Faculty of Information Technology, Monash University

FIT2004: Algorithms and Data Structures

Week 7: Burrows-Wheeler Transform

These slides are prepared by M. A. Cheema and are based on the material developed by Arun Konagurthu and Lloyd Allison.

Outline

1. Compression
2. Burrows-Wheeler Transform (BWT)
3. Why BWT is effective for compression
4. Decompressing BWT
   A. Naïve Approach
   B. Efficient Approach
5. Substring search using BWT

Compression problem

Suppose you have a large sequence of characters (e.g., English text or DNA sequence). How can you compress the data?

Idea:

Original Text: this is mississippi's history. is this mississippi's history?

Modified: (rearrange such that we get many "runs" of the same characters)

hhhhiiiiooiiiiiiiiiittttmmsssssssssssrrppppyysssss (text length: 50)

Compressed: 4h4i2o10i4t2m11s2r4p2y5s (compressed length: 24)

Compression problem

- Sorting the text provides "runs" of maximal lengths.
  - hhhhiiiiiiiiiiiiiimmooopppprrsssssssssssssssssttttyy (text length: 50)
  - 4h14i2m2o4p2r16s4t2y (Compressed length: 20)
- However, sorting is not an acceptable solution! We must be able to recover the original text from the compressed data, i.e., decompression.

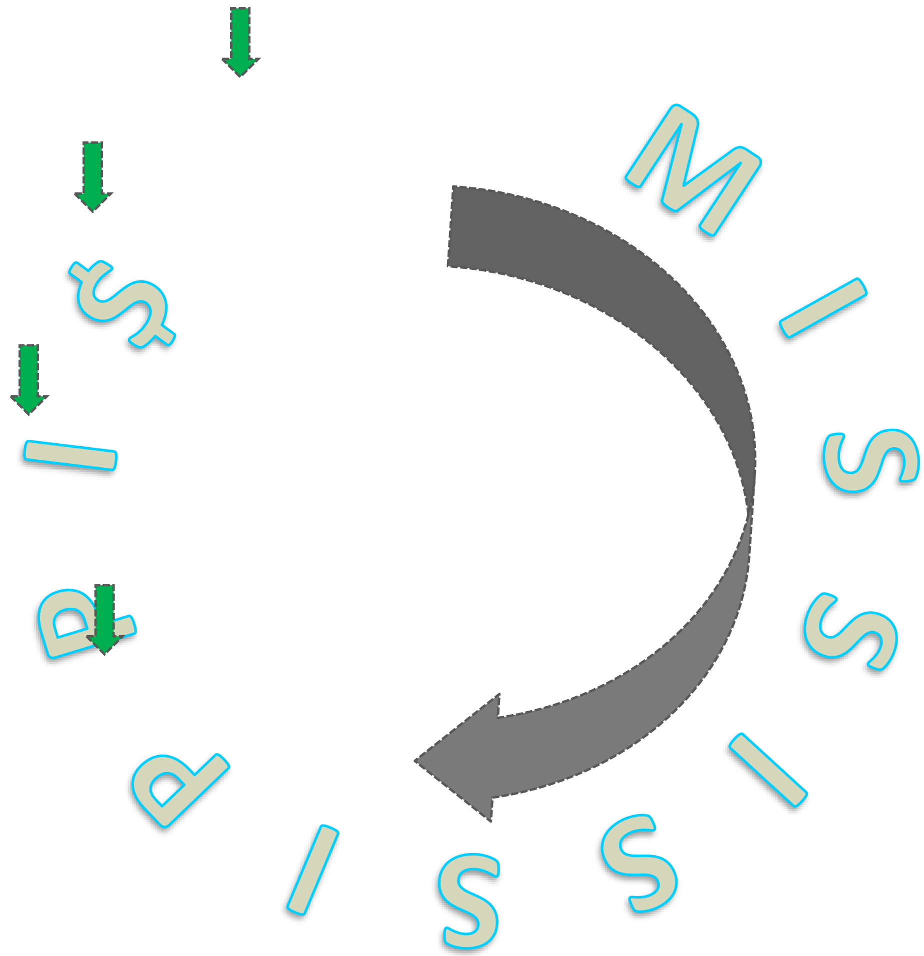
- So, the question is how to modify the original text such that there are many "runs" of the characters (to effectively compress the data) and the original text can be recovered from the decompressed data.
- Burrows-Wheeler Transform! Used in bzip2.

Outline

1. Compression
2. Burrows-Wheeler Transform (BWT)
3. Why BWT is effective for compression
4. Decompressing BWT
   A. Naïve Approach
   B. Efficient Approach
5. Substring search using BWT

Burrows-Wheeler Transform

M I S S I S S I P P I $
$ M I S S I S S I P P I
I $ M I S S I S S I P P
P I $ M I S S I S S I P
P P I $ M I S S I S S I
I P P I $ M I S S I S S
S I P P I $ M I S S I S
S S I P P I $ M I S S I
I S S I P P I $ M I S S
S I S S I P P I $ M I S
S S I S S I P P I $ M I
I S S I S S I P P I $ M



All cyclic rotations of the text

M	I	S	S	I	S	S	I	P	P	I	$
$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
I	P	P	I	$	M	I	S	S	I	S	S
S	I	P	P	I	$	M	I	S	S	I	S
S	S	I	P	P	I	$	M	I	S	S	I
I	S	S	I	P	P	I	$	M	I	S	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	S	S	I	P	P	I	$	M	I
I	S	S	I	S	S	I	P	P	I	$	M

All cyclic rotations of the text

$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
I	P	P	I	$	M	I	S	S	I	S	S
I	S	S	I	P	P	I	$	M	I	S	S
I	S	S	I	S	S	I	P	P	I	$	M
M	I	S	S	I	S	S	I	P	P	I	$
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
S	I	P	P	I	$	M	I	S	S	I	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	P	P	I	$	M	I	S	S	I
S	S	I	S	S	I	P	P	I	$	M	I

Sort the strings in alphabetical order assuming $ is the smallest

M	I	S	S	I	S	S	I	P	P	I	$
$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
I	P	P	I	$	M	I	S	S	I	S	S
S	I	P	P	I	$	M	I	S	S	I	S
S	S	I	P	P	I	$	M	I	S	S	I
I	S	S	I	P	P	I	$	M	I	S	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	S	S	I	P	P	I	$	M	I
I	S	S	I	S	S	I	P	P	I	$	M

