

# Web Data Compression and Search

Fast BWT Construction

# Space Efficient Linear Time Construction of Suffix Arrays

A good paper by Pang Ko and Srinivas Aluru  
J. Discrete Algorithms 3(2-4): 143-156(2005)

# Suffix Array

- Sorted order of suffixes of a string  $T$ .
- Represented by the starting position of the suffix.

Text	M	I	S	S	I	S	S	I	P	P	I	\$
Index	1	2	3	4	5	6	7	8	9	10	11	12
Suffix Array	12	11	8	5	2	1	10	9	7	4	6	3

# Notation

- String  $T = t_1 \dots t_n$ .
- Over the alphabet  $\Sigma = \{1 \dots n\}$ .
- $t_n = '\$'$ , ‘\$’ is a unique character.
- $T_i = t_i \dots t_n$ , denotes the  $i$ -th suffix of  $T$ .
- For strings  $\alpha$  and  $\beta$ ,  $\alpha < \beta$  denotes  $\alpha$  is lexicographically smaller than  $\beta$ .

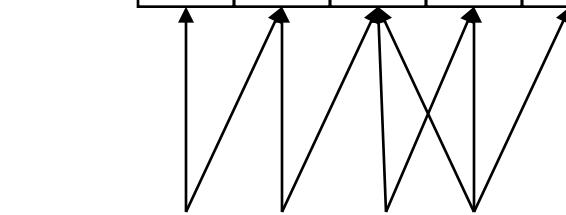
# Overview

- Divide all suffixes of  $T$  into two types.
  - Type S suffixes =  $\{T_i \mid T_i < T_{i+1}\}$
  - Type L suffixes =  $\{T_j \mid T_j > T_{j+1}\}$
  - The last suffix is both type  $S$  and  $L$ .
- Sort all suffixes of one of the types.
- Obtain lexicographical order of all suffixes from the sorted ones.

# Identify Suffix Types

Type	L	S	L	L	S	L	L	S	L	L	L	L/S
------	---	---	---	---	---	---	---	---	---	---	---	-----

Text	M	I	S	S	I	S	S	I	P	P	I	\$
------	---	---	---	---	---	---	---	---	---	---	---	----



$M > II < SS \Rightarrow S,$

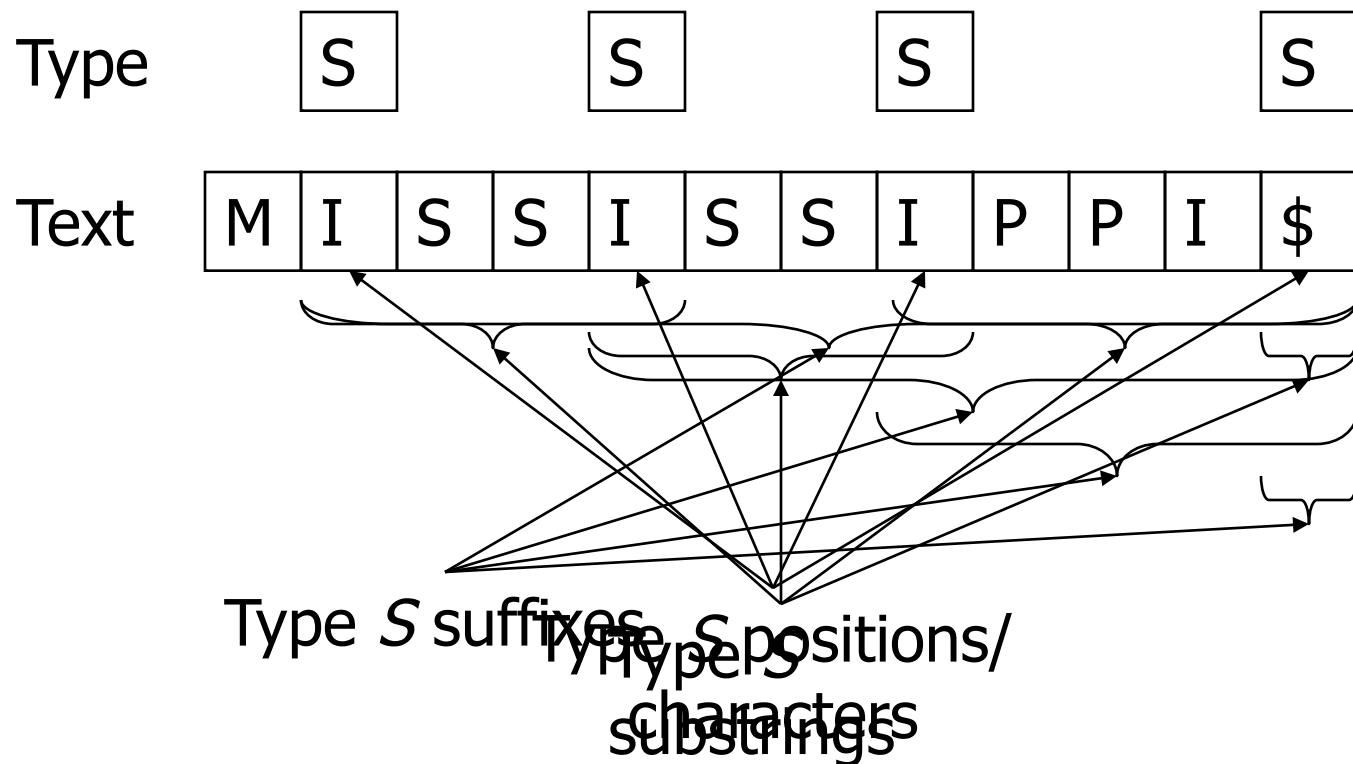
$T_1 > T_2 < T_3 \text{ and } T_4 > T_5$

