} T_1, T_2 \xrightarrow{\text{recursion}} SA_1 \xrightarrow{\text{infer}} SA_2 \xrightarrow{\text{merge}} SA$

$\mathcal{O}(n)$

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

- PSAC [Flick & Aluru 2015]
- dPD [Kurpicz & Fischer 2019]

**inducing** [Itoh & Tanaka 1999]

$T \xrightarrow{\text{classify}} C_1, C_2 \xrightarrow{\text{sort}} SA_1 \xrightarrow{\text{infer}} SA$

$\mathcal{O}(n)$

- dDivSuffSort [Kurpicz & Fischer 2019]

**recursion** [Kärkkäinen and Sanders 2003]

$T \xrightarrow{\text{classify}} T_1, T_2 \xrightarrow{\text{recursion}} SA_1 \xrightarrow{\text{infer}} SA_2 \xrightarrow{\text{merge}} SA$

$\mathcal{O}(n)$

- pDC3 [Kulla & Sanders 2007]

# Related Work - Distributed Suffix Sorting

**prefix doubling** [Manber & Myers 1990]

$SA_1 \longrightarrow SA_2 \longrightarrow SA_4 \ldots$

$\mathcal{O}(n \log n)$

- PSAC [Flick & Aluru 2015]
- dPD [Kurpicz & Fischer 2019]

**inducing** [Itoh & Tanaka 1999]

$T \xrightarrow{\text{classify}} C_1, C_2 \xrightarrow{\text{sort}} SA_1 \xrightarrow{\text{infer}} SA$

$\mathcal{O}(n)$

- dDivSuffSort [Kurpicz & Fischer 2019]

**recursion** [Kärkkäinen and Sanders 2003]

$T \xrightarrow{\text{classify}} T_1, T_2 \xrightarrow{\text{recursion}} SA_1 \xrightarrow{\text{infer}} SA_2 \xrightarrow{\text{merge}} SA$

$\mathcal{O}(n)$

- pDC3 [Kulla & Sanders 2007]

basis for our algorithm

# Constribution

- space-efficient implementation of DCX [Kärkkäinen et al 2006] with MPI + KaMPIng
- new load-balancing scheme chunking for overlapping strings
- speedups of up to 3.2× (6144 PEs)
- 3× more memory-efficient than competitor

# Difference Cover Modulo 3 (DC3) - Basic Idea

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

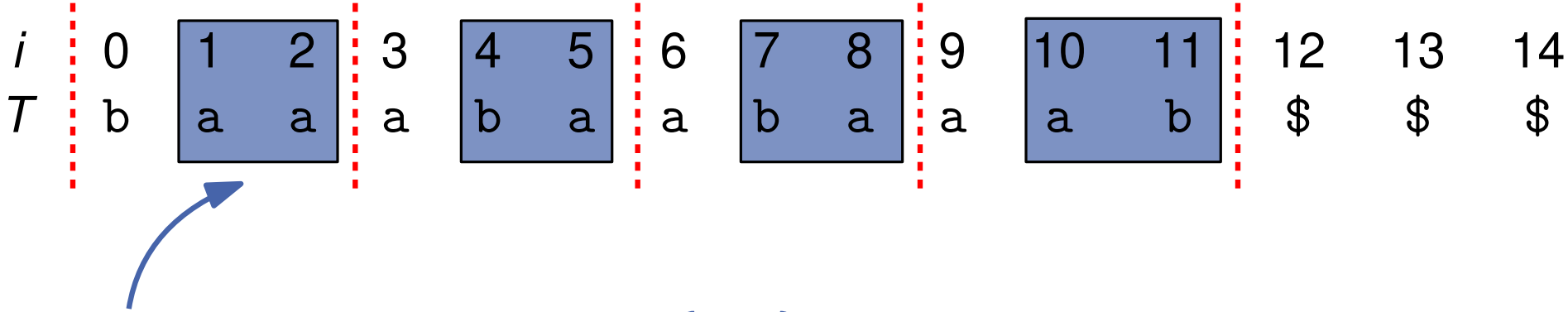
# Difference Cover Modulo 3 (DC3) - Basic Idea

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

**Definition**

A set $D_X$ is a difference cover modulo $X$ if $\{(i-j) \bmod X \mid i,j \in D_X\} = [0,X)$.

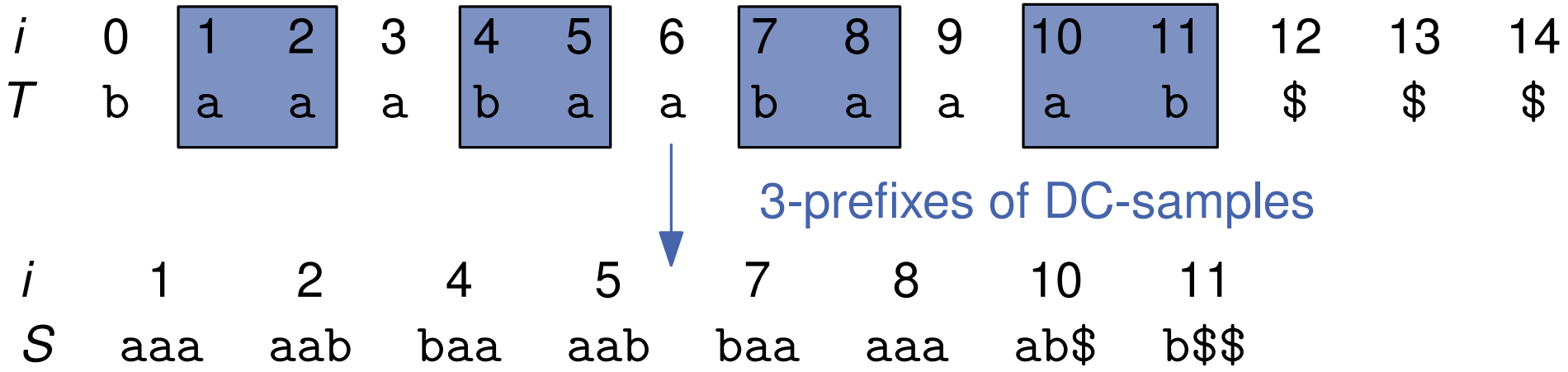
# Difference Cover Modulo 3 (DC3) - Basic Idea



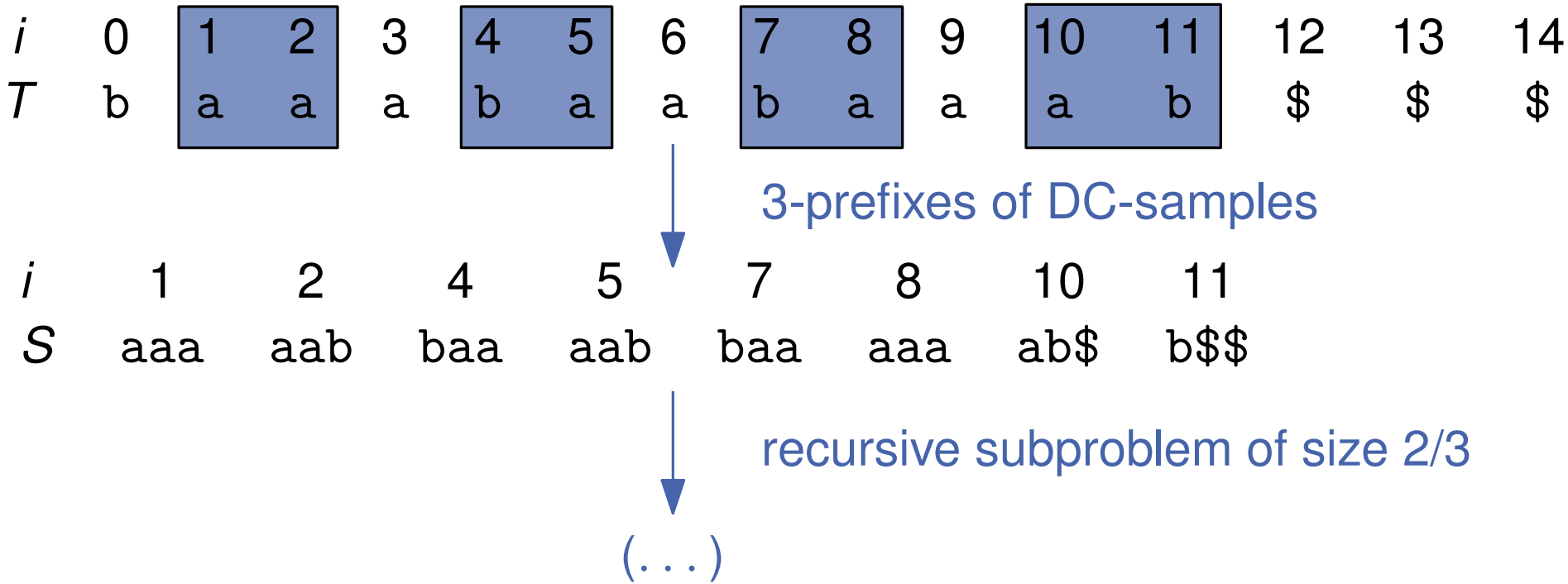

**Definition**

A set $D_X$ is a difference cover modulo $X$ if $\{(i - j) \bmod X \mid i, j \in D_X\} = [0, X)$.

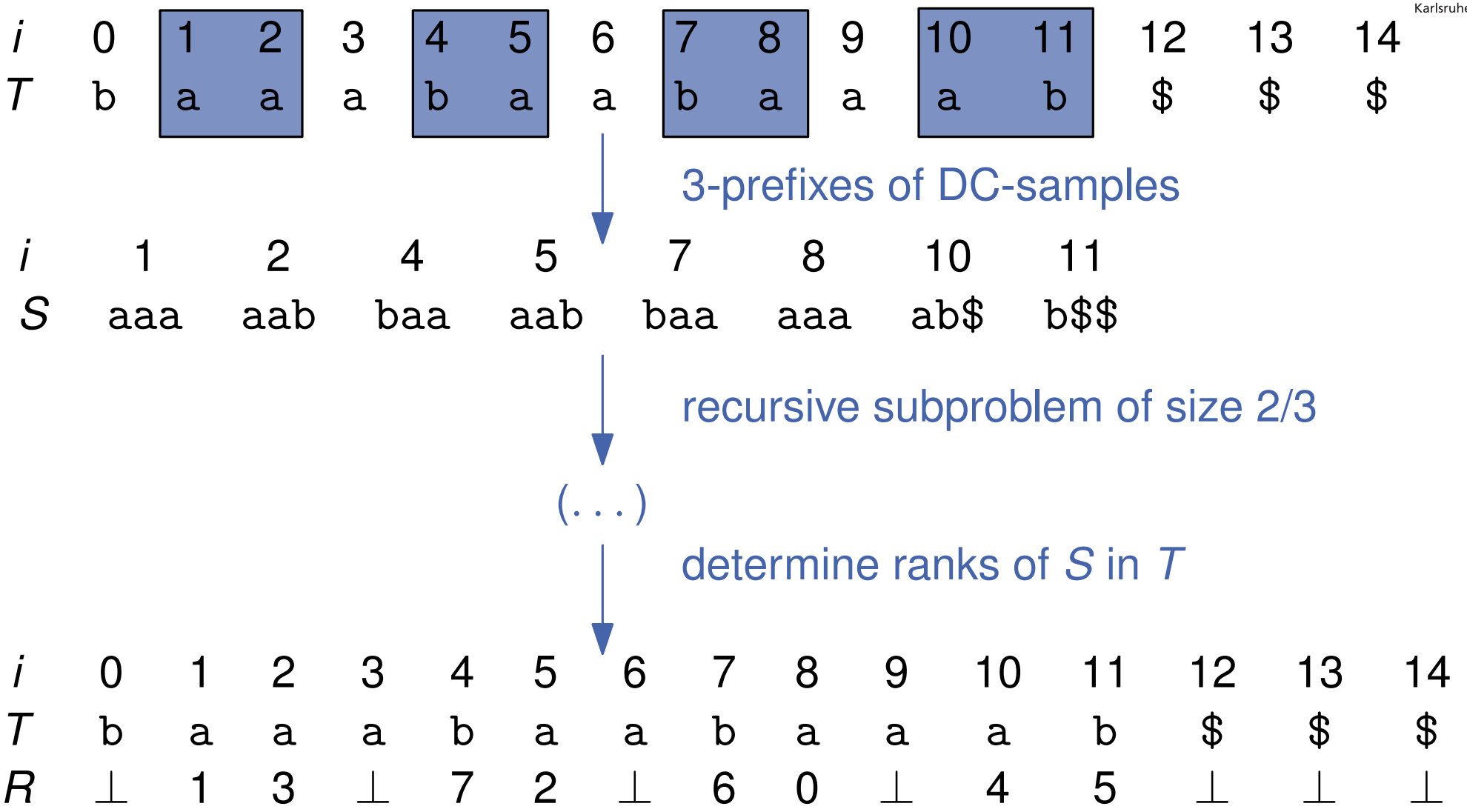
# Difference Cover Modulo 3 (DC3) - Basic Idea



# Difference Cover Modulo 3 (DC3) - Basic Idea



# Difference Cover Modulo 3 (DC3) - Basic Idea



# Difference Cover Modulo 3 (DC3) - Basic Idea

DC-samples ranks are now determined.

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison:

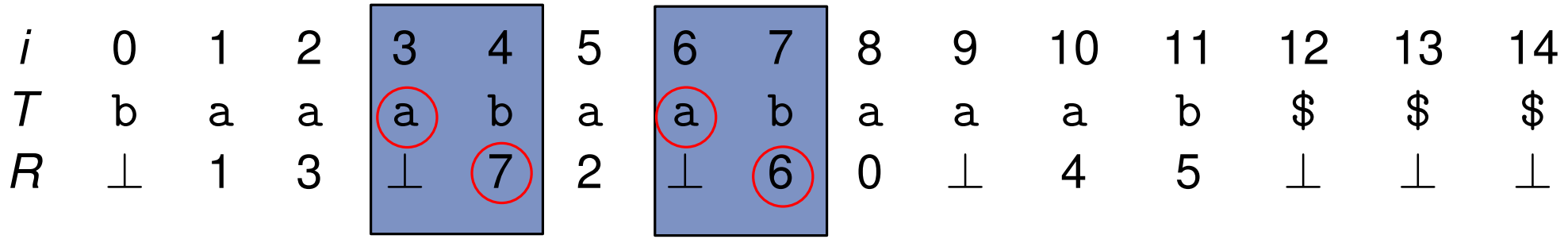
| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (DC3) - Basic Idea

DC-samples ranks are now determined.