All cyclic rotations of the text

$	M	I	S	S	I	S	S	I	P	P	**I**
I	$	M	I	S	S	I	S	S	I	P	**P**
I	P	P	I	$	M	I	S	S	I	S	**S**
I	S	S	I	P	P	I	$	M	I	S	**S**
I	S	S	I	S	S	I	P	P	I	$	**M**
M	I	S	S	I	S	S	I	P	P	I	**$**
P	I	$	M	I	S	S	I	S	S	I	**P**
P	P	I	$	M	I	S	S	I	S	S	**I**
S	I	P	P	I	$	M	I	S	S	I	**S**
S	I	S	S	I	P	P	I	$	M	I	**S**
S	S	I	P	P	I	$	M	I	S	S	**I**
S	S	I	S	S	I	P	P	I	$	M	**I**

The last column of the sorted matrix is Burrows-Wheeler Transform

Note similarity with suffix array which corresponds to IDs of these suffixes/cyclic rotations

M	I	S	S	I	S	S	I	P	P	I	$
$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
I	P	P	I	$	M	I	S	S	I	S	S
S	I	P	P	I	$	M	I	S	S	I	S
S	S	I	P	P	I	$	M	I	S	S	I
I	S	S	I	P	P	I	$	M	I	S	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	S	S	I	P	P	I	$	M	I
I	S	S	I	S	S	I	P	P	I	$	M

All cyclic rotations of the text

$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
I	P	P	I	$	M	I	S	S	I	S	S
I	S	S	I	P	P	I	$	M	I	S	S
I	S	S	I	S	S	I	P	P	I	$	M
M	I	S	S	I	S	S	I	P	P	I	$
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
S	I	P	P	I	$	M	I	S	S	I	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	P	P	I	$	M	I	S	S	I
S	S	I	S	S	I	P	P	I	$	M	I

The last column of the sorted matrix is Burrows-Wheeler Transform

Once you get BWT, you can use run-length encoding to compress it (if the goal is compression).

M	I	S	S	I	S	S	I	P	P	I	$
$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
I	P	P	I	$	M	I	S	S	I	S	S
S	I	P	P	I	$	M	I	S	S	I	S
S	S	I	P	P	I	$	M	I	S	S	I
I	S	S	I	P	P	I	$	M	I	S	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	S	S	I	P	P	I	$	M	I
I	S	S	I	S	S	I	P	P	I	$	M

$	M	I	S	S	I	S	S	I	P	P	**I**
I	$	M	I	S	S	I	S	S	I	P	**P**
I	P	P	I	$	M	I	S	S	I	S	**S**
I	S	S	I	P	P	I	$	M	I	S	**S**
I	S	S	I	S	S	I	P	P	I	$	**M**
M	I	S	S	I	S	S	I	P	P	I	**$**
P	I	$	M	I	S	S	I	S	S	I	**P**
P	P	I	$	M	I	S	S	I	S	S	**I**
S	I	P	P	I	$	M	I	S	S	I	**S**
S	I	S	S	I	P	P	I	$	M	I	**S**
S	S	I	P	P	I	$	M	I	S	S	**I**
S	S	I	S	S	I	P	P	I	$	M	**I**

All cyclic rotations of the text

The last column of the sorted matrix is Burrows-Wheeler Transform

Exercise

What is the Burrows-Wheeler Transform of BIRD?

A. $BIRD
B. BI$RD
C. D$RBI
D. IRBD$
E. RDI$B
F. None of the above

Quiz time!
https://flux.qa - YTJMAZ

Outline

1. Compression
2. Burrows-Wheeler Transform (BWT)
3. Why BWT is effective for compression
4. Decompressing BWT
   A. Naïve Approach
   B. Efficient Approach
5. Substring search using BWT

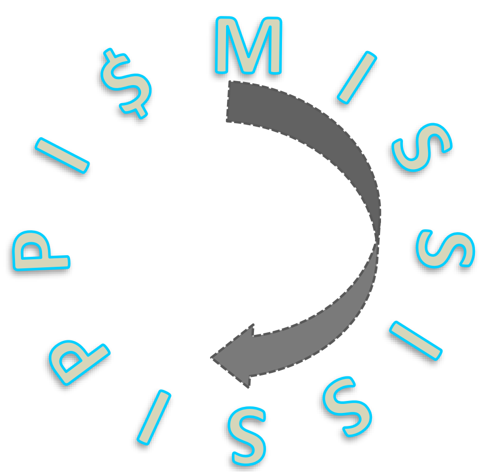
Why is BWT effective for compression?

Last-First Property:

The last character of a row comes before the first character of the row in the input string.

- because each string in the matrix is a cyclic rotation of the text

$	M	I	S	S	I	S	S	I	P	P	**I**
I	$	M	I	S	S	I	S	S	I	P	**P**
I	P	P	I	$	M	I	S	S	I	S	**S**
I	S	S	I	P	P	I	$	M	I	S	**S**
I	S	S	I	S	S	I	P	P	I	$	**M**
M	I	S	S	I	S	S	I	P	P	I	**$**
P	I	$	M	I	S	S	I	S	S	I	**P**
P	P	I	$	M	I	S	S	I	S	S	**I**
S	I	P	P	I	$	M	I	S	S	I	**S**
S	I	S	S	I	P	P	I	$	M	I	**S**
S	S	I	P	P	I	$	M	I	S	S	**I**
S	S	I	S	S	I	P	P	I	$	M	**I**



M	I	S	S	I	S	S	I	P	P	I	$
$	M	I	S	S	I	S	S	I	P	P	I
I	$	M	I	S	S	I	S	S	I	P	P
P	I	$	M	I	S	S	I	S	S	I	P
P	P	I	$	M	I	S	S	I	S	S	I
I	P	P	I	$	M	I	S	S	I	S	S
S	I	P	P	I	$	M	I	S	S	I	S
S	S	I	P	P	I	$	M	I	S	S	I
I	S	S	I	P	P	I	$	M	I	S	S
S	I	S	S	I	P	P	I	$	M	I	S
S	S	I	S	S	I	P	P	I	$	M	I
I	S	S	I	S	S	I	P	P	I	$	M

All cyclic rotations of the text

$	M	I	S	S	I	S	S	I	P	P	**I**
I	$	M	I	S	S	I	S	S	I	P	**P**
I	P	P	I	$	M	I	S	S	I	S	**S**
I	S	S	I	P	P	I	$	M	I	S	**S**
I	S	S	I	S	S	I	P	P	I	$	**M**
M	I	S	S	I	S	S	I	P	P	I	**$**
P	I	$	M	I	S	S	I	S	S	I	**P**
P	P	I	$	M	I	S	S	I	S	S	**I**
S	I	P	P	I	$	M	I	S	S	I	**S**
S	I	S	S	I	P	P	I	$	M	I	**S**
S	S	I	P	P	I	$	M	I	S	S	**I**
S	S	I	S	S	I	P	P	I	$	M	**I**

For each rotation, the char in the BWT comes **before** the first char

Why is BWT effective for compression?