$T_1$  is type  $L$ ,  $T_2$  is type  $S$ ,  $T_3$  is type  $S$ ,  $T_4$  is type  $L$ , and  $T_5$  is type  $S$ .

The type of each suffix in  $T$  can be determined in one scan of the string.



# Notation



# Sorting Type $S$ Suffixes

- Sort all type  $S$  substrings.
- Replace each type  $S$  substrings by its bucket number.
- New string is the sequence of bucket numbers.
- Sorting all type  $S$  suffixes = Sorting all suffixes of the new string.

# Sorting Type *S* Substrings

The diagram illustrates the Bucket Sort algorithm for string processing, showing the steps from an input text to a sorted output.

**Type**: **S** **S** **S** **S**

**Text**: **B** **B** **g** **\$** **I** **S** **S** with the buckets numbers to obtain a new string.

**Bucket Sort**: A 4x4 grid representing buckets. The columns are labeled \$, I, I, I. The rows are labeled P, \$, S; P, S, S; I, I, I. Arrows point from the text to the grid, and from the grid to the sorted output.

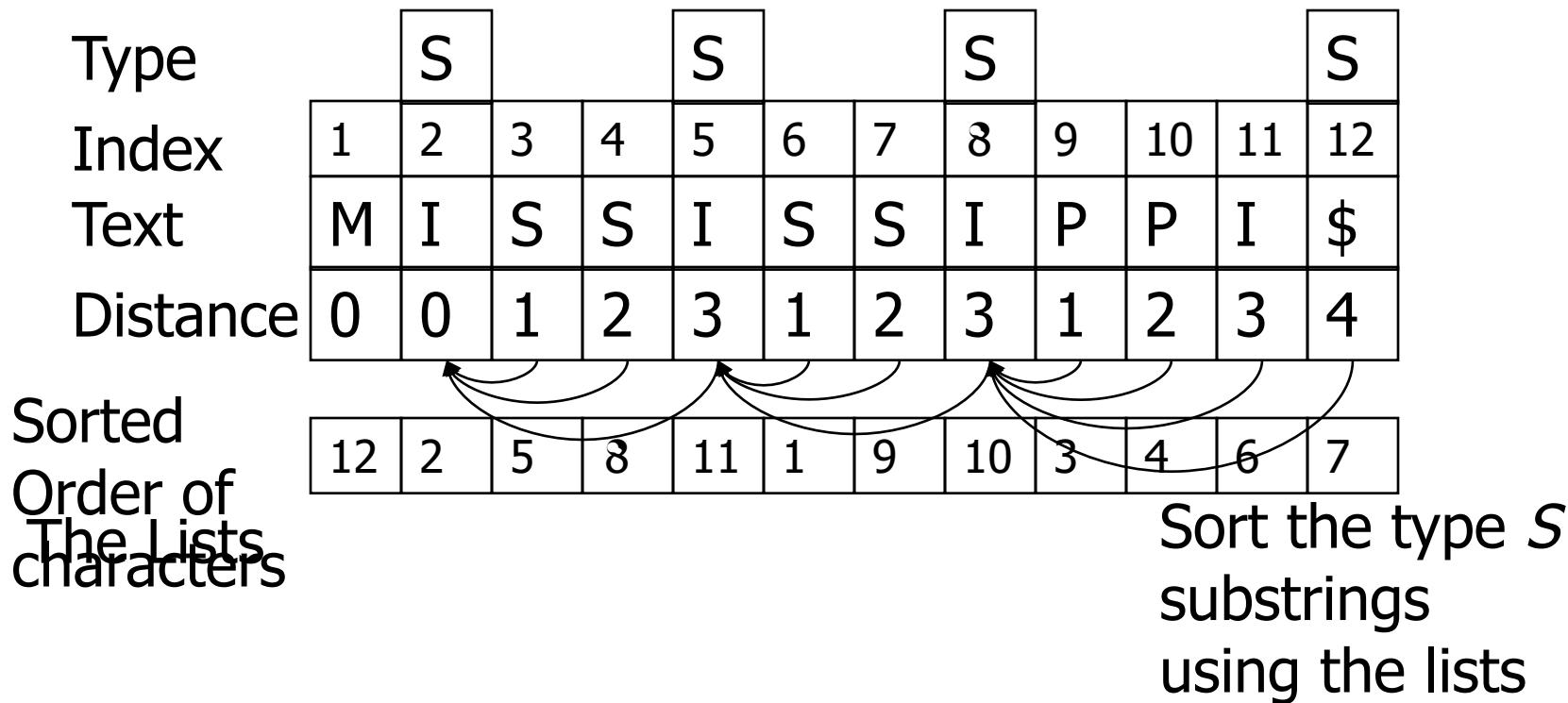
Annotations explain the process:

- Substitute the substrings with the bucket numbers to obtain a new string.
- Apply sorting recursively to the new string.
- Sort each substring according to the next type character.
- Break into 2 buckets.

# Solution

- Observation: Each character participates in the bucket sort at most twice.
  - Type  $L$  characters only participate in the bucket sort once.
- Solution:
  - Sort all the characters once.
  - Construct  $m$  lists according the distance to the closest type  $S$  character to the left