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison: $S_3 > S_6$



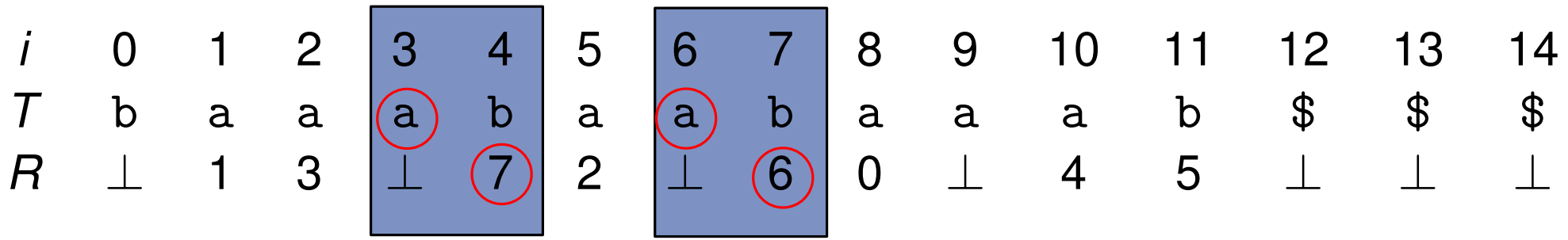
# Difference Cover Modulo 3 (DC3) - Basic Idea

DC-samples ranks are now determined.

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison: $S_3 > S_6$

$\leq$ 3 comparisons



# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed Memory

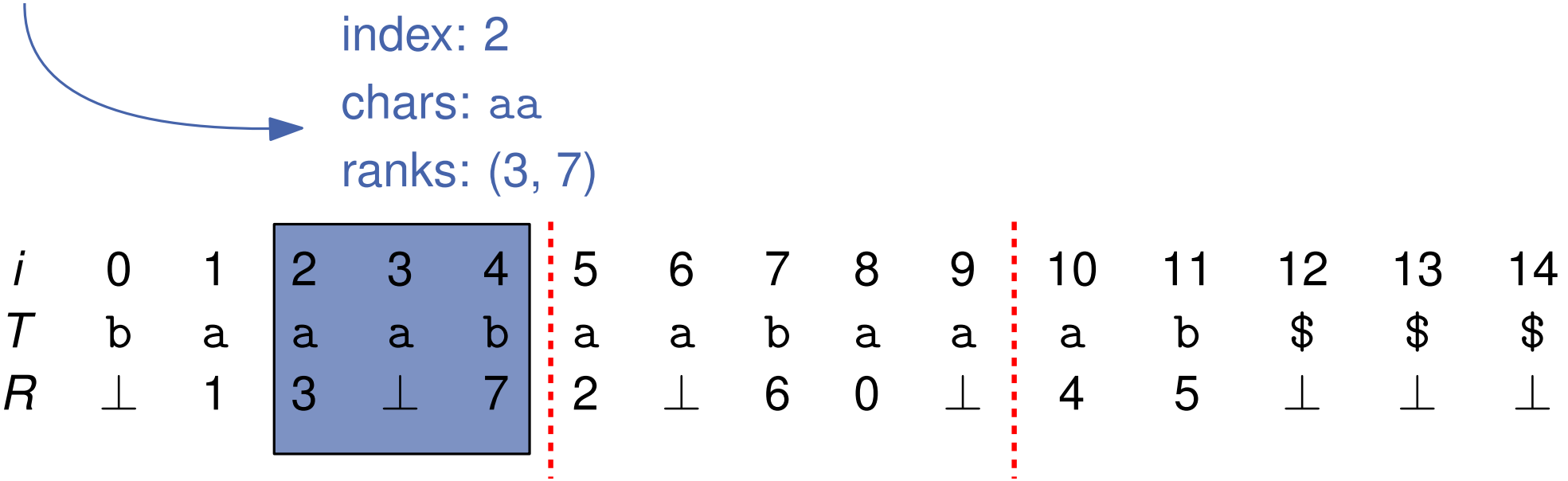
- distributed sorting routines
- shifts characters/ranks between adjacent PEs
- materialize chars and prefixes to send none-local data

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed Memory

- distributed sorting routines
- shifts characters/ranks between adjacent PEs
- materialize chars and prefixes to send none-local data



# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed Memory

- distributed sorting routines
- shifts characters/ranks between adjacent PEs
- materialize chars and prefixes to send none-local data

index: 2
chars: aa
ranks: (3, 7)

memory blowup by 3×

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow X$-prefixes
- size of recursive subproblem:

  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow X$-prefixes $\longrightarrow$ memory blowup by $X\times$
- size of recursive subproblem:

  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow X$-prefixes → memory blowup by $X\times$
- size of recursive subproblem:
  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$ → smaller subproblem

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (DC3) - Basic Idea

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow$ $X$-prefixes
- size of recursive subproblem:
  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$

→ memory blowup by $X\times$

→ smaller subproblem

need for space-efficient implementation

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

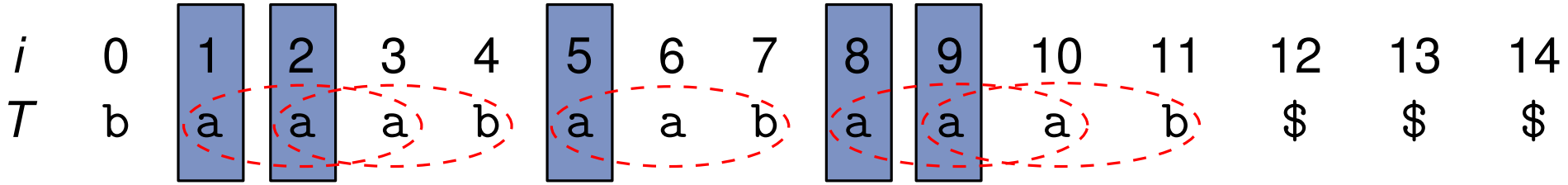
## Sorting 3-prefixes with 3 buckets

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

**splitters**: ab$, b$$

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

## Sorting 3-prefixes with 3 buckets

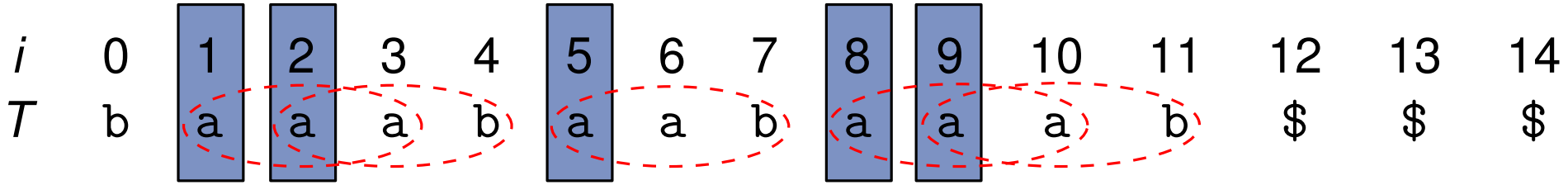



**splitters**: ab$, b$$

Bucket 0: (1, aaa), (2, aab), (5, aab), (8, aaa), (9, aab) $<$ ab$

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

## Sorting 3-prefixes with 3 buckets




**splitters**: ab$, b$$

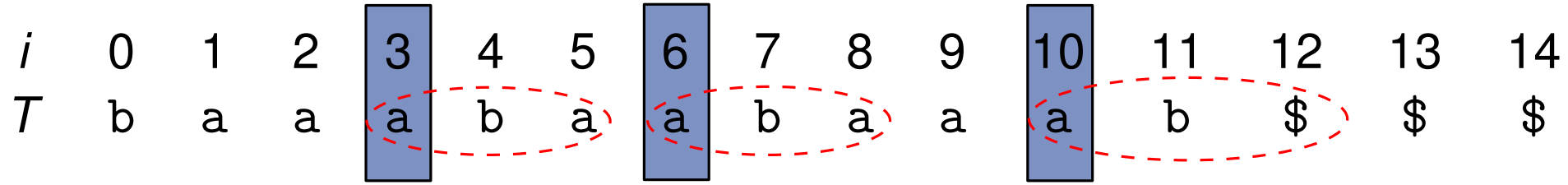
Bucket 0: (1, aaa), (2, aab), (5, aab), (8, aaa), (9, aab) $<$ ab$

**sorted order:** 1, 8, 2, 5, 9

Bucket 0

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

## Sorting 3-prefixes with 3 buckets




**splitters**: ab$, b$$

Bucket 0: (1, aaa), (2, aab), (5, aab), (8, aaa), (9, aab) $< ab\$$

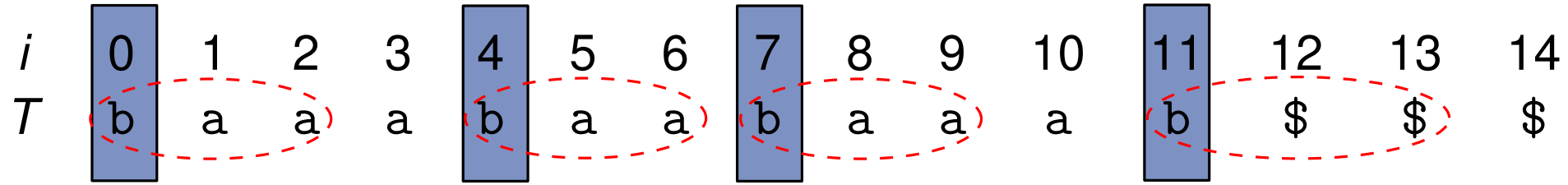
Bucket 1: (3, aba), (6, aba), (10, ab$) $< b\$\$$

**sorted order:** 1, 8, 2, 5, 9 | 10, 3, 6

Bucket 0 | Bucket 1

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

## Sorting 3-prefixes with 3 buckets




**splitters**: ab$, b$$

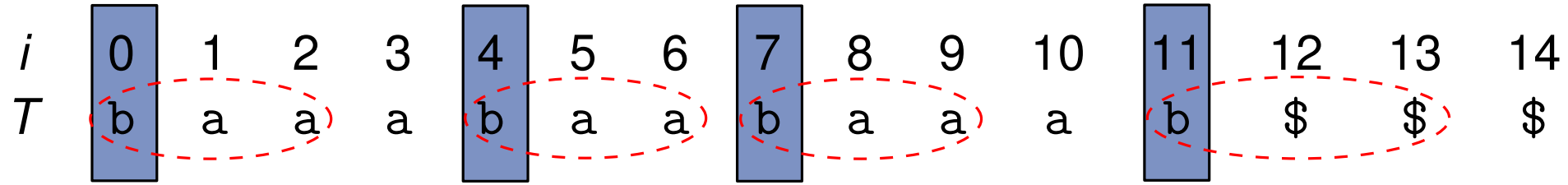
| | |
|---|---|
| Bucket 0: (1, aaa), (2, aab), (5, aab), (8, aaa), (9, aab) | $<$ ab$ |
| Bucket 1: (3, aba), (6, aba), (10, ab$) | $<$ b$$ |
| Bucket 2: (0, baa), (4, baa), (7, baa), (11, b$$) | $\geq$ b$$ |

**sorted order:** 1, 8, 2, 5, 9 | 10, 3, 6 | 11, 0, 4, 7

Bucket 0 | Bucket 1 | Bucket 2

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

## Sorting 3-prefixes with 3 buckets




**splitters**: ab$, b$$

| | |
|---|---|
| Bucket 0: (1, aaa), (2, aab), (5, aab), (8, aaa), (9, aab) | $<$ ab$ |
| Bucket 1: (3, aba), (6, aba), (10, ab$) | $<$ b$$ |
| Bucket 2: (0, baa), (4, baa), (7, baa), (11, b$$) | $\geq$ b$$ |

**sorted order:** 1, 8, 2, 5, 9 (Bucket 0) | 10, 3, 6 (Bucket 1) | 11, 0, 4, 7 (Bucket 2)

$\approx$ 1/3 memory consumption

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

Karlsruhe Institute of Technology

**Problem:** imbalances within a bucket

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

**Problem:** imbalances within a bucket

| | PE 0 | | | | | PE 1 | | | | | PE 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| $T$ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

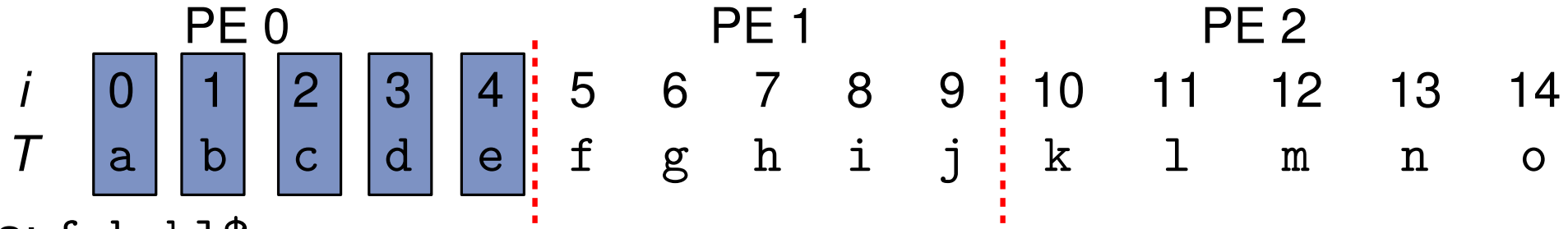
**Problem:** imbalances within a bucket

| | PE 0 | | | | | PE 1 | | | | | PE 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| $T$ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |

**splitters**: `fgh`, `kl$`

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

**Problem:** imbalances within a bucket

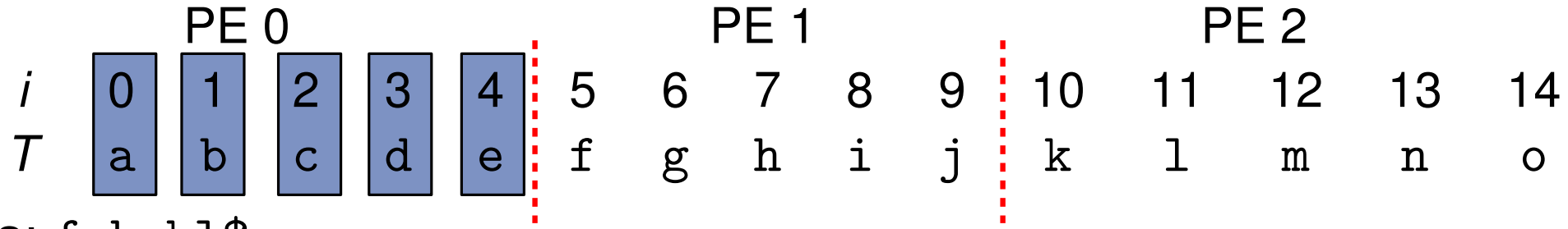



**splitters**: `fgh`, `kl$`

Bucket 0: (0, `abc`), (1, `bcd`), (2, `cde`), (3, `def`), (4, `efg`),

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

**Problem:** imbalances within a bucket




**splitters**: `fgh`, `kl$`

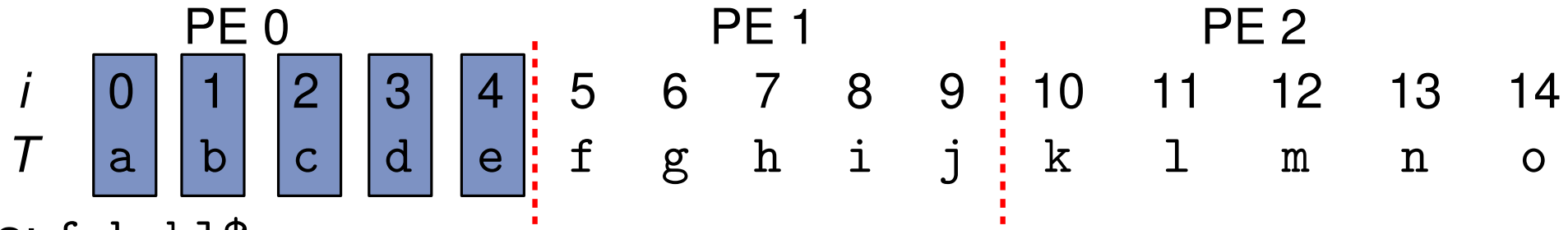
Bucket 0: (0, `abc`), (1, `bcd`), (2, `cde`), (3, `def`), (4, `efg`),

→ completly on PE 0

PE 1 and PE 2 are empty

# Space-Efficient Sorting Using Bucketing [Mehnert 2024]

**Problem:** imbalances within a bucket




**splitters**: `fgh`, `kl$`

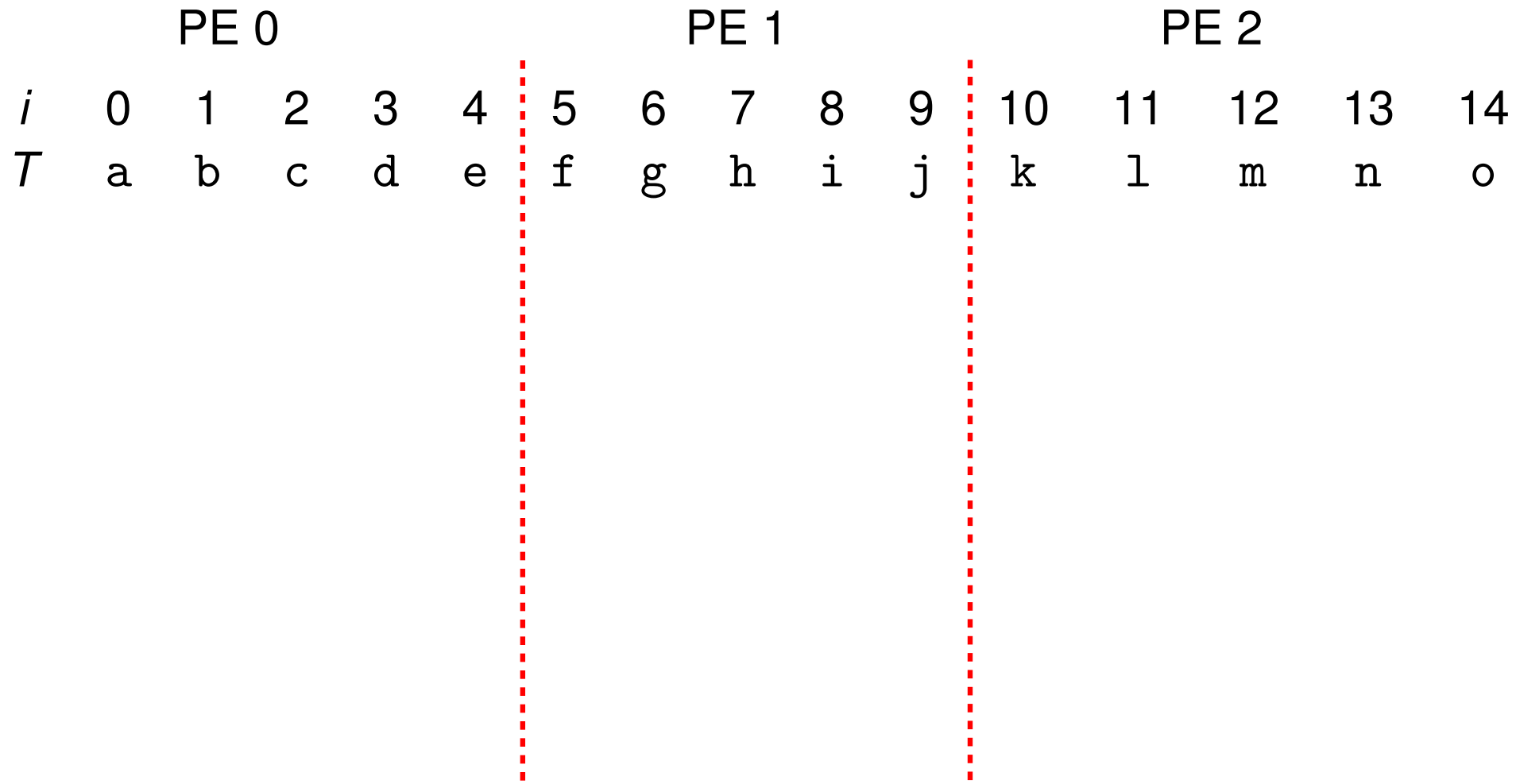
Bucket 0: (0, `abc`), (1, `bcd`), (2, `cde`), (3, `def`), (4, `efg`),