- Consider a large English text. IS is a very common word. Thus, I appears before S in the text much more frequently compared to some other letters, e.g., IS is more frequent than CABS, BOSS etc.
- When the cyclic rotation matrix is sorted, all the occurrences of S in the first column appear together. The last column which is BWT will contain a lot of occurrences of Is because I appears before S much more frequently than the other letters.
- E.g., this-is-a-historical-story (space replaced with – for clarity)

  ………………………………
  s-a-historical-story$this-i
  s-is-a-historical-story$thi
  storical-story$this-is-a-hi
  story$this-is-a-historical-
  ………………………………

- Effective for compression when text is large and has such biases in it (i.e., some letters appear before some others much more frequently).

Outline

1. Compression
2. Burrows-Wheeler Transform (BWT)
3. Why BWT is effective for compression
4. Decompressing BWT
    A. Naïve Approach
    B. Efficient Approach
5. Substring search using BWT

Decompressing (Inverting) BWT

- We saw that BWT produces "runs" of characters which is effective in compression.
- But how do we invert BWT, i.e., how do we decompress the data to recover original text.
- If we could rebuild all the sorted permutations, the one that ends in $ is the original text

k-mers

k-mers of a string refers to its all possible substrings of size k (considering cyclic rotation).

- 2-mers of \$MISSISSIPPI are \$M, MI, IS, SS, SI, IS, SS, SI, IP, PP, PI, I\$.
- 3-mers of \$MISSISSIPPI are \$MI, MIS, ISS, SSI, SIS, ISS, SSI, SIP, IPP, PPI, PI\$, I\$M.

Which of the following represents 2-mers of BIRD\$.

A. D\$, RI, BI, RD, \$B
B. IR, D\$, BI, \$B, RD
C. \$B, DR, BI, IR, D\$
D. \$D, DR, RI, IB, B\$
E. None of the above

Quiz time!
https://flux.qa - YTJMAZ

Inverting BWT

$ M I S S I S S I P P	**I**		$
I $ M I S S I S S I P	**P**		I
I P P I $ M I S S I S	**S**		I
I S S I P P I $ M I S	**S**		I
I S S I S S I P P I $	**M**		I
M I S S I S S I P P I	**$**	→ sort	M
P I $ M I S S I S S I	**P**		P
P P I $ M I S S I S S	**I**		P
S I P P I $ M I S S I	**S**		S
S I S S I P P I $ M I	**S**		S
S S I P P I $ M I S S	**I**		S
S S I S S I P P I $ M	**I**		S

The first column is the last column, sorted

Inverting BWT

$	M	I	S	S	I	S	S	I	P	P	**I**
I	$	M	I	S	S	I	S	S	I	P	**P**
I	P	P	I	$	M	I	S	S	I	S	**S**
I	S	S	I	P	P	I	$	M	I	S	**S**
I	S	S	I	S	S	I	P	P	I	$	**M**
M	I	S	S	I	S	S	I	P	P	I	**$**
P	I	$	M	I	S	S	I	S	S	I	**P**
P	P	I	$	M	I	S	S	I	S	S	**I**
S	I	P	P	I	$	M	I	S	S	I	**S**
S	I	S	S	I	P	P	I	$	M	I	**S**
S	S	I	P	P	I	$	M	I	S	S	**I**
S	S	I	S	S	I	P	P	I	$	M	**I**

The first column is the last column, sorted

The letters in the first column come after the letter in the last column

Inverting BWT

```
$ M I S S I S S I P P I
I $ M I S S I S S I P P
I P P I $ M I S S I S S
I S S I P P I $ M I S S
I S S I S S I P P I $ M
M I S S I S S I P P I $
P I $ M I S S I S S I P
P P I $ M I S S I S S I
S I P P I $ M I S S I S
S I S S I P P I $ M I S
S S I P P I $ M I S S I
S S I S S I P P I $ M I
```

The first column is the last column, sorted

The letters in the first column come after the letter in the last column

Concatenating the first column with the last column gives us all 2-mers

Inverting BWT

$ M I S S I S S I P P	**I**	I	$
I $ M I S S I S S I P	**P**	P	I
I P P I $ M I S S I S	**S**	S	I
I S S I P P I $ M I S	**S**	S	I
I S S I S S I P P I $	**M**	M	I
M I S S I S S I P P I	**$**	$	M
P I $ M I S S I S S I	**P**	P	P
P P I $ M I S S I S S	**I**	I	P
S I P P I $ M I S S I	**S**	S	S
S I S S I P P I $ M I	**S**	S	S
S S I P P I $ M I S S	**I**	I	S
S S I S S I P P I $ M	**I**	I	S

The first column is the last column, sorted

The letters in the first column come after the letter in the last column

Concatenating last column + first column gives us all 2-mers

Inverting BWT

$ M I S S I S S I P P	**I**	I	$
I $ M I S S I S S I P	**P**	P	I
I P P I $ M I S S I S	**S**	S	I
I S S I P P I $ M I S	**S**	S	I
I S S I S S I P P I $	**M**	M	I
M I S S I S S I P P I	**$**	$	M
P I $ M I S S I S S I	**P**	P	P
P P I $ M I S S I S S	**I**	I	P
S I P P I $ M I S S I	**S**	S	S
S I S S I P P I $ M I	**S**	S	S
S S I P P I $ M I S S	**I**	I	S
S S I S S I P P I $ M	**I**	I	S

The first column is the last column, sorted

The letters in the first column come after the letter in the last column

Concatenating the first column with the last column gives us all 2-mers

Sorting the 2-mers gives us the first 2 columns

Inverting BWT

$ M I S S I S S I P P	**I**	I $		$ M
I $ M I S S I S S I P	**P**	P I		I $
I P P I $ M I S S I S	**S**	S I		I P
I S S I P P I $ M I S	**S**	S I		I S
I S S I S S I P P I $	**M**	M I		I S
M I S S I S S I P P I	**$**	$ M	sort	M I
P I $ M I S S I S S I	**P**	P P		P I
P P I $ M I S S I S S	**I**	I P		P P
S I P P I $ M I S S I	**S**	S S		S I
S I S S I P P I $ M I	**S**	S S		S I
S S I P P I $ M I S S	**I**	I S		S S
S S I S S I P P I $ M	**I**	I S		S S

Inverting BWT

$ M I S S I S S I P P	**I**	$ M
I $ M I S S I S S I P	**P**	I $
I P P I $ M I S S I S	**S**	I P
I S S I P P I $ M I S	**S**	I S
I S S I S S I P P I $	**M**	I S
M I S S I S S I P P I	**$**	M I
P I $ M I S S I S S I	**P**	P I
P P I $ M I S S I S S	**I**	P P
S I P P I $ M I S S I	**S**	S I
S I S S I P P I $ M I	**S**	S I
S S I P P I $ M I S S	**I**	S S
S S I S S I P P I $ M	**I**	S S

We have all sorted 2-mers (first 2 columns)

We want all the 3-mers, so we can sort them and get the first 3 columns

Last column comes before first column

Concatenate again!