# Illustration

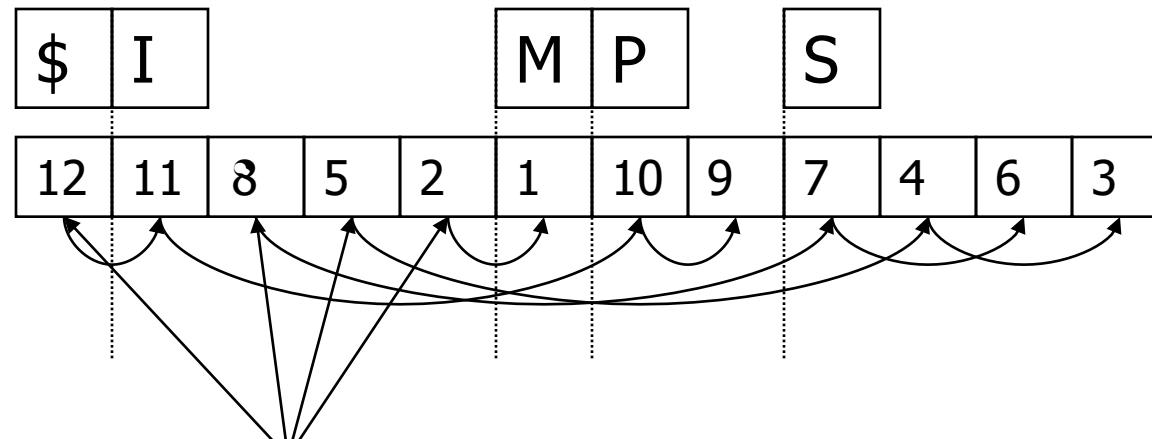


T	M	I	S	S	I	S	S	I	P	P	I	\$	Step 3. Sort all type S substrings
Type		S		S		S		S		S		S	Original
Pos	1	2	3	4	5	6	7	8	9	10	11	12	
A	12	12	5	8	11	11	9	10	3	4	6	7	
	12	12	5	8	11	11	9	10	3	4	6	7	
	12	12	5	8	11	11	9	10	3	4	6	7	Sort according to list 1
	12	12	5	8	11	11	9	10	3	4	6	7	12 8 5
	12	12	5	8	11	11	9	10	3	4	6	7	Sort according to list 2
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8
	12	12	5	8	11	11	9	10	3	4	6	7	Sort according to list 3
	12	12	5	8	11	11	9	10	3	4	6	7	12 8 5
	12	12	5	8	11	11	9	10	3	4	6	7	Sort according to list 4
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6 5
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6 5 4
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6 5 4 3
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6 5 4 3 2