completly on PE 0

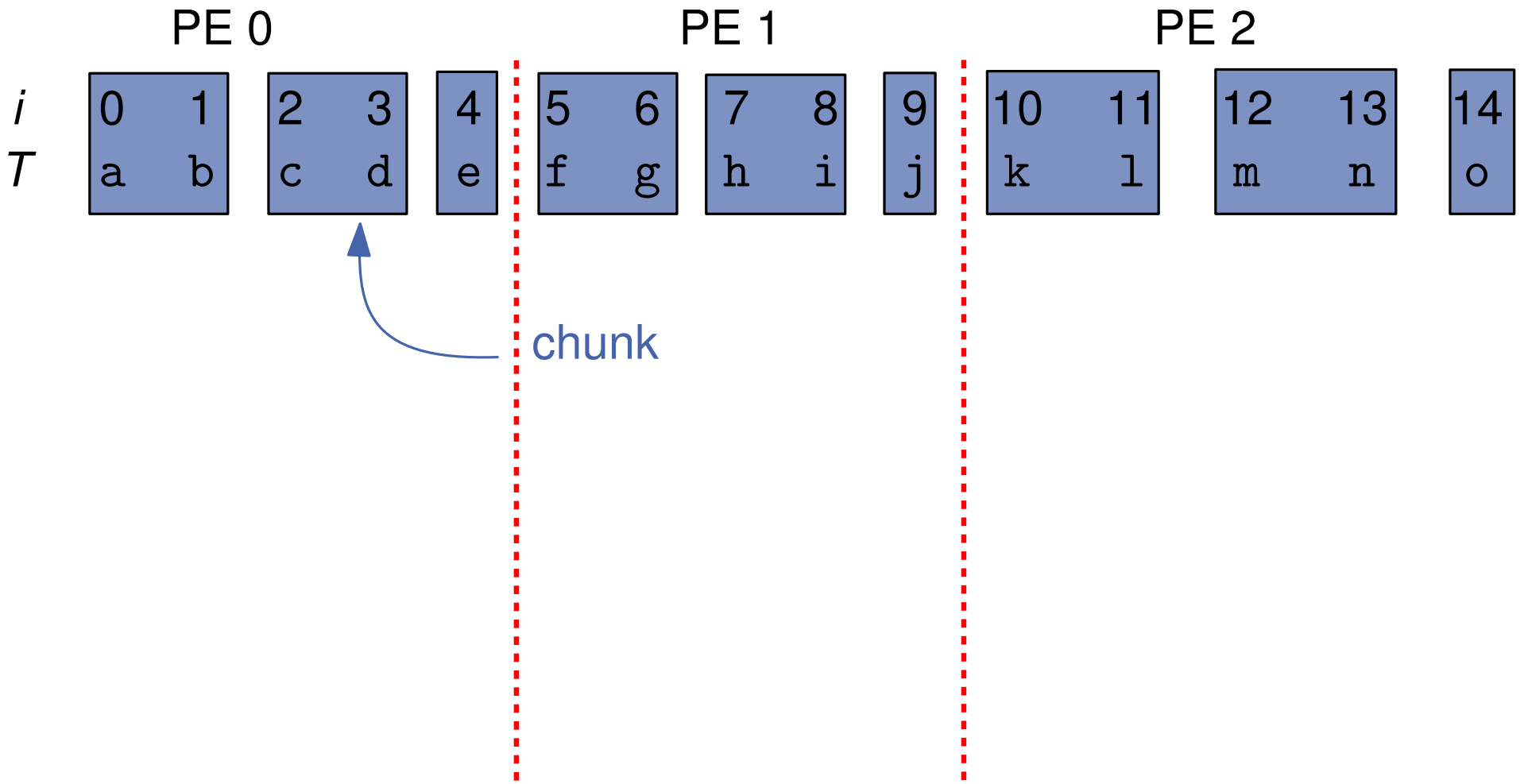
PE 1 and PE 2 are empty

**Solution:** random redistribution of chunks

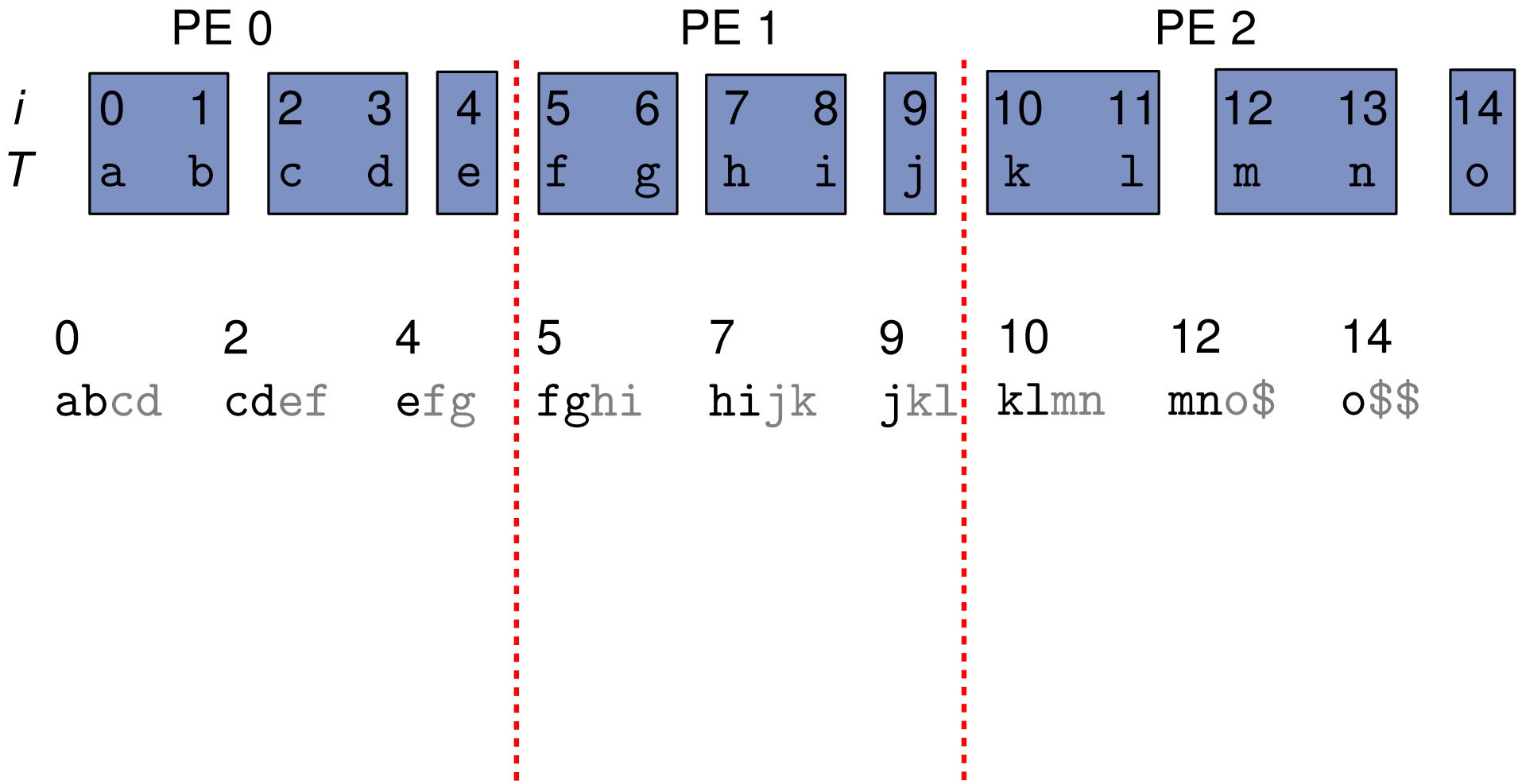
# Random Redistribution Using Chunking



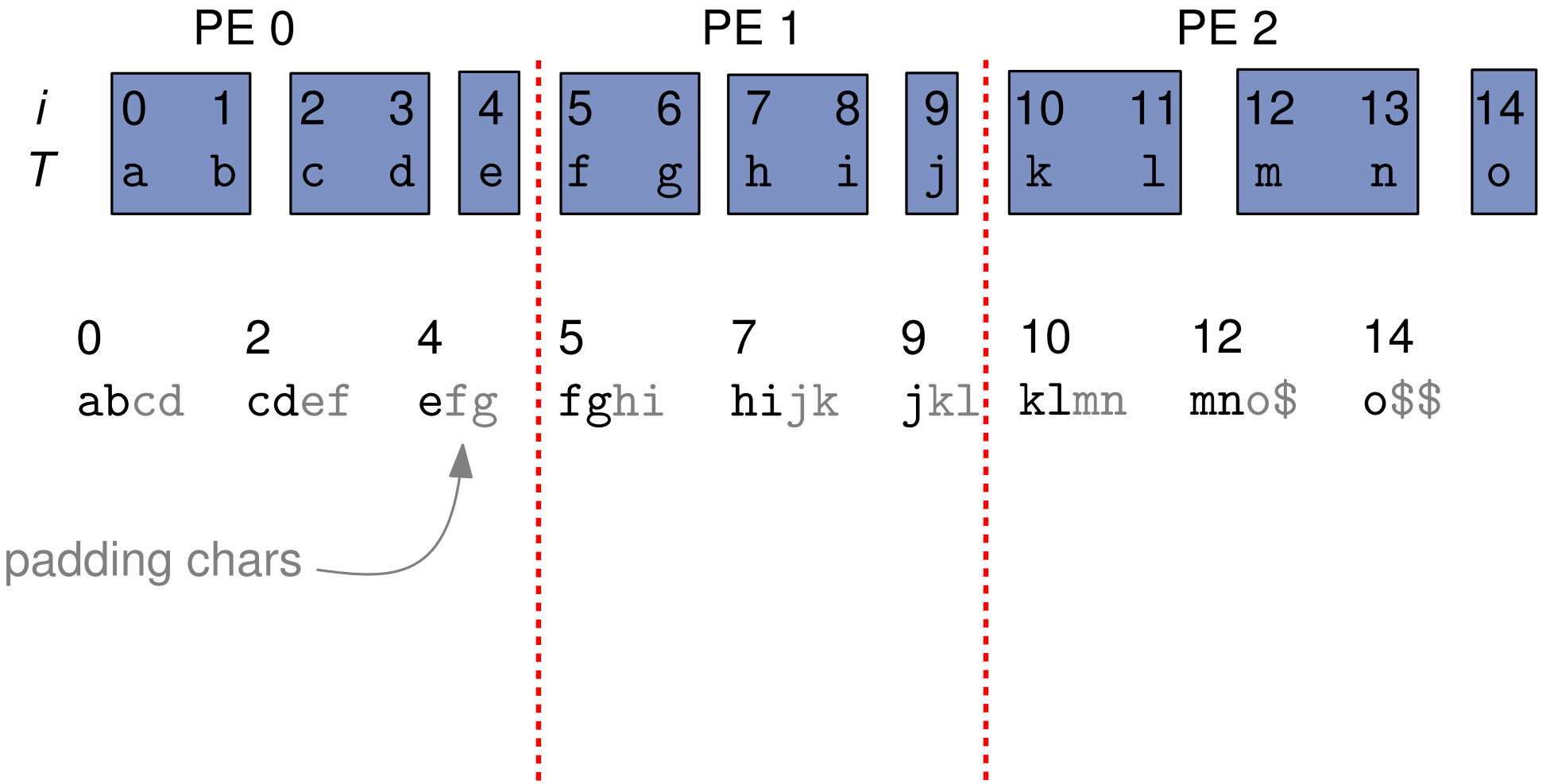
# Random Redistribution Using Chunking



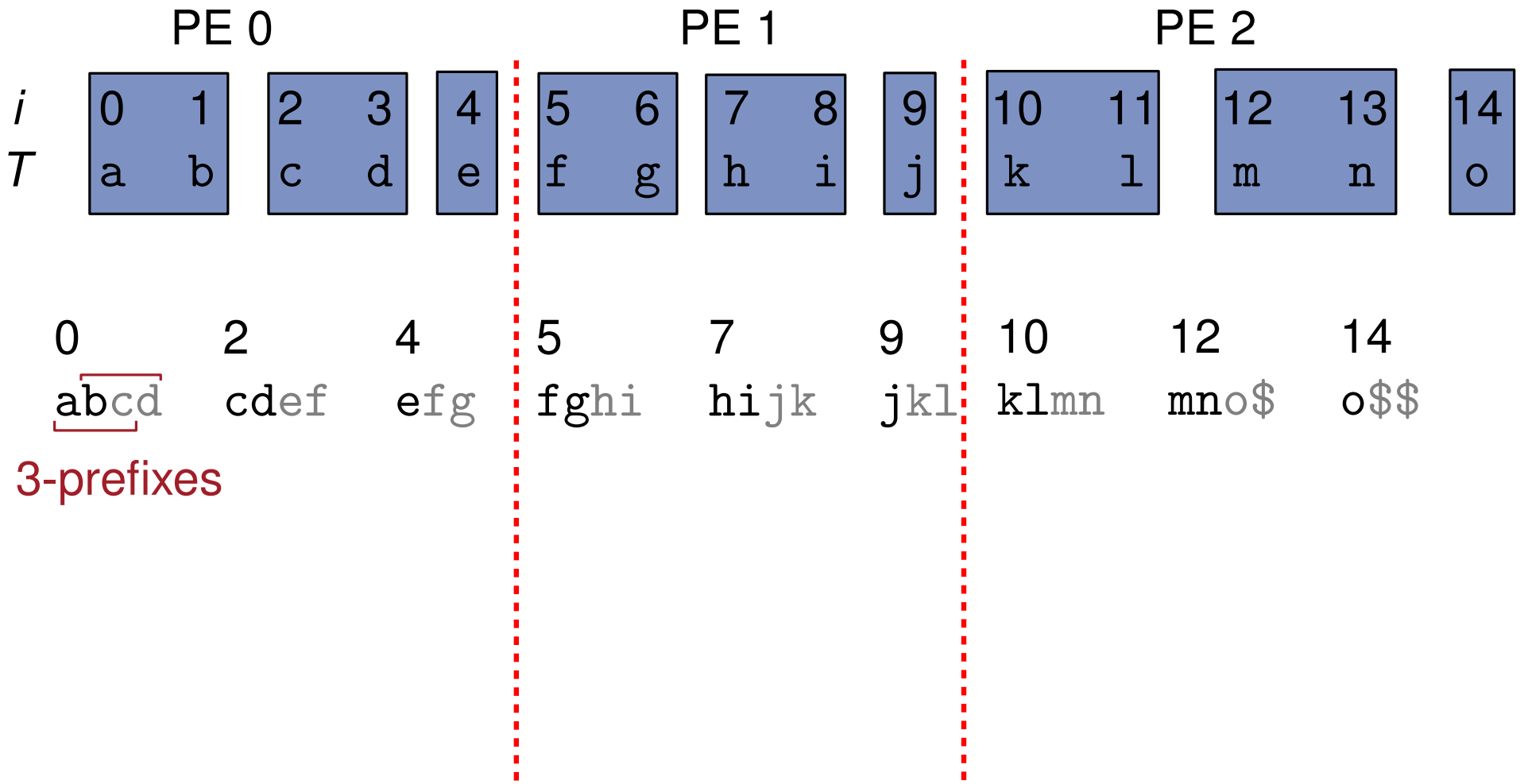
# Random Redistribution Using Chunking



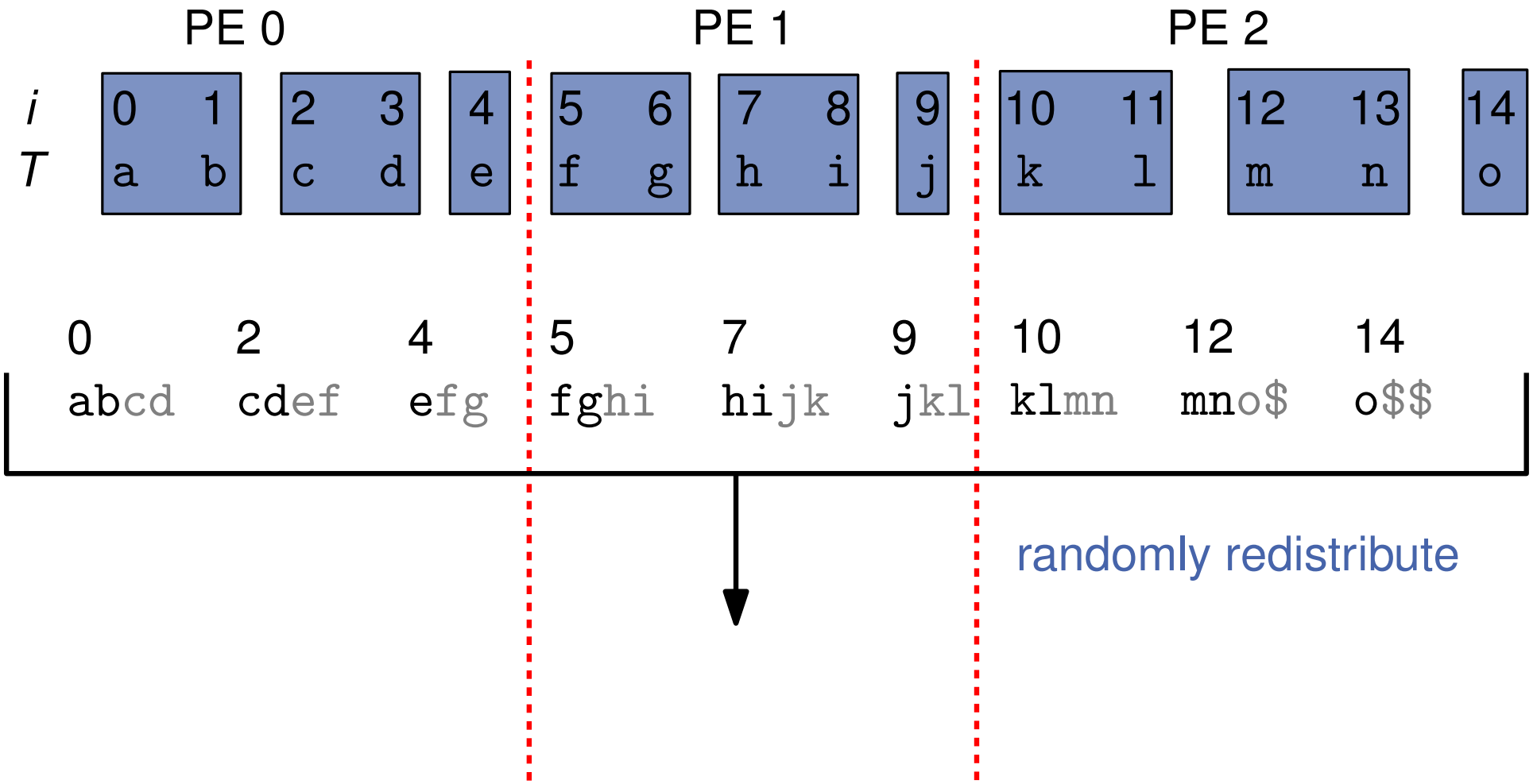
# Random Redistribution Using Chunking



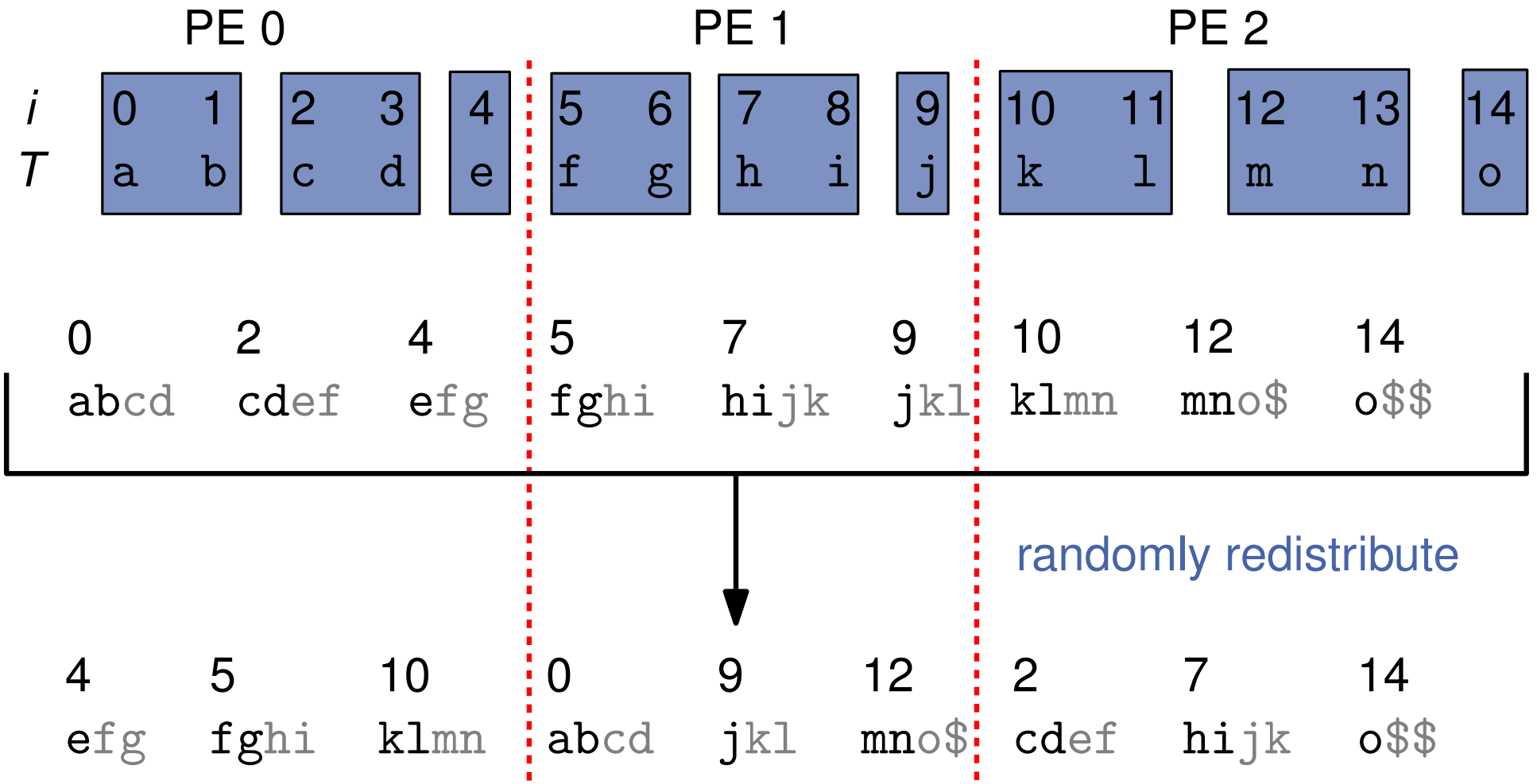
# Random Redistribution Using Chunking



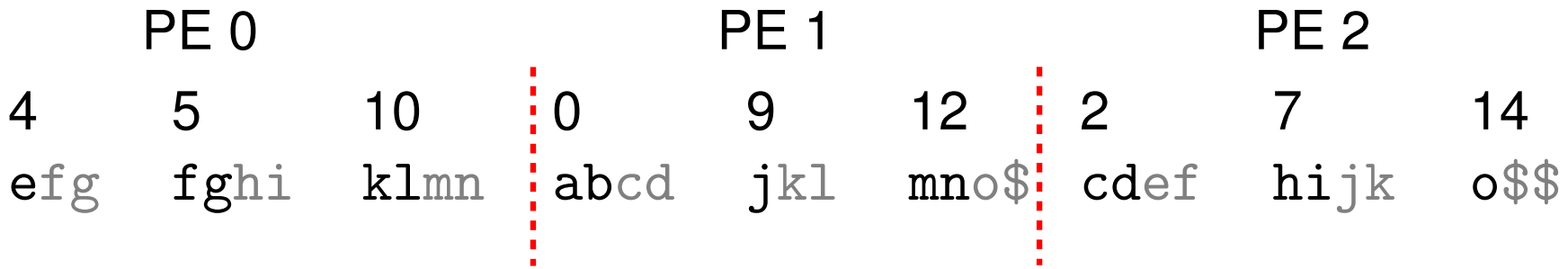
# Random Redistribution Using Chunking



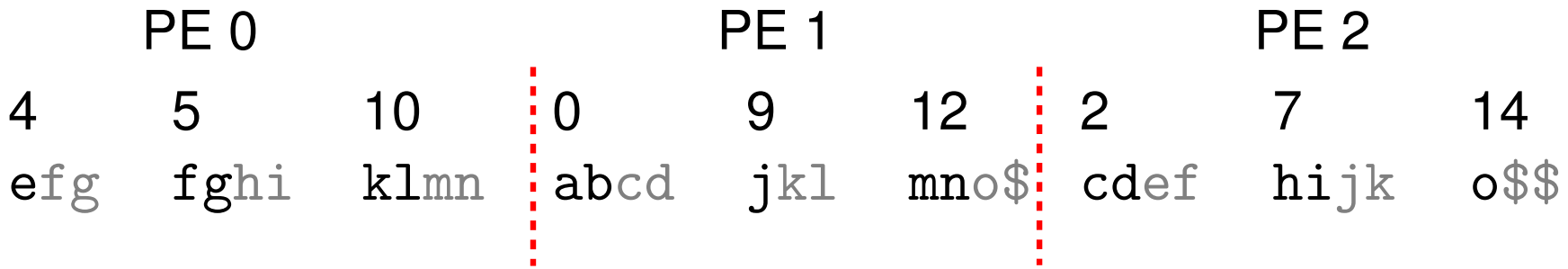
# Random Redistribution Using Chunking



# Random Redistribution Using Chunking

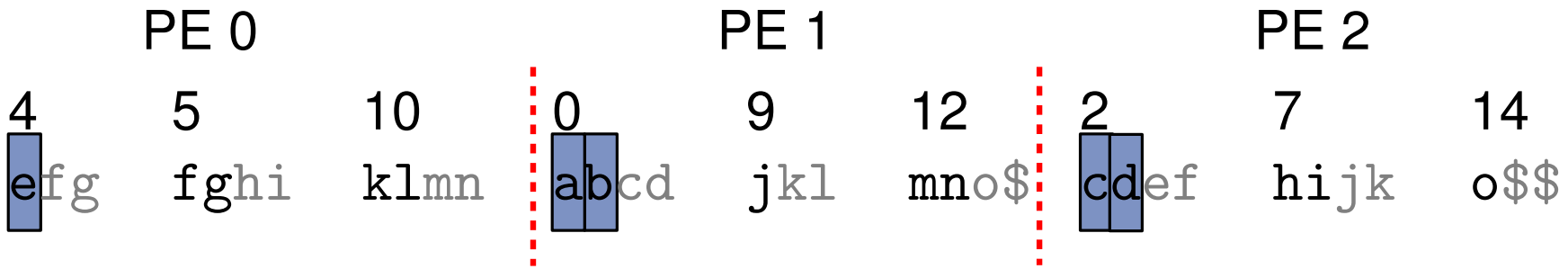


# Random Redistribution Using Chunking



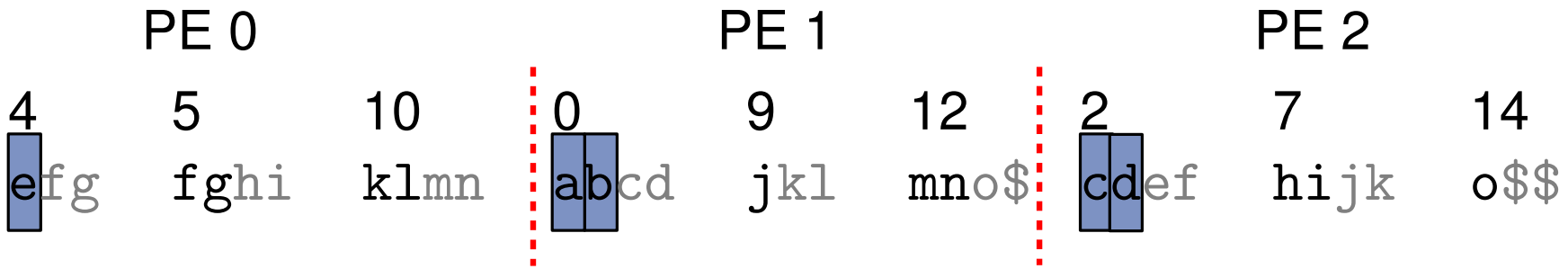

**splitters**: `fgh`, `kl$`

# Random Redistribution Using Chunking




**splitters**: `fgh`, `kl$`

# Random Redistribution Using Chunking




**splitters**: `fgh`, `kl$`

→ good load-balancing with high probabilty in theory and practice