Inverting BWT

$ M I S S I S S I P P **I**	**I**	$ M	I $ M		$ M I
I $ M I S S I S S I P **P**	**P**	I $	P I $		I $ M
I P P I $ M I S S I S **S**	**S**	I P	S I P		I P P
I S S I P P I $ M I S **S**	**S**	I S	S I S		I S S
I S S I S S I P P I $ **M**	**M**	I S	M I S		I S S
M I S S I S S I P P I **$**	**$** +	M I =	$ M I	sort	M I S
P I $ M I S S I S S I **P**	**P**	P I	P P I		P I $
P P I $ M I S S I S S **I**	**I**	P P	I P P		P P I
S I P P I $ M I S S I **S**	**S**	S I	S S I		S I P
S I S S I P P I $ M I **S**	**S**	S I	S S I		S I S
S S I P P I $ M I S S **I**	**I**	S S	I S S		S S I
S S I S S I P P I $ M **I**	**I**	S S	I S S		S S I

Inverting BWT

$ M I S S I S S I P P **I**	$ M I
I $ M I S S I S S I P **P**	I $ M
I P P I $ M I S S I S **S**	I P P
I S S I P P I $ M I S **S**	I S S
I S S I S S I P P I $ **M**	I S S
M I S S I S S I P P I **$**	M I S
P I $ M I S S I S S I **P**	P I $
P P I $ M I S S I S S **I**	P P I
S I P P I $ M I S S I **S**	S I P
S I S S I P P I $ M I **S**	S I S
S S I P P I $ M I S S **I**	S S I
S S I S S I P P I $ M **I**	S S I

Now we have sorted 3-mers (first 3 columns)

Keep

- prepending the last column to the sorted k-mers
- sorting the result

until you have reconstructed all the sorted cyclic rotations

Return first row without $

Inverting BWT

Inverting BWT

Create an empty table M

Make a column C containing BWT

Repeat len(BWT) times

    Concatenate C with M

    Sort C||M alphabetically

Return the first row (ignore $).

Let N be the total number of characters in the original string. What is the complexity?

Time complexity:

Requires N calls to sorting

Cost of sorting N rows where each row has N characters: $O(N^2)$ using radix sort

Total cost for sorting: $O(N^3)$ if radix sort is being used

Space complexity:

Size of matrix: $O(N^2)$

Can we improve?

Yes! It is possible to invert in O(N) time complexity and O(N) space complexity

Outline

1. Compression
2. Burrows-Wheeler Transform (BWT)
3. Why BWT is effective for compression
4. Decompressing BWT
   A. Naïve Approach
   B. Efficient Approach
5. Substring search using BWT

Faster Inversion of BWT

1	$ M I S S I S S I P P I
2	I $ M I S S I S S I P P
3	I P P I $ M I S S I S S
4	I S S I P P I $ M I S S
5	I S S I S S I P P I $ M
6	M I S S I S S I P P I $
7	P I $ M I S S I S S I P
8	P P I $ M I S S I S S I
9	S I P P I $ M I S S I S
10	S I S S I P P I $ M I S
11	S S I P P I $ M I S S I
12	S S I S S I P P I $ M I

$ M I S S I S S I P P I

We will use different colors for different occurrences of S in $MISSISSIPPI.

Quiz time!
https://flux.qa - YTJMAZ

Faster Inversion of BWT

1	$	M	I	S	S	I	S	S	I	P	P	I
2	I	$	M	I	S	S	I	S	S	I	P	P
3	I	P	P	I	$	M	I	S	S	I	S	S
4	I	S	S	I	P	P	I	$	M	I	S	S
5	I	S	S	I	S	S	I	P	P	I	$	M
6	M	I	S	S	I	S	S	I	P	P	I	$
7	P	I	$	M	I	S	S	I	S	S	I	P
8	P	P	I	$	M	I	S	S	I	S	S	I
9	S	I	P	P	I	$	M	I	S	S	I	S
10	S	I	S	S	I	P	P	I	$	M	I	S
11	S	S	I	P	P	I	$	M	I	S	S	I
12	S	S	I	S	S	I	P	P	I	$	M	I

$ M I S S I S S I P P I

We have used different colors for different occurrences of S in $MISSISSIPPI.

Faster Inversion of BWT

1	**$**	M	I	S	S	I	S	S	I	P	P	**I**
2	**I**	$	M	I	S	S	I	S	S	I	P	**P**
3	**I**	P	P	I	$	M	I	S	S	I	S	**S**
4	**I**	S	S	I	P	P	I	$	M	I	S	**S**
5	**I**	S	S	I	S	S	I	P	P	I	$	**M**
6	**M**	I	S	S	I	S	S	I	P	P	I	**$**
7	**P**	I	$	M	I	S	S	I	S	S	I	**P**
8	**P**	P	I	$	M	I	S	S	I	S	S	**I**
9	**S**	I	P	P	I	$	M	I	S	S	I	**S**
10	**S**	I	S	S	I	P	P	I	$	M	I	**S**
11	**S**	S	I	P	P	I	$	M	I	S	S	**I**
12	**S**	S	I	S	S	I	P	P	I	$	M	**I**

$ M I S S I S S I P P I

We have used different colors for different occurrences of S in $MISSISSIPPI.

Observation

The relative orders of the **same** characters in the first column and the last column is the same.