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6 5 4 3 2 1
	12	12	5	8	11	11	9	10	3	4	6	7	12 5 8 11 10 9 8 7 6 5 4 3 2 1 0

Fig. 3. Illustration of the sorting of type S substrings of the string

# Construct Suffix Array for all Suffixes

- The first suffix in the suffix array is a type S suffix.
- For  $1 \leq i \leq n$ , if  $T_{SA[i]-1}$  is type  $L$ , move it to the current front of its bucket
- $[\$:12][I:2,5,8,11][M:1][P:9,10][S:3,4,6,7]$



Sorted order of  
type  $S$  suffixes



# Exercise

- Consider the popular example string S:
  - $\$$
1. Construct the suffix array of S using the linear time algorithm
  2. Then compute the  $\text{BWT}(S)$
  3. What's the relationship between the suffix array and BWT ?

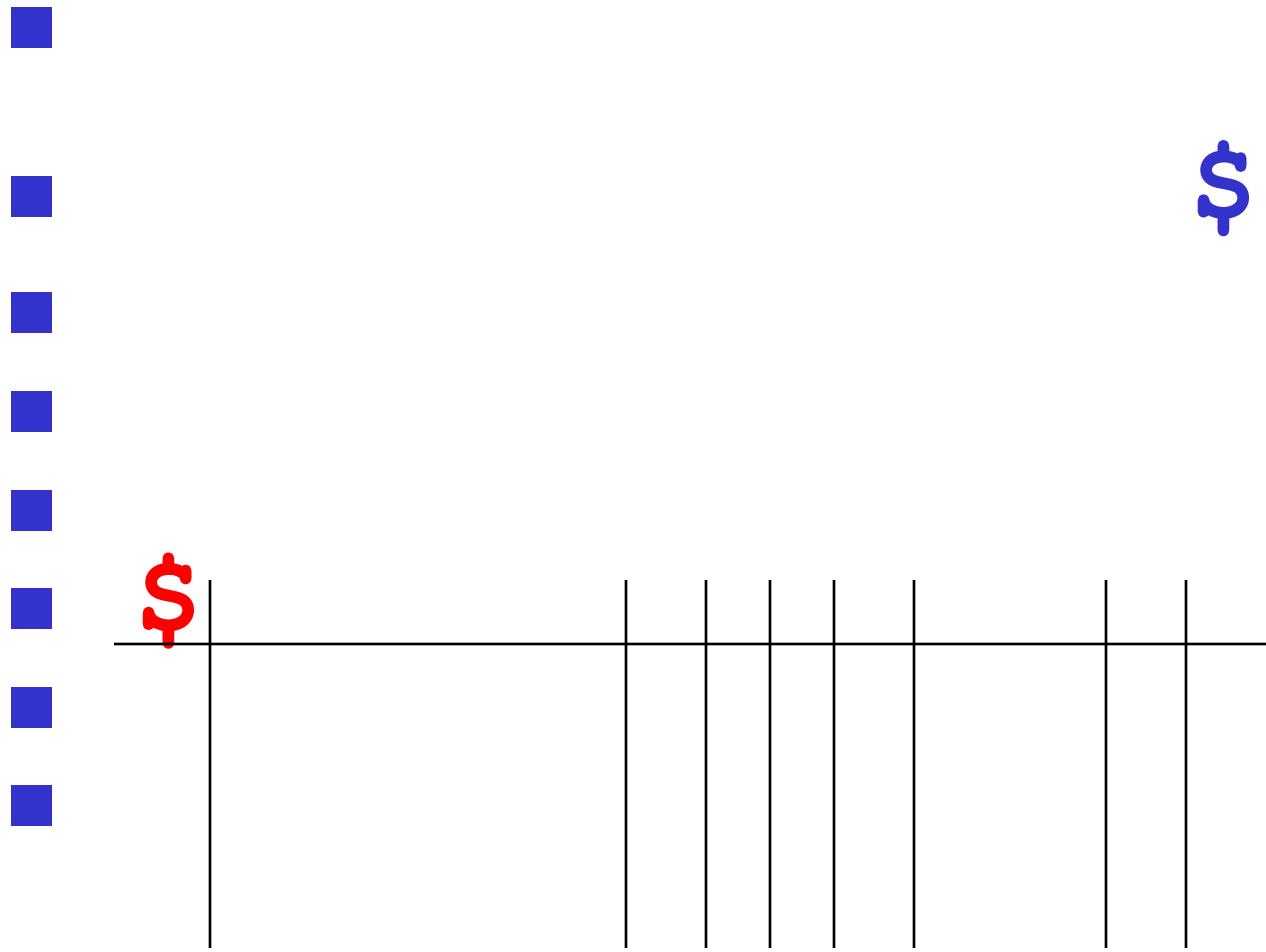
# Step – Identify the type of each suffix