# Further Optimizations

- discard suffixes not needed in recursive subproblem [Dementiev et al. 2008]
- packing chars into 64-bit words [Flick & Aluru 2015]
- string sample sort for $X$-prefixes
- 5-byte integer for ranks and indices

# Experimental Setup

## Input Texts

| Name | Mean LCP | Alphabet Size |
|---|---|---|
| DNA | 25.17 | 4 |
| Proteins | 179.31 | 26 |
| Wiki | 396.07 | 213 |
| CC | 10396.42 | 243 |

# Experimental Setup

## Input Texts

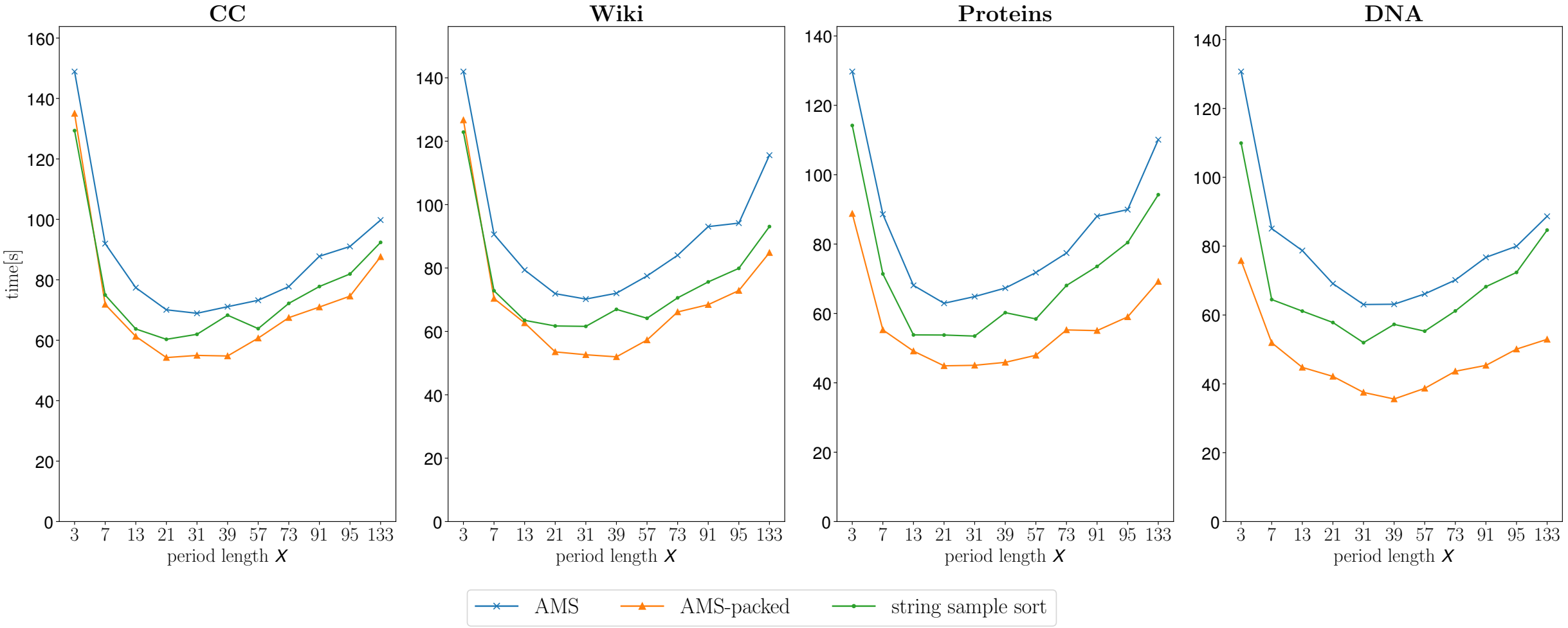
| Name | Mean LCP | Alphabet Size |
|---|---|---|
| DNA | 25.17 | 4 |
| Proteins | 179.31 | 26 |
| Wiki | 396.07 | 213 |
| CC | 10396.42 | 243 |

## Setup

- `c++20` and `MPI` with configuration for lower memory footprint
- experiments on up to 6144 cores of SuperMUC-NG

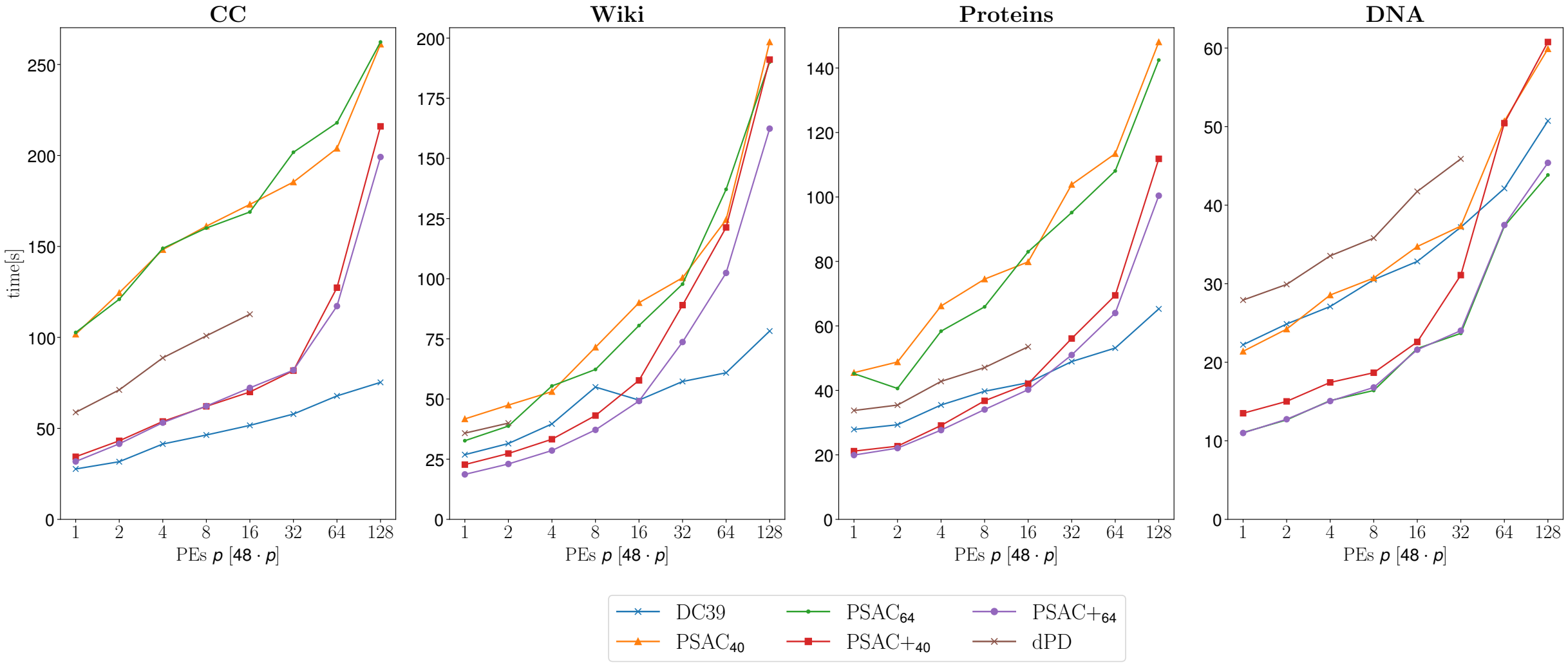
# Results - Sorting Variants

768 PEs, 20 MB input per PE



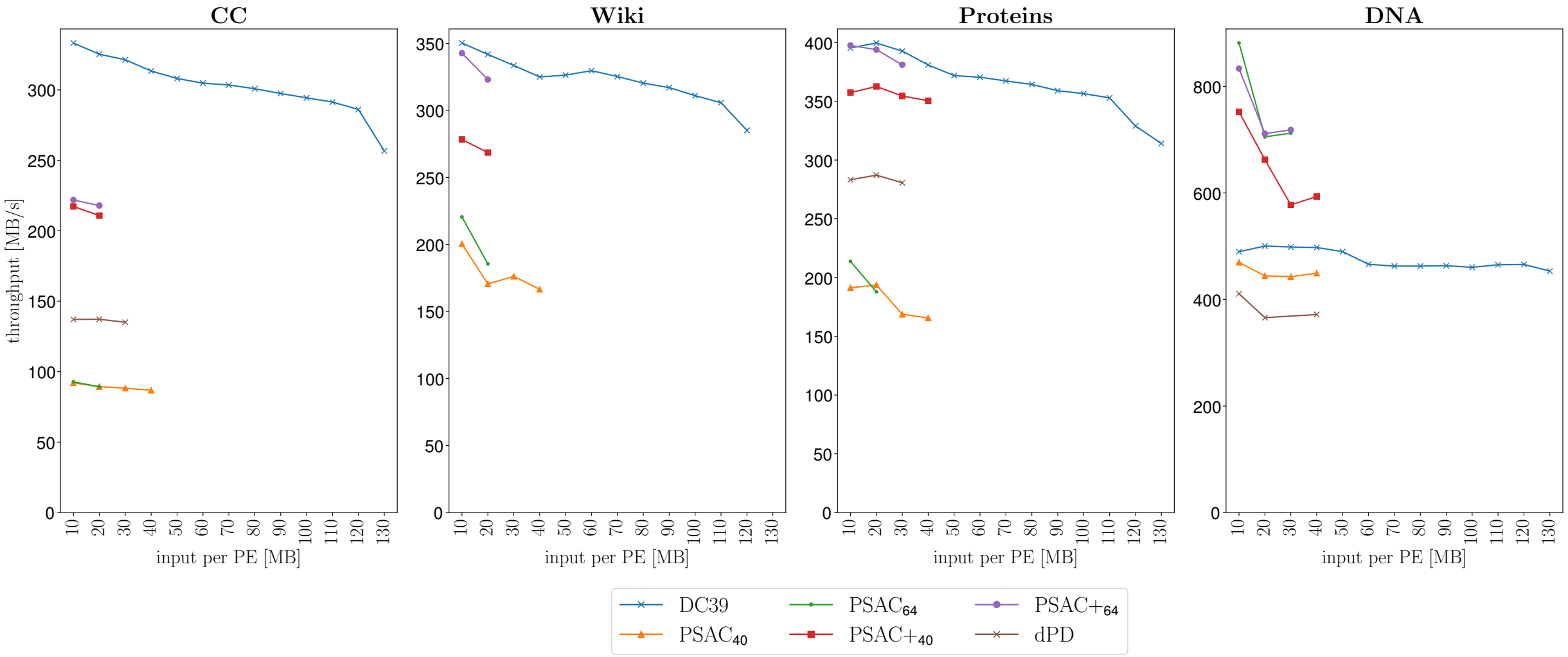
# Results - Weak Scaling Experiment

## 20 MB input per PE



# Results - Breakdown Test

768 PEs



# Conclusion

- DCX can process inputs 3× as large as competitors
- fastest on 3 out of 4 inputs (speedups up to 3.2×)
- good time-space trade-off on DNA