Faster Inversion of BWT

1	$	M	I	S	S	I	S	S	I	P	P	I
2	I	$	M	I	S	S	I	S	S	I	P	P
3	I	P	P	I	$	M	I	S	S	I	S	S
4	I	S	S	I	P	P	I	$	M	I	S	S
5	I	S	S	I	S	S	I	P	P	I	$	M
6	M	I	S	S	I	S	S	I	P	P	I	$
7	P	I	$	M	I	S	S	I	S	S	I	P
8	P	P	I	$	M	I	S	S	I	S	S	I
9	S	I	P	P	I	$	M	I	S	S	I	S
10	S	I	S	S	I	P	P	I	$	M	I	S
11	S	S	I	P	P	I	$	M	I	S	S	I
12	S	S	I	S	S	I	P	P	I	$	M	I

$ M I S S I S S I P P I

We have used different colors for different occurrences of S in $MISSISSIPPI.

Observation

The relative orders of the **same** characters in the first column and the last column is the same.

E.g., the i-th S in the first column is the i-th S in the last column

Faster Inversion of BWT

\$	M	I	S	S	I	S	S	I	P	P	I_1
I_1	\$	M	I	S	S	I	S	S	I	P	P_1
I_2	P	P	I	\$	M	I	S	S	I	S	S_1
I_3	S	S	I	P	P	I	\$	M	I	S	S_2
I_4	S	S	I	S	S	I	P	P	I	\$	M_1
M_1	I	S	S	I	S	S	I	P	P	I	\$
P_1	I	\$	M	I	S	S	I	S	S	I	P_2
P_2	P	I	\$	M	I	S	S	I	S	S	I_2
S_1	I	P	P	I	\$	M	I	S	S	I	S_3
S_2	I	S	S	I	P	P	I	\$	M	I	S_4
S_3	S	I	P	P	I	\$	M	I	S	S	I_3
S_4	S	I	S	S	I	P	P	I	\$	M	I_4

i-th occurrence of a letter in first column and i-th occurrence of the letter in the last column point to the same letter.

Faster Inversion of BWT

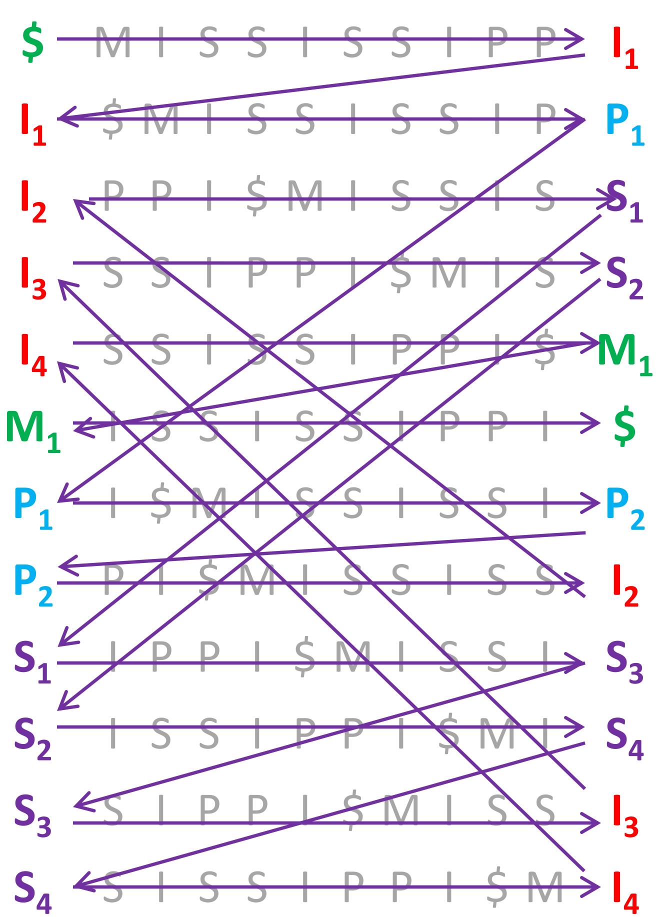
```
$ M I S S I S S I P P I
I $ M I S S I S S I P P
I P P I $ M I S S I S S
I S S I P P I $ M I S S
I S S I S S I P P I $ M
M I S S I S S I P P I $
P I $ M I S S I S S I P
P P I $ M I S S I S S I
S I P P I $ M I S S I S
S I S S I P P I $ M I S
S S I P P I $ M I S S I
S S I S S I P P I $ M I
```

Why does this observation hold?

- Rotate each row that ends at S by one character
- First characters of all these are the same (i.e., S)
- This means the sorting is based on the remaining characters, i.e., the sorting order is determined by stripping off S.
- Hence, the row that appeared earlier before rotation must appear earlier after rotation.

```
S I P P I $ M I S S I S
S I S S I P P I $ M I S
S S I P P I $ M I S S I
S S I S S I P P I $ M I
```

Faster Inversion of BWT



- So, we know which character in the last column corresponds to which character in the first column. The inversion can then be done as follows.
- Start from $ in the first column (F)
- The previous letter in this row **I** is the letter before $ in the original string (Last-First property). Recover this letter.
- Now, find this **I** in the first column
- The previous letter in this row **P** is the letter before this **I** in the original string (Last-First property). Recover this letter
- Now, find this P in the first column.
- The previous letter in this row P is the letter before this P in the original string (Last-First property). Recover it.
- and so on …

M I S S I S S I P P I $

Faster Inversion of BWT

$\$$	M I S S I S S I P P	I_1
I_1	$\$$ M I S S I S S I P	P_1
I_2	P P I $\$$ M I S S I S	S_1
I_3	S S I P P I $\$$ M I S	S_2
I_4	S S I S S I P P I $\$$	M_1
M_1	I S S I S S I P P I	$\$$
P_1	I $\$$ M I S S I S S I	P_2
P_2	P I $\$$ M I S S I S S	I_2
S_1	I P P I $\$$ M I S S I	S_3
S_2	I S S I P P I $\$$ M I	S_4
S_3	S I P P I $\$$ M I S S	I_3
S_4	S I S S I P P I $\$$ M	I_4

- Each step should be O(1)
- What do we need to know?

Quiz time!
https://flux.qa - YTJMAZ

Faster Inversion of BWT

$\$$	M I S S I S S I P P	I_1
I_1	$ M I S S I S S I P	P_1
I_2	P P I $ M I S S I S	S_1
I_3	S S I P P I $ M I S	S_2
I_4	S S I S S I P P I $	M_1
M_1	I S S I S S I P P I	$\$$
P_1	I $ M I S S I S S I	P_2
P_2	P I $ M I S S I S S	I_2
S_1	I P P I $ M I S S I	S_3
S_2	I S S I P P I $ M I	S_4
S_3	S I P P I $ M I S S	I_3
S_4	S I S S I P P I $ M	I_4

- Each step should be O(1)
- What do we need to know?