- 
- \$
- 
-

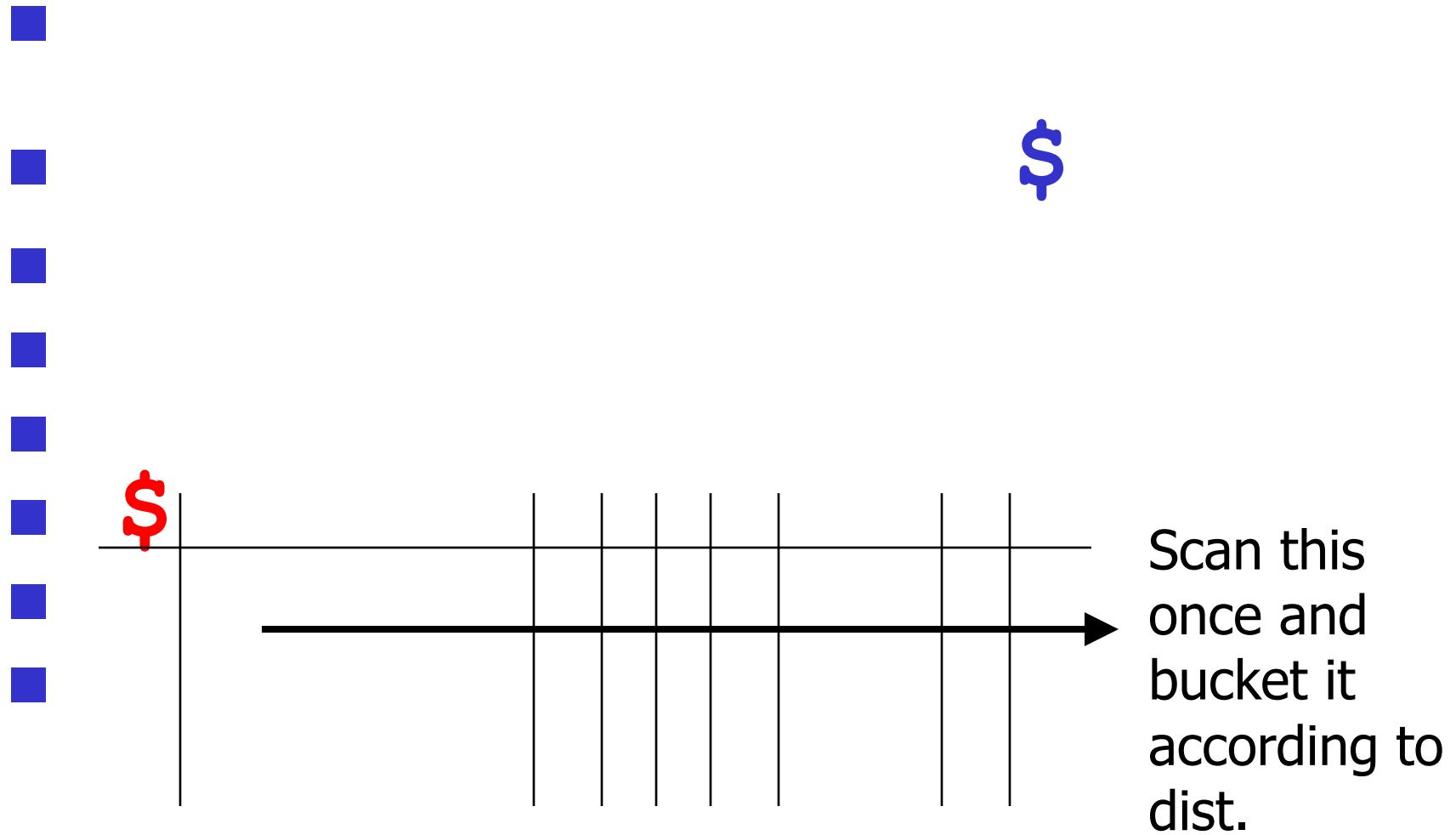
# Step – Compute the distance from S



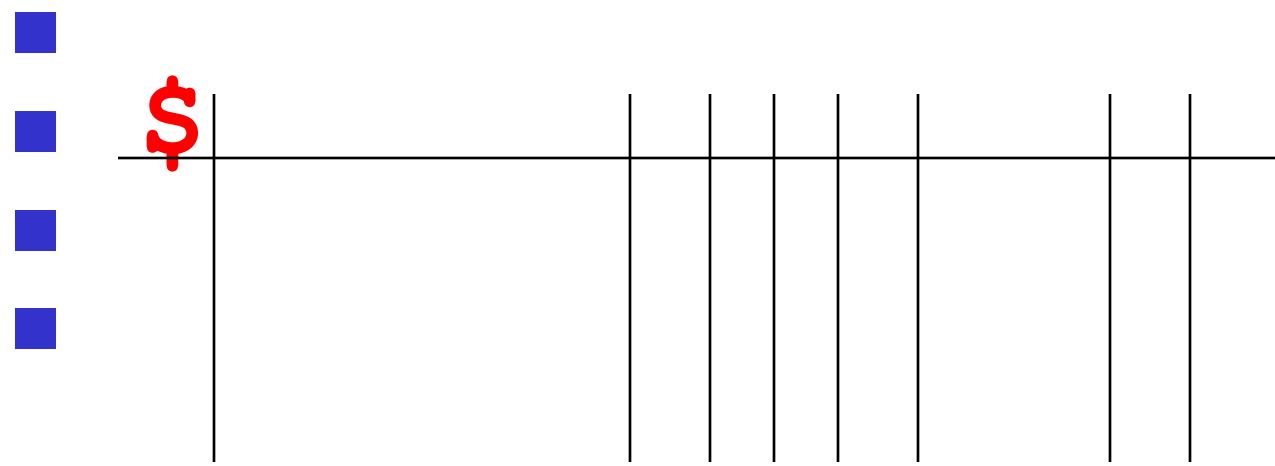
# Step – Sort order of chars



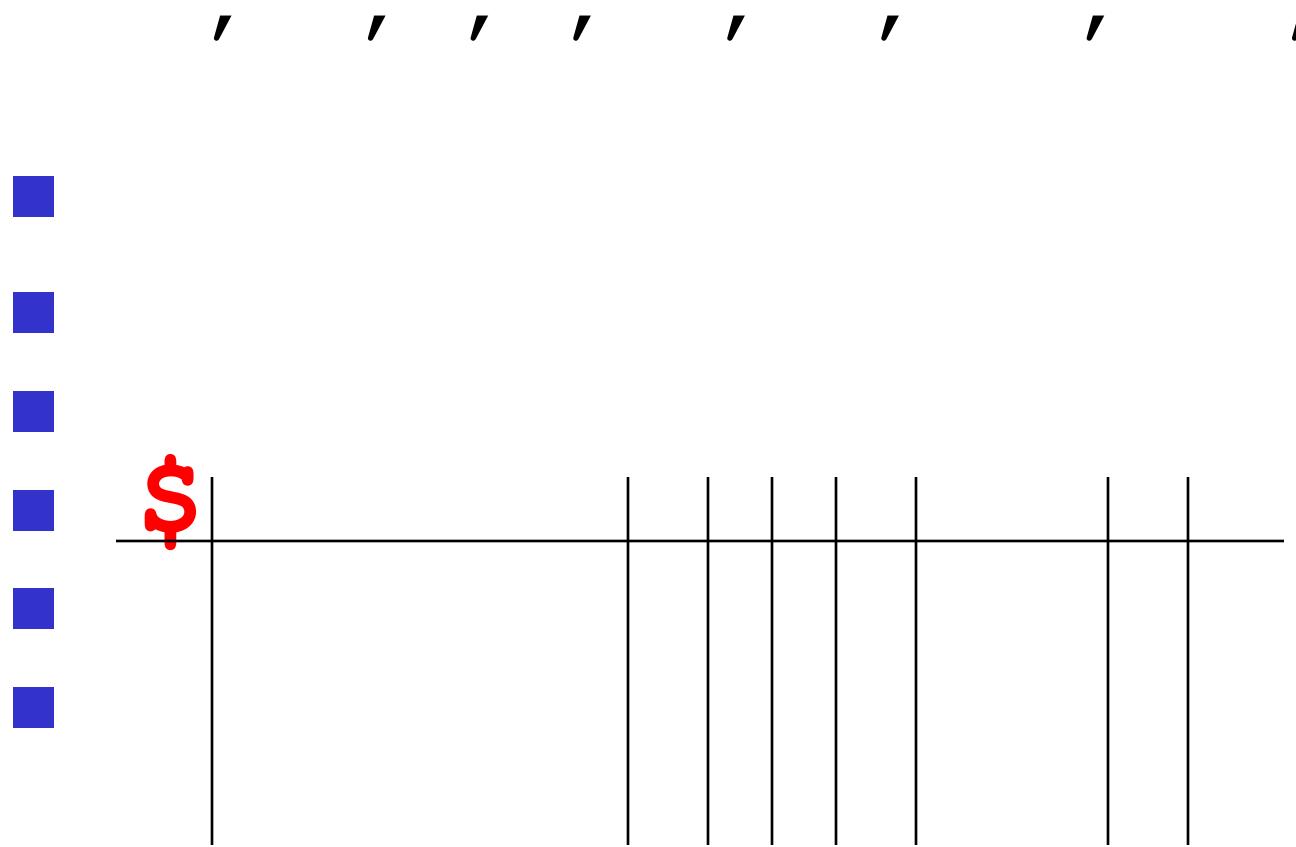
# Step – Construct m-Lists



# Step – Generate m-Lists



# Step – Sort S substrings



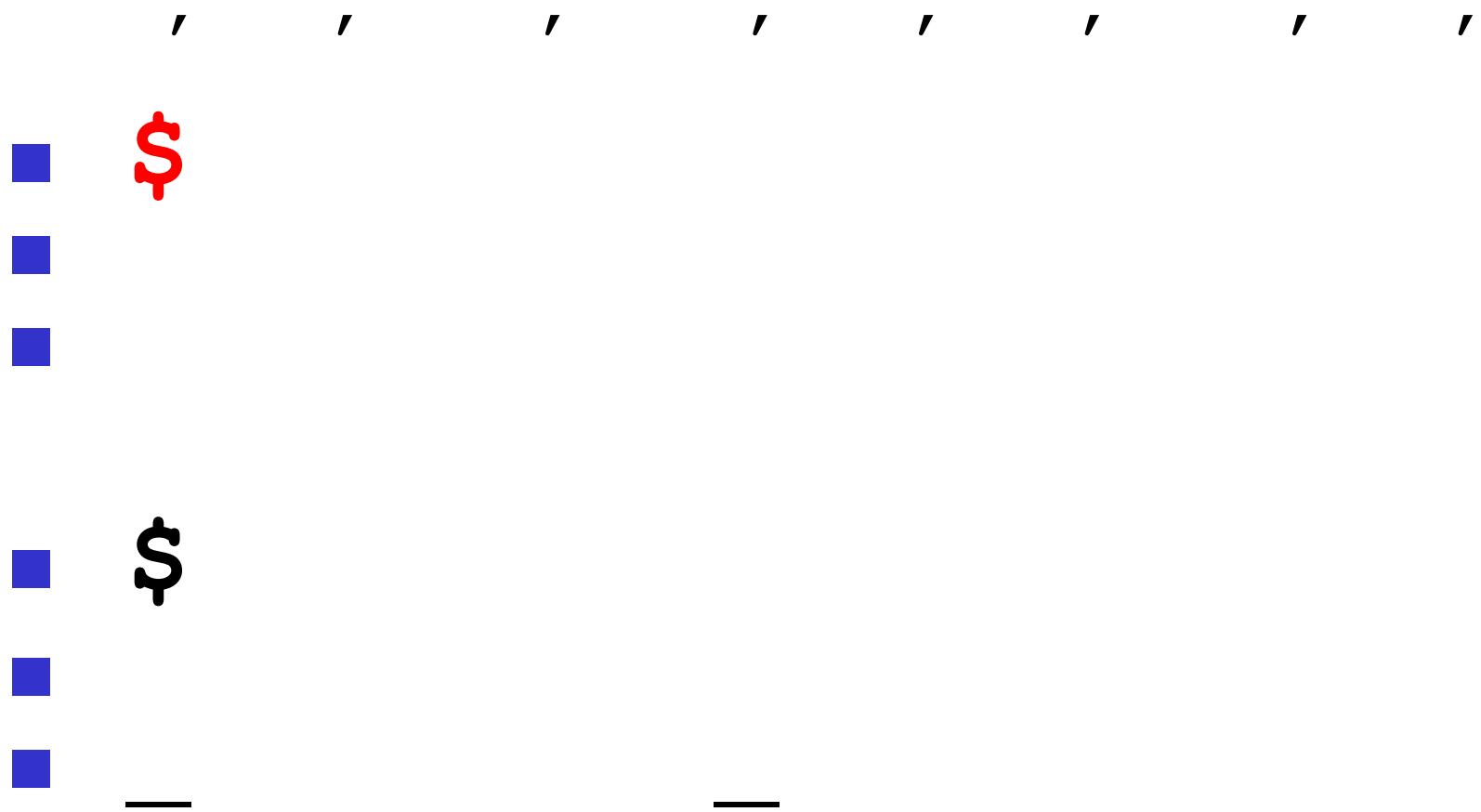
# Step – Sort S substrings



# Step – Sort S substrings



# Step – Generate the Suffix Array



# Step – Generate the Suffix Array

- \$
- 
- 
- \$
- 
- 
- 
-

# Step – Generate the Suffix Array

- A scatter plot with four data points marked by blue squares. The x-axis is labeled with 'A' and 'B'. The y-axis is labeled with 'C' and 'D'. A red dollar sign (\$) is positioned above the top-left data point.

# Step – Generate the Suffix Array

- \$
- 
- 
- 
- \$
- 
- 
- 
- 
-

# Step – Generate the Suffix Array

- \$



- \$



—

—



type S

# Step – Generate the Suffix Array

- \$
  - 
  - 
  - \$
  - 
  -
- —

# Step – Generate the Suffix Array

- \$
- 
- 
- \$
- 
- 

— —

# Final answer

- \$
- 
- 
- 
- 
- 
- 
-

# Final answer

- \$
- 
- 
- 
- 
- 
- 

**What is the BWT(S) ?**

# BWT is easy!

- \$

# BWT construction in linear time