# Conclusion

- DCX can process inputs 3× as large as competitors
- fastest on 3 out of 4 inputs (speedups up to 3.2×)
- good time-space trade-off on DNA

## Future Work

- semi-external DCX
- hybrid between DCX and prefix doubling / inducing
- DCX on the GPU
- comparison with dDivSuffSort

# Bibliography


[Manber & Myers 1990] Suffix arrays: A new method for on-line string searches.

[Itoh & Tanaka 1999] An efficient method for in memory construction

[Kärkkäinen & Sanders 2003] Simple Linear Work Suffix Array

[Kärkkäinen et al. 2006] Linear work suffix array construction.

[Kulla & Sanders 2007]. Scalable parallel suffix array construction of suffix arrays.

[Dementiev et al. 2008] Better external memory suffix array construction.

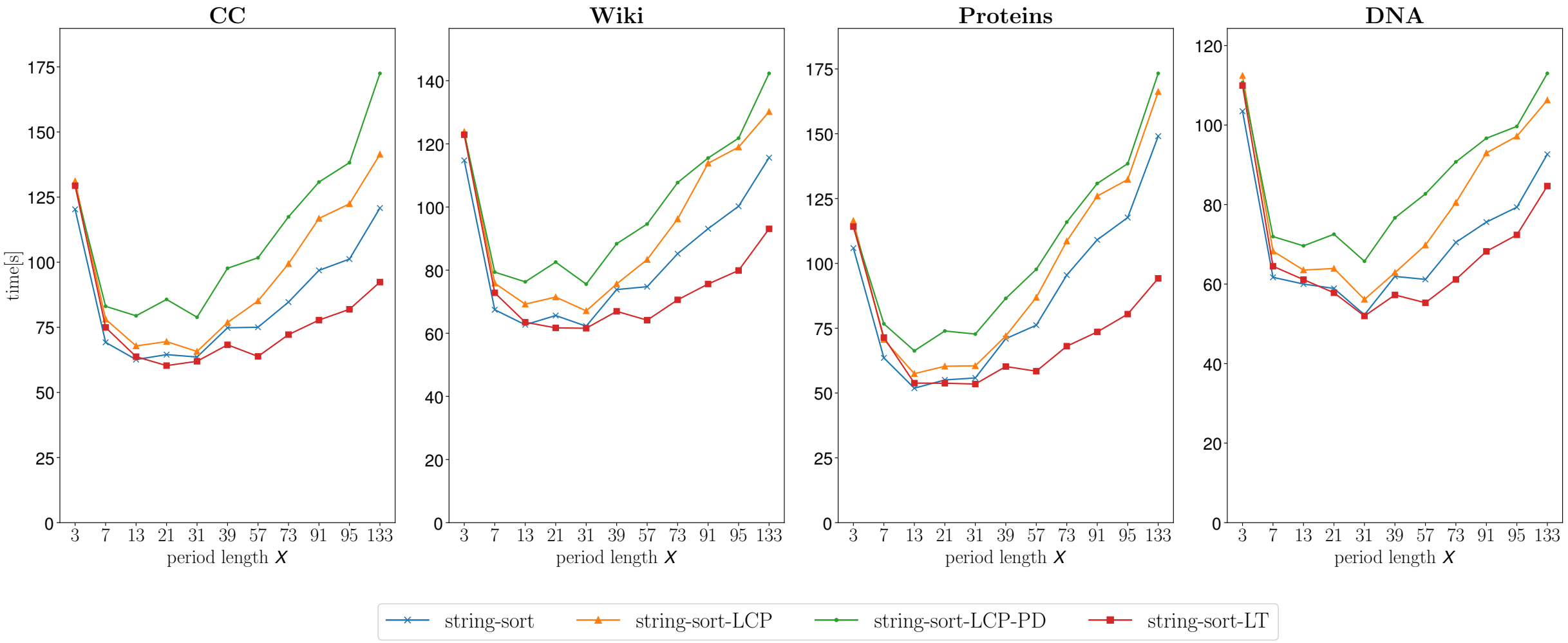
[Flick & Aluru 2015]. Parallel distributed memory construction of suffix and longest common prefix arrays.

[Fischer & Kurpicz 2019] Lightweight distributed suffix array construction.

[Mehnert 2024] Scalable distributed string sorting algorithms. Master's thesis.

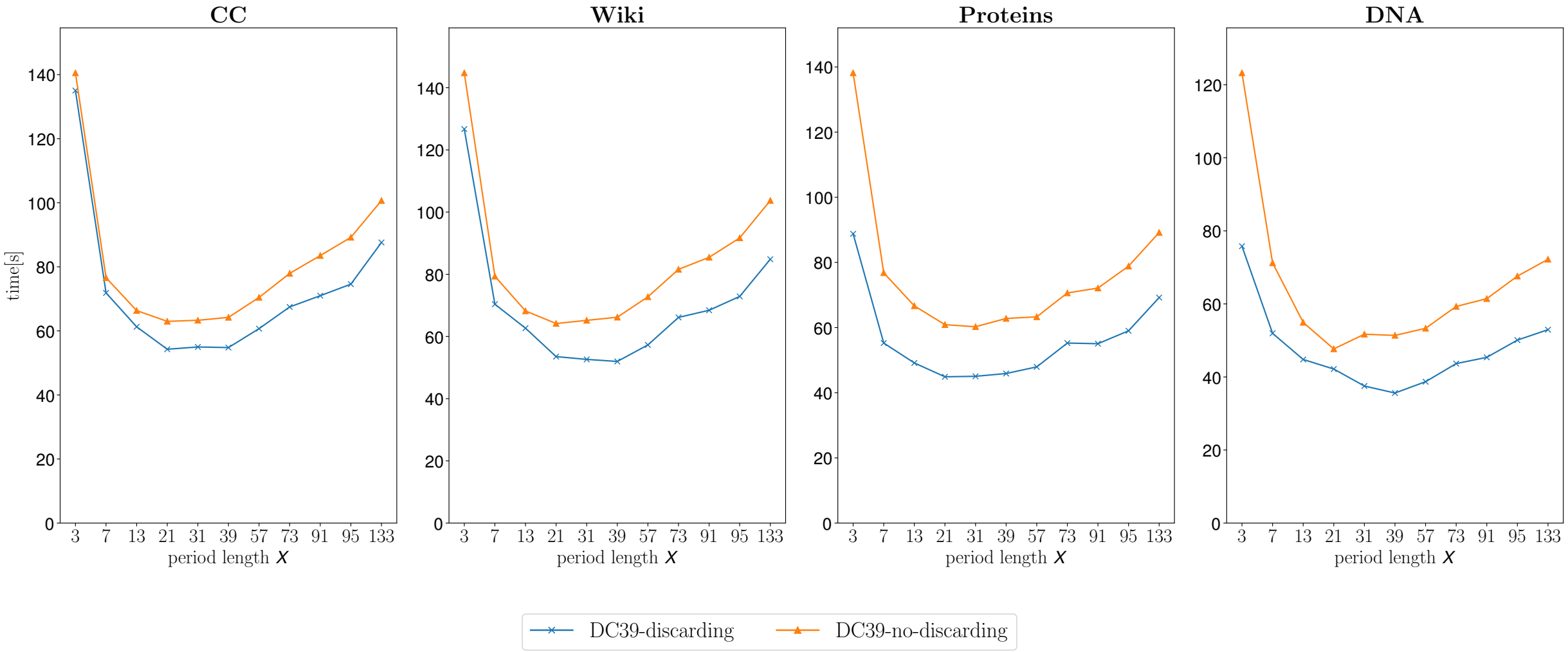
# Backup Slides

# Plots - String Sorting Variants

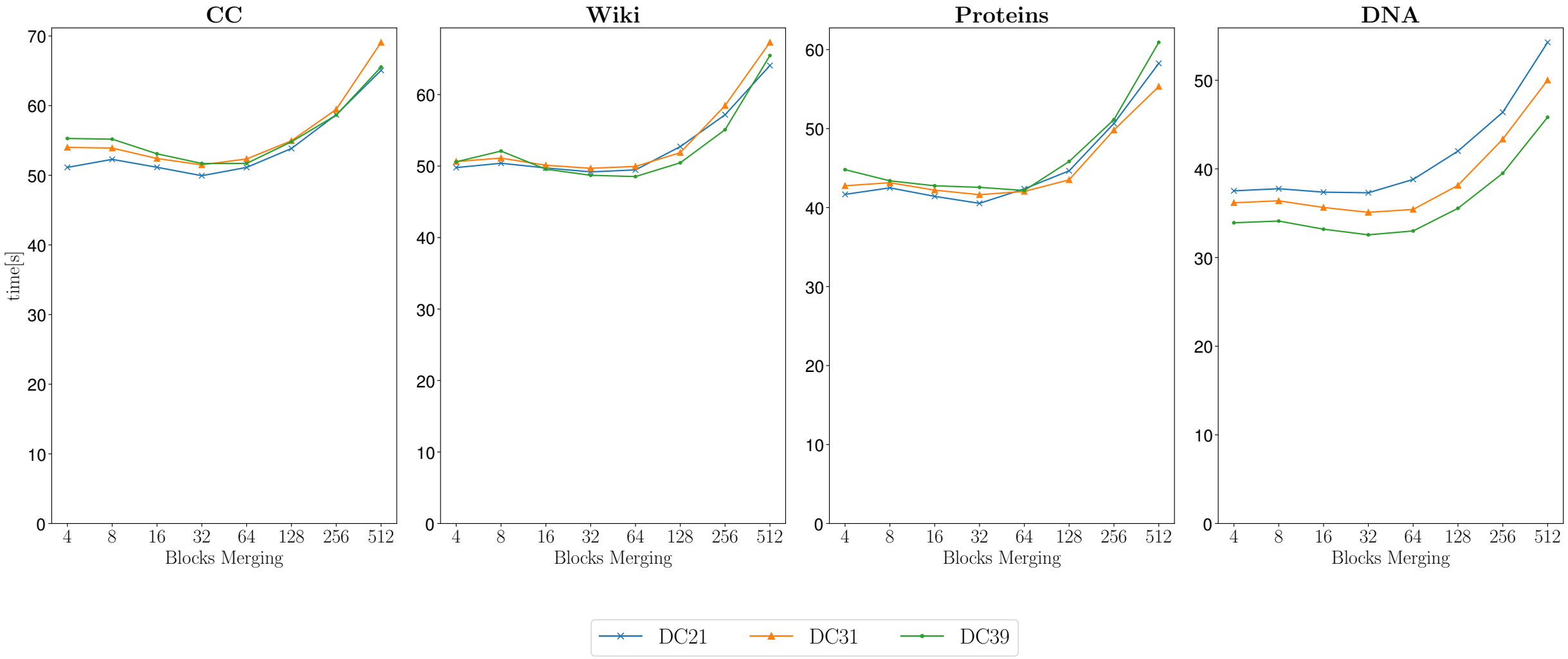


# Plots - Phase Times AMS-packed

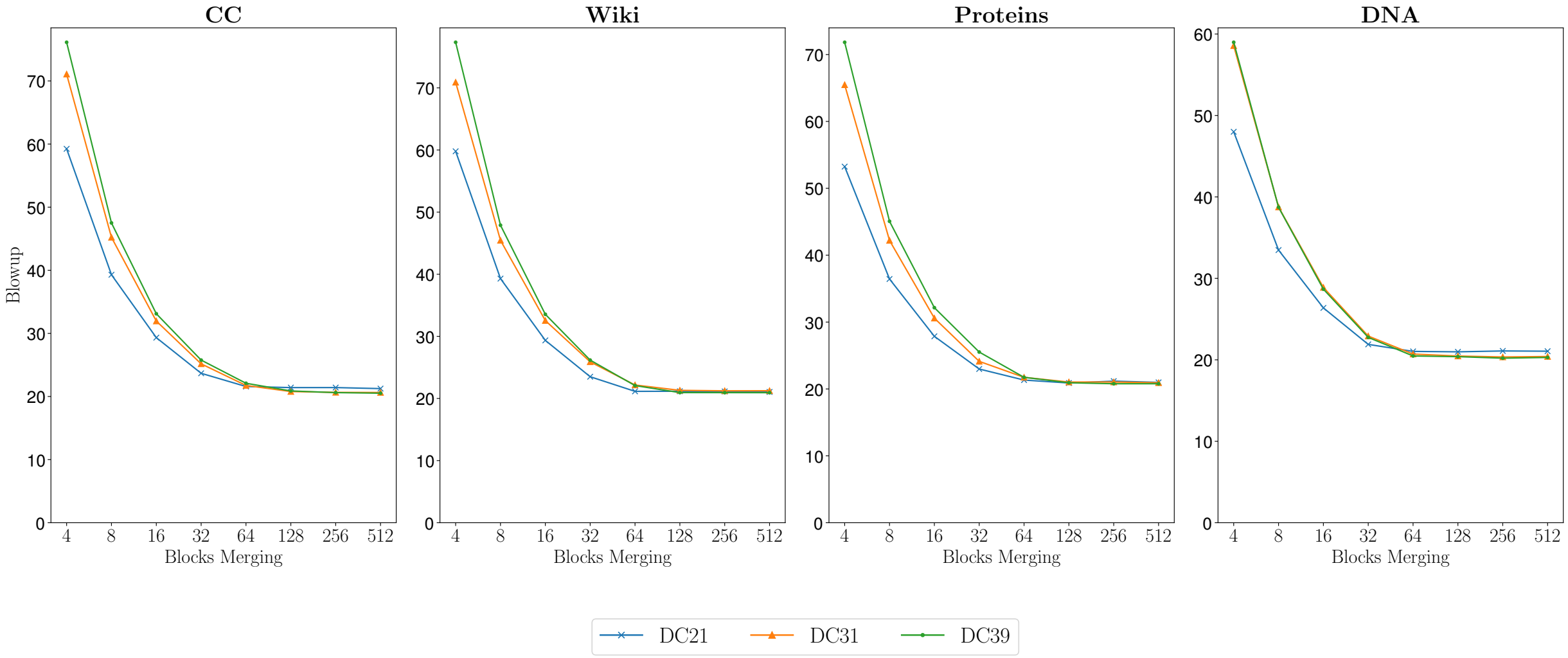
# Plots - AMS-packed - Discarding vs No-Discarding