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

- How can we collect this information quickly?

Faster Inversion of BWT

\$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	\$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	\$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	\$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	\$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**\$**
P_1	I	\$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	\$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	\$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	\$	M	I	**S_4**
S_3	S	I	P	P	I	\$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	\$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
\$	I	M	P	S

Rank

0	0	0	0	0
\$	I	M	P	S

Count

Faster Inversion of BWT

$	M I S S I S S I P P	I_1
I_1	$ M I S S I S S I P	P_1
I_2	P P I $ M I S S I S	S_1
I_3	S S I P P I $ M I S	S_2
I_4	S S I S S I P P I $	M_1
M_1	I S S I S S I P P I	$
P_1	I $ M I S S I S S I	P_2
P_2	P I $ M I S S I S S	I_2
S_1	I P P I $ M I S S I	S_3
S_2	I S S I P P I $ M I	S_4
S_3	S I P P I $ M I S S	I_3
S_4	S I S S I P P I $ M	I_4

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

0	1	0	0	0
$	I	M	P	S

Count

Faster Inversion of BWT

$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**$**
P_1	I	$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	$	M	I	**S_4**
S_3	S	I	P	P	I	$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

0	1	0	1	0
$	I	M	P	S

Count

Faster Inversion of BWT

\$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	\$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	\$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	\$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	\$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**\$**
P_1	I	\$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	\$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	\$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	\$	M	I	**S_4**
S_3	S	I	P	P	I	\$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	\$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
\$	I	M	P	S

Rank

0	1	0	1	1
\$	I	M	P	S

Count

Faster Inversion of BWT

$	M I S S I S S I P P	I_1
I_1	$ M I S S I S S I P	P_1
I_2	P P I $ M I S S I S	S_1
I_3	S S I P P I $ M I S	S_2
I_4	S S I S S I P P I $	M_1
M_1	I S S I S S I P P I	$
P_1	I $ M I S S I S S I	P_2
P_2	P I $ M I S S I S S	I_2
S_1	I P P I $ M I S S I	S_3
S_2	I S S I P P I $ M I	S_4
S_3	S I P P I $ M I S S	I_3
S_4	S I S S I P P I $ M	I_4

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

0	1	0	1	2
$	I	M	P	S

Count

Faster Inversion of BWT

$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**$**
P_1	I	$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	$	M	I	**S_4**
S_3	S	I	P	P	I	$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

0	1	1	1	2
$	I	M	P	S

Count

Faster Inversion of BWT

$	M I S S I S S I P P	I_1
I_1	$ M I S S I S S I P	P_1
I_2	P P I $ M I S S I S	S_1
I_3	S S I P P I $ M I S	S_2
I_4	S S I S S I P P I $	M_1
M_1	I S S I S S I P P I	$
P_1	I $ M I S S I S S I	P_2
P_2	P I $ M I S S I S S	I_2
S_1	I P P I $ M I S S I	S_3
S_2	I S S I P P I $ M I	S_4
S_3	S I P P I $ M I S S	I_3
S_4	S I S S I P P I $ M	I_4

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

1	1	1	1	2
$	I	M	P	S

Count

Faster Inversion of BWT

$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**$**
P_1	I	$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	$	M	I	**S_4**
S_3	S	I	P	P	I	$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

1	1	1	2	2
$	I	M	P	S

Count

Faster Inversion of BWT

$	M I S S I S S I P P	I_1
I_1	$ M I S S I S S I P	P_1
I_2	P P I $ M I S S I S	S_1
I_3	S S I P P I $ M I S	S_2
I_4	S S I S S I P P I $	M_1
M_1	I S S I S S I P P I	$
P_1	I $ M I S S I S S I	P_2
P_2	P I $ M I S S I S S	I_2
S_1	I P P I $ M I S S I	S_3
S_2	I S S I P P I $ M I	S_4
S_3	S I P P I $ M I S S	I_3
S_4	S I S S I P P I $ M	I_4

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	0
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

1	2	1	2	2
$	I	M	P	S

Count

Faster Inversion of BWT

$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**$**
P_1	I	$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	$	M	I	**S_4**
S_3	S	I	P	P	I	$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	0
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

1	2	1	2	3
$	I	M	P	S

Count

Faster Inversion of BWT

$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**$**
P_1	I	$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	$	M	I	**S_4**
S_3	S	I	P	P	I	$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	0
12	0

0	0	0	0	0
$	I	M	P	S

Rank

1	2	1	2	4
$	I	M	P	S

Count

Faster Inversion of BWT

$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**$**
P_1	I	$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	$	M	I	**S_4**
S_3	S	I	P	P	I	$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	0

0	0	0	0	0
$	I	M	P	S

Rank

1	3	1	2	4
$	I	M	P	S

Count

Faster Inversion of BWT

$	M I S S I S S I P P	**I_1**
I_1	$ M I S S I S S I P	**P_1**
I_2	P P I $ M I S S I S	**S_1**
I_3	S S I P P I $ M I S	**S_2**
I_4	S S I S S I P P I $	**M_1**
M_1	I S S I S S I P P I	**$**
P_1	I $ M I S S I S S I	**P_2**
P_2	P I $ M I S S I S S	**I_2**
S_1	I P P I $ M I S S I	**S_3**
S_2	I S S I P P I $ M I	**S_4**
S_3	S I P P I $ M I S S	**I_3**
S_4	S I S S I P P I $ M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

0	0	0	0	0
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Faster Inversion of BWT

$	M I S S I S S I P P	I_1
I_1	$ M I S S I S S I P	P_1
I_2	P P I $ M I S S I S	S_1
I_3	S S I P P I $ M I S	S_2
I_4	S S I S S I P P I $	M_1
M_1	I S S I S S I P P I	$
P_1	I $ M I S S I S S I	P_2
P_2	P I $ M I S S I S S	I_2
S_1	I P P I $ M I S S I	S_3
S_2	I S S I P P I $ M I	S_4
S_3	S I P P I $ M I S S	I_3
S_4	S I S S I P P I $ M	I_4

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	0	0	0	0
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Faster Inversion of BWT

\$	M I S S I S S I P P	I_1
I_1	\$ M I S S I S S I P	P_1
I_2	P P I \$ M I S S I S	S_1
I_3	S S I P P I \$ M I S	S_2
I_4	S S I S S I P P I \$	M_1
M_1	I S S I S S I P P I	**\$**
P_1	I \$ M I S S I S S I	P_2
P_2	P I \$ M I S S I S S	I_2
S_1	I P P I \$ M I S S I	S_3
S_2	I S S I P P I \$ M I	S_4
S_3	S I P P I \$ M I S S	I_3
S_4	S I S S I P P I \$ M	I_4

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	0	0	0
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Faster Inversion of BWT

\$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	\$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	\$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	\$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	\$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**\$**
P_1	I	\$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	\$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	\$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	\$	M	I	**S_4**
S_3	S	I	P	P	I	\$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	\$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	6	0	0
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Faster Inversion of BWT

\$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	\$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	\$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	\$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	\$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**\$**
P_1	I	\$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	\$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	\$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	\$	M	I	**S_4**
S_3	S	I	P	P	I	\$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	\$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	6	7	0
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Faster Inversion of BWT

\$	M	I	S	S	I	S	S	I	P	P	**I_1**
I_1	\$	M	I	S	S	I	S	S	I	P	**P_1**
I_2	P	P	I	\$	M	I	S	S	I	S	**S_1**
I_3	S	S	I	P	P	I	\$	M	I	S	**S_2**
I_4	S	S	I	S	S	I	P	P	I	\$	**M_1**
M_1	I	S	S	I	S	S	I	P	P	I	**\$**
P_1	I	\$	M	I	S	S	I	S	S	I	**P_2**
P_2	P	I	\$	M	I	S	S	I	S	S	**I_2**
S_1	I	P	P	I	\$	M	I	S	S	I	**S_3**
S_2	I	S	S	I	P	P	I	\$	M	I	**S_4**
S_3	S	I	P	P	I	\$	M	I	S	S	**I_3**
S_4	S	I	S	S	I	P	P	I	\$	M	**I_4**

- Position of each block of letters in left column
- Which letter we are looking at in right column (i.e. which S are we looking at?)

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	6	7	9
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Faster Inversion of BWT

1	$	~~M~~	~~I~~	~~S~~	~~S~~	~~I~~	~~S~~	~~S~~	~~I~~	~~P~~	~~P~~	I_1
2	I_1	$	M	I	S	S	I	S	S	I	P	P_1
3	I_2	P	P	I	$	M	I	S	S	I	S	S_1
4	I_3	S	S	I	P	P	I	$	M	I	S	S_2
5	I_4	S	S	I	S	S	I	P	P	I	$	M_1
6	M_1	I	S	S	I	S	S	I	P	P	I	$
7	P_1	I	$	M	I	S	S	I	S	S	I	P_2
8	P_2	P	I	$	M	I	S	S	I	S	S	I_2
9	S_1	I	P	P	I	$	M	I	S	S	I	S_3
10	S_2	I	S	S	I	P	P	I	$	M	I	S_4
11	S_3	S	I	P	P	I	$	M	I	S	S	I_3
12	S_4	S	I	S	S	I	P	P	I	$	M	I_4

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	6	7	9
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Faster Inversion of BWT

1	$	~~M~~	~~I~~	~~S~~	~~S~~	~~I~~	~~S~~	~~S~~	~~I~~	~~P~~	~~P~~	→	I_1
2	I_1	$	M	I	S	S	I	S	S	I	P		P_1
3	I_2	P	P	I	$	M	I	S	S	I	S		S_1
4	I_3	S	S	I	P	P	I	$	M	I	S		S_2
5	I_4	S	S	I	S	S	I	P	P	I	$		M_1
6	M_1	I	S	S	I	S	S	I	P	P	I		$
7	P_1	I	$	M	I	S	S	I	S	S	I		P_2
8	P_2	P	I	$	M	I	S	S	I	S	S		I_2
9	S_1	I	P	P	I	$	M	I	S	S	I		S_3
10	S_2	I	S	S	I	P	P	I	$	M	I		S_4
11	S_3	S	I	P	P	I	$	M	I	S	S		I_3
12	S_4	S	I	S	S	I	P	P	I	$	M		I_4

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	6	7	9
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Faster Inversion of BWT

1	\$	M I S S I S S I P P	I_1
2	I_1	\$ M I S S I S S I P	P_1
3	I_2	P P I \$ M I S S I S	S_1
4	I_3	S S I P P I \$ M I S	S_2
5	I_4	S S I S S I P P I \$	M_1
6	M_1	I S S I S S I P P I	\$
7	P_1	I \$ M I S S I S S I	P_2
8	P_2	P I \$ M I S S I S S	I_2
9	S_1	I P P I \$ M I S S I	S_3
10	S_2	I S S I P P I \$ M I	S_4
11	S_3	S I P P I \$ M I S S	I_3
12	S_4	S I S S I P P I \$ M	I_4

Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

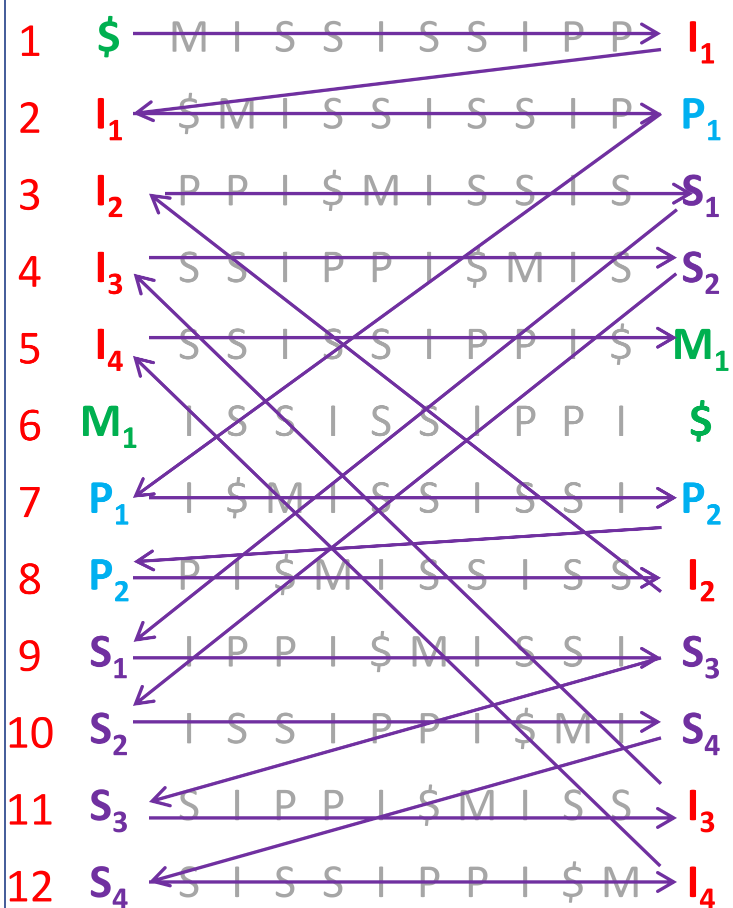
1	2	6	7	9
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Faster Inversion of BWT