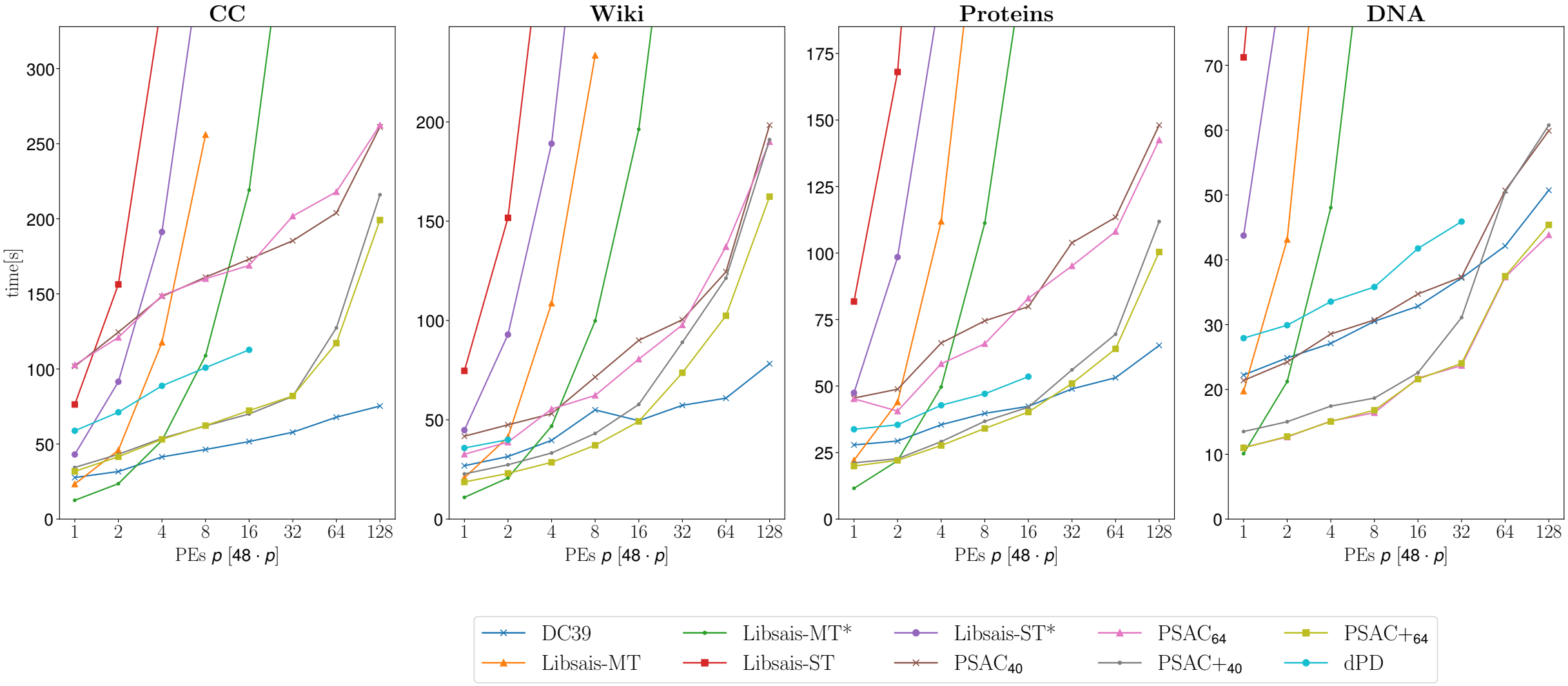
# Plots - AMS-packed - Blocks Sizes Time



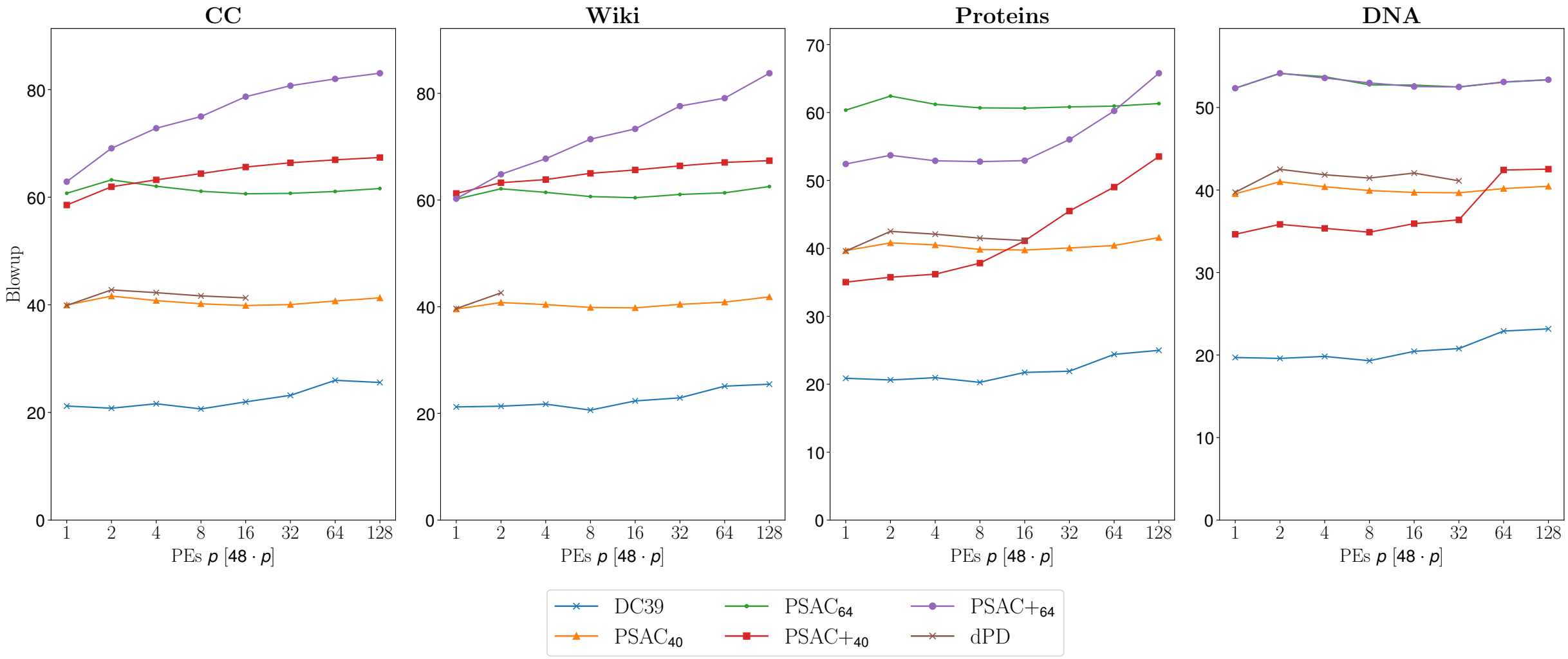
# Plots - AMS-packed - Blocks Sizes Memory



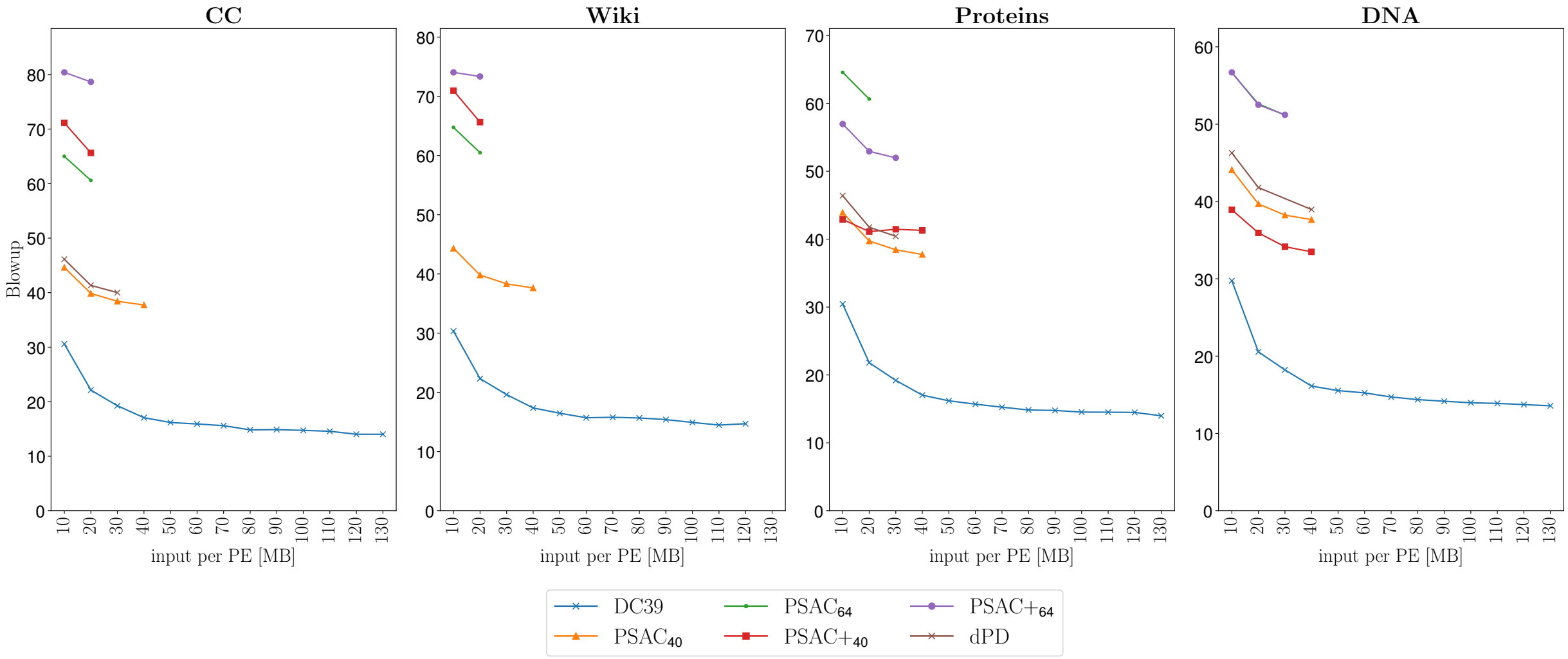
# Plots - Scaling Experiment Time Libsais



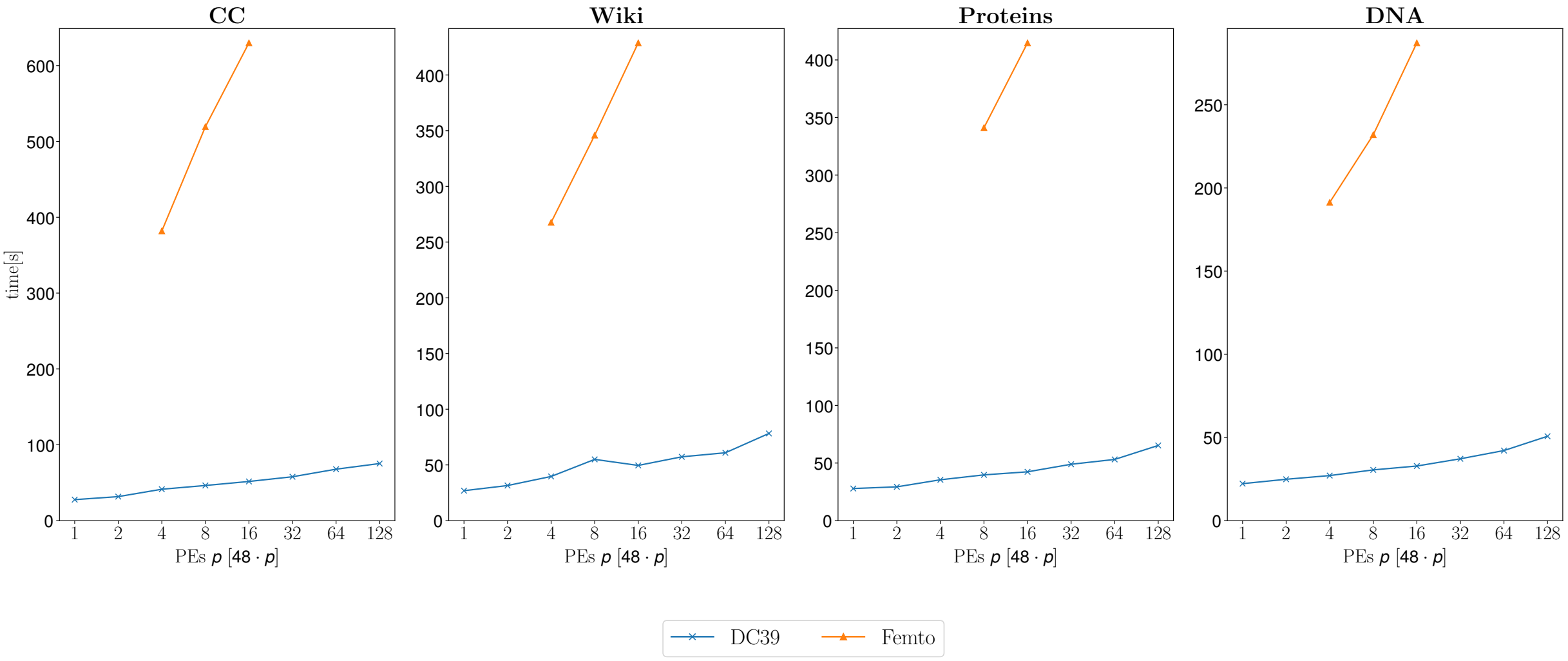
# Plots - Scaling Experiment Memory



# Plots - Breakdown Test Memory



# Plots - AMS-packed vs. Femto



# Difference Cover Modulo 3 (pDC3)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

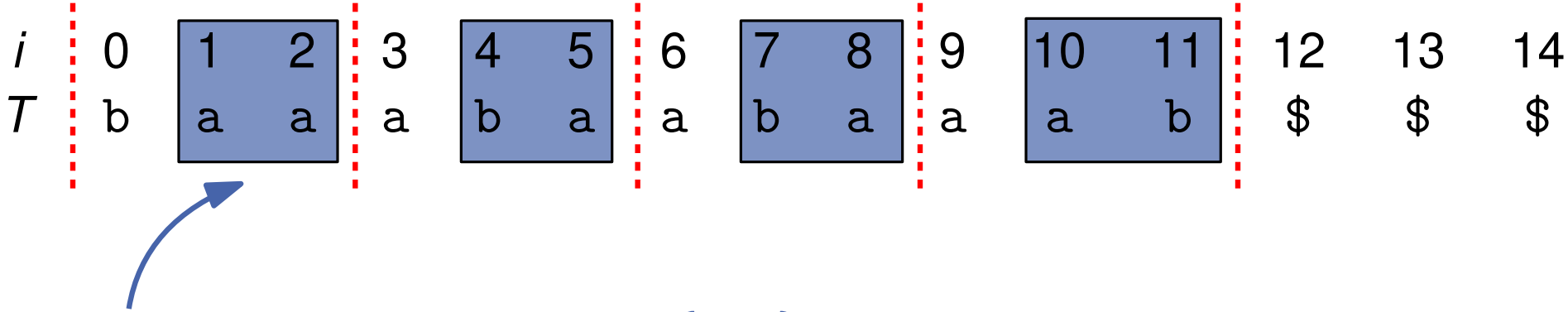
# Difference Cover Modulo 3 (pDC3)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

**Definition**

A set $D_X$ is a difference cover modulo $X$ if $\{(i-j) \bmod X \mid i,j \in D_X\} = [0,X)$.

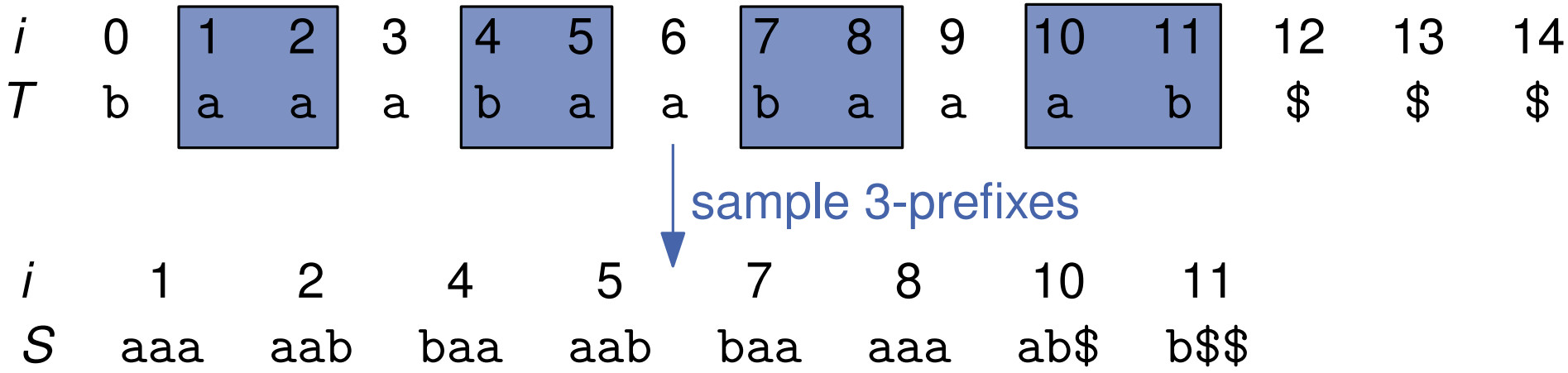
# Difference Cover Modulo 3 (pDC3)




**Definition**

A set $D_X$ is a difference cover modulo $X$ if $\{(i - j) \bmod X \mid i, j \in D_X\} = [0, X)$.

# Difference Cover Modulo 3 (pDC3)



# Difference Cover Modulo 3 (pDC3)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

| $i$ | 1 | 2 | 4 | 5 | 7 | 8 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|
| $S$ | aaa | aab | baa | aab | baa | aaa | ab$ | b$$ |

sort lexicographically

# Difference Cover Modulo 3 (pDC3)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

| $i$ | 1 | 8 | 2 | 5 | 10 | 11 | 4 | 7 |
|---|---|---|---|---|---|---|---|---|
| $S$ | aaa | aaa | aab | aab | ab$ | b$$ | baa | baa |
| rank | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 4 |

sort lexicographically

# Difference Cover Modulo 3 (pDC3)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

| $i$ | 1 | 8 | 2 | 5 | 10 | 11 | 4 | 7 |
|---|---|---|---|---|---|---|---|---|
| $S$ | aaa | aaa | aab | aab | ab$ | b$$ | baa | baa |
| rank | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 4 |

sort by ($i$ **mod** 3, $i$ **div** 3)

| $i$ | 1 | 4 | 7 | 10 | 2 | 5 | 8 | 11 |
|---|---|---|---|---|---|---|---|---|
| $P$ | 0 | 4 | 4 | 2 | 1 | 1 | 0 | 3 |

# Difference Cover Modulo 3 (pDC3)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |

| $i$ | 1 | 8 | 2 | 5 | 10 | 11 | 4 | 7 |
|---|---|---|---|---|---|---|---|---|
| $S$ | aaa | aaa | aab | aab | ab$ | b$$ | baa | baa |
| rank | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 4 |

| $i$ | 1 | 4 | 7 | 10 | 2 | 5 | 8 | 11 |
|---|---|---|---|---|---|---|---|---|
| $P$ | 0 | 4 | 4 | 2 | 1 | 1 | 0 | 3 |

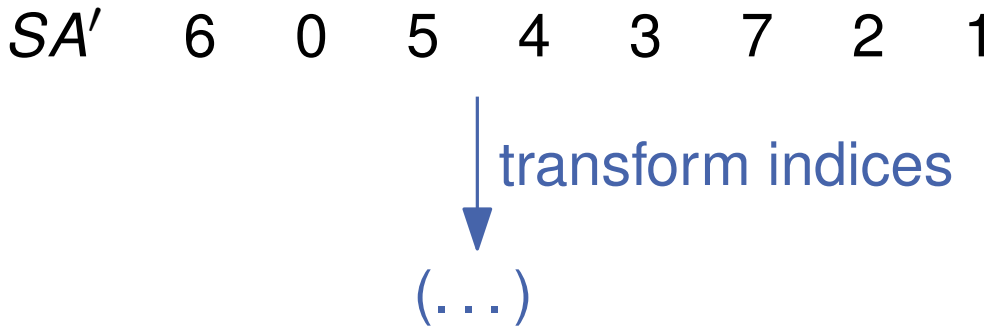
recursion on $P$

| $SA'$ | 6 | 0 | 5 | 4 | 3 | 7 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|

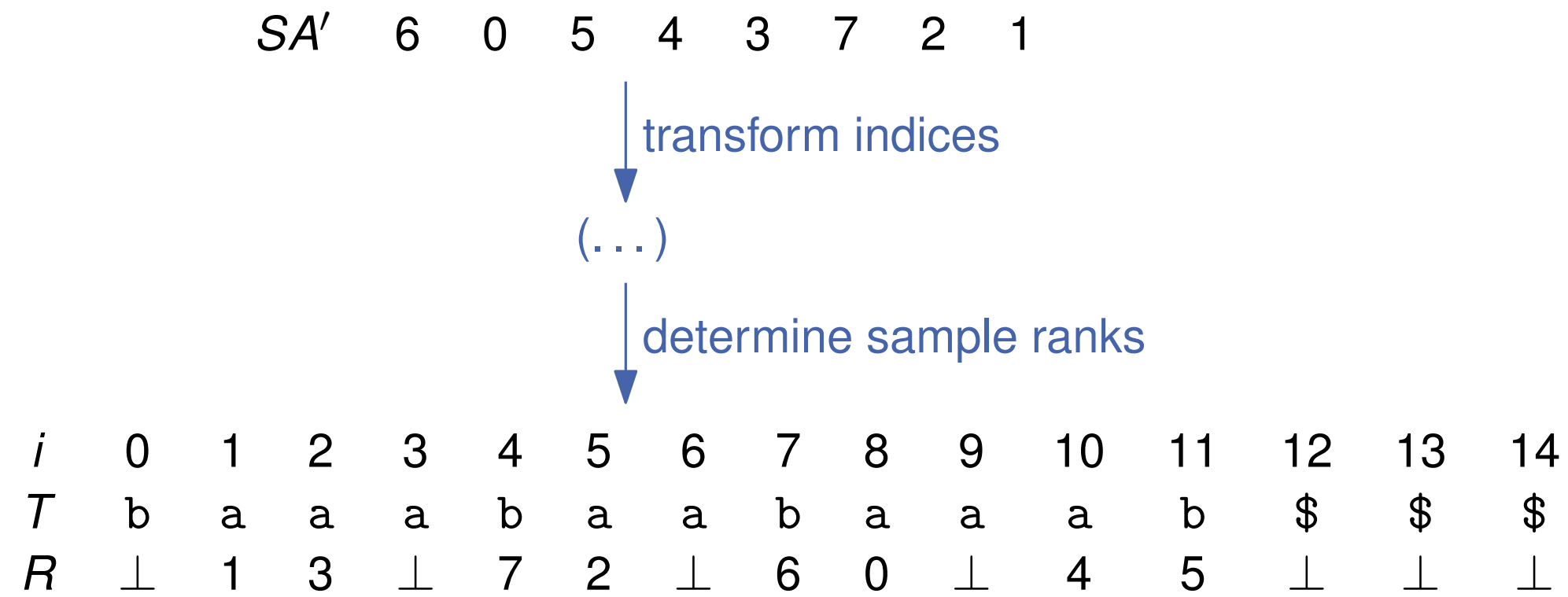
# Difference Cover Modulo 3 (pDC3)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $SA'$ | 6 | 0 | 5 | 4 | 3 | 7 | 2 | 1 |

# Difference Cover Modulo 3 (pDC3)



# Difference Cover Modulo 3 (pDC3)



# Difference Cover Modulo 3 (pDC3)

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison:

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (pDC3)

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison:

**Lemma 1**

If $D_X$ is a difference cover modulo $X$, and $i$ and $j$ are integers, there exists $l \in [0, X)$ such that $(i + l)$ **mod** $X$ and $(j + l)$ **mod** $X$ are in $D_X$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (pDC3)

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison: $S_1 > S_8$

**Lemma 1**

If $D_X$ is a difference cover modulo $X$, and $i$ and $j$ are integers, there exists $l \in [0, X)$ such that $(i + l)$ **mod** $X$ and $(j + l)$ **mod** $X$ are in $D_X$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (pDC3)

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison: $S_3 > S_6$

**Lemma 1**

If $D_X$ is a difference cover modulo $X$, and $i$ and $j$ are integers, there exists $l \in [0, X)$ such that $(i + l)$ **mod** $X$ and $(j + l)$ **mod** $X$ are in $D_X$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (pDC3)

**Globally sort all suffixes:** Compare characters until both positions are samples.

comparison: $S_2 < S_6$

**Lemma 1**

If $D_X$ is a difference cover modulo $X$, and $i$ and $j$ are integers, there exists $l \in [0, X)$ such that $(i + l)$ **mod** $X$ and $(j + l)$ **mod** $X$ are in $D_X$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (pDC3)

## Distributed Memory

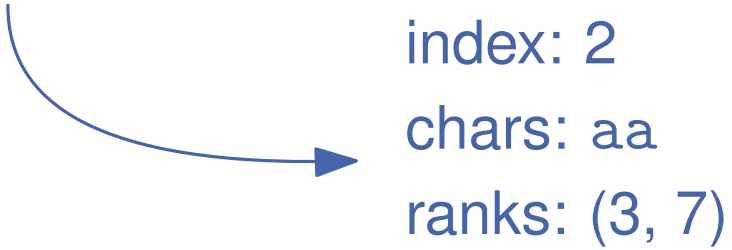
- distributed sorting routines
- shifts characters/ranks between adjacent PEs
- materialize chars and prefixes to send none-local data

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (pDC3)

## Distributed Memory

- distributed sorting routines
- shifts characters/ranks between adjacent PEs
- materialize chars and prefixes to send none-local data



index: 2

chars: aa

ranks: (3, 7)

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (pDC3)

## Distributed Memory

- distributed sorting routines
- shifts characters/ranks between adjacent PEs
- materialize chars and prefixes to send none-local data

index: 2

chars: aa

ranks: (3, 7)

memory blowup by 3×

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | ⊥ | 1 | 3 | ⊥ | 7 | 2 | ⊥ | 6 | 0 | ⊥ | 4 | 5 | ⊥ | ⊥ | ⊥ |

# Difference Cover Modulo 3 (pDC3)

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow X$-prefixes
- size of recursive subproblem:

  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (pDC3)

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow$ $X$-prefixes $\longrightarrow$ memory blowup by $X\times$
- size of recursive subproblem:

  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (pDC3)

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow$ $X$-prefixes $\longrightarrow$ memory blowup by $X\times$
- size of recursive subproblem:
  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$ $\longrightarrow$ higher reduction

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |

# Difference Cover Modulo 3 (pDC3)

## Distributed DCX

- $D_3 \rightarrow D_X$, e.g. $D_7 = \{1, 2, 4\}$
- 3-prefixes $\rightarrow$ $X$-prefixes
- size of recursive subproblem:
  $|D_X|/X \in \mathcal{O}(1/\sqrt{X})$

→ memory blowup by $X\times$

→ higher reduction

need for space-efficient implementation

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | b | a | a | a | b | a | a | b | a | a | a | b | $ | $ | $ |
| $R$ | $\perp$ | 1 | 3 | $\perp$ | 7 | 2 | $\perp$ | 6 | 0 | $\perp$ | 4 | 5 | $\perp$ | $\perp$ | $\perp$ |