Occ

1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	1
9	2
10	3
11	2
12	3

1	2	6	7	9
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Practice

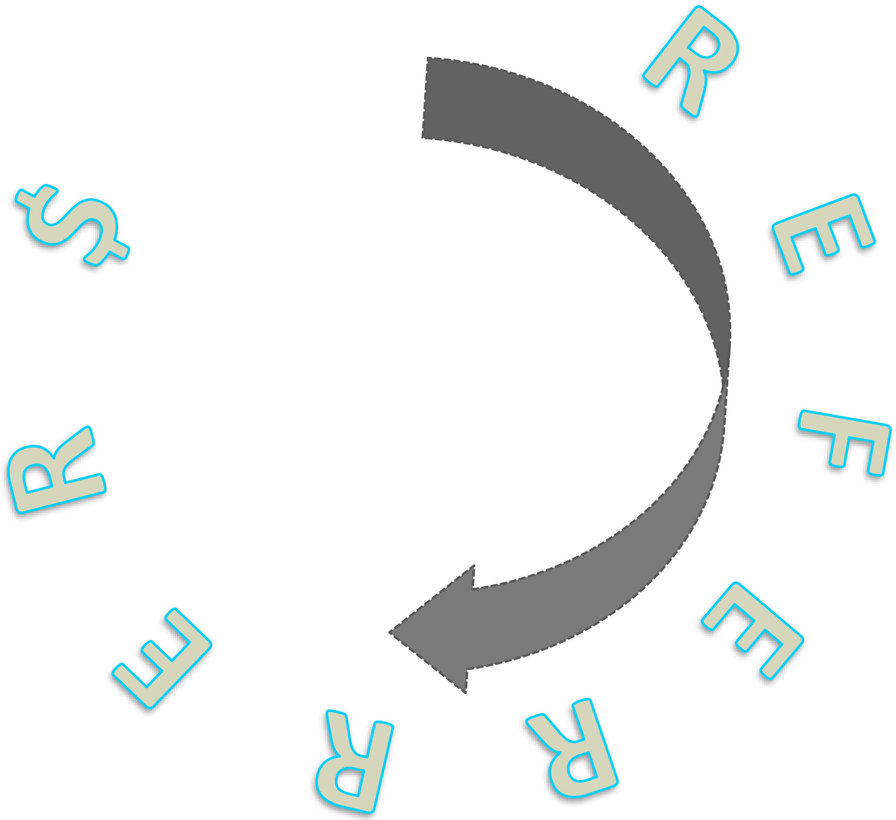
What is Burrows-Wheeler Transform of REFERRER?

A. RRRFEE$RE
B. $REFERRER
C. RRRFE$ERE
D. RRREFEE$R
E. None of the above

Quiz time!
https://flux.qa - YTJMAZ

Practice: Burrows-Wheeler Transform

R E F E R R E R $
$ R E F E R R E R
R $ R E F E R R E
E R $ R E F E R R
R E R $ R E F E R
R R E R $ R E F E
E R R E R $ R E F
F E R R E R $ R E
E F E R R E R $ R



All cyclic rotations of the text

Practice: Burrows-Wheeler Transform

R	E	F	E	R	R	E	R	$
$	R	E	F	E	R	R	E	R
R	$	R	E	F	E	R	R	E
E	R	$	R	E	F	E	R	R
R	E	R	$	R	E	F	E	R
R	R	E	R	$	R	E	F	E
E	R	R	E	R	$	R	E	F
F	E	R	R	E	R	$	R	E
E	F	E	R	R	E	R	$	R

All cyclic rotations of the text

$	R	E	F	E	R	R	E	R
E	F	E	R	R	E	R	$	R
E	R	$	R	E	F	E	R	R
E	R	R	E	R	$	R	E	F
F	E	R	R	E	R	$	R	E
R	$	R	E	F	E	R	R	E
R	E	F	E	R	R	E	R	$
R	E	R	$	R	E	F	E	R
R	R	E	R	$	R	E	F	E

Sort all rows alphabetically

The last colum is BWT.

Practice: Efficient Inversion of BWT

1	$	R	E	F	E	R	R	E	R
2	E	F	E	R	R	E	R	$	R
3	E	R	$	R	E	F	E	R	R
4	E	R	R	E	R	$	R	E	F
5	F	E	R	R	E	R	$	R	E
6	R	$	R	E	F	E	R	R	E
7	R	E	F	E	R	R	E	R	$
8	R	E	R	$	R	E	F	E	R
9	R	R	E	R	$	R	E	F	E

Quiz time!
https://flux.qa - YTJMAZ

What are the values in the Rank array?

A. 1, 2, 4, 5
B. 1, 4, 5, 9
C. 1, 2, 5, 6
D. None of the above

Rank				
	$	E	F	R

Practice: Efficient Inversion of BWT

1	$	R	E	F	E	R	R	E	R
2	E	F	E	R	R	E	R	$	R
3	E	R	$	R	E	F	E	R	R
4	E	R	R	E	R	$	R	E	F
5	F	E	R	R	E	R	$	R	E
6	R	$	R	E	F	E	R	R	E
7	R	E	F	E	R	R	E	R	$
8	R	E	R	$	R	E	F	E	R
9	R	R	E	R	$	R	E	F	E

What are the values in the Rank array?

A. 1, 2, 4, 5
B. 1, 4, 5, 9
C. 1, 2, 5, 6
D. None of the above

Rank

1	2	5	6
$	E	F	R

Practice: Efficient Inversion of BWT

1	$	R	E	F	E	R	R	E	R
2	E	F	E	R	R	E	R	$	R
3	E	R	$	R	E	F	E	R	R
4	E	R	R	E	R	$	R	E	F
5	F	E	R	R	E	R	$	R	E
6	R	$	R	E	F	E	R	R	E
7	R	E	F	E	R	R	E	R	$
8	R	E	R	$	R	E	F	E	R
9	R	R	E	R	$	R	E	F	E

Occ

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

What are the values in the Occ array?

Practice: Efficient Inversion of BWT

1	$	R	E	F	E	R	R	E	R
2	E	F	E	R	R	E	R	$	R
3	E	R	$	R	E	F	E	R	R
4	E	R	R	E	R	$	R	E	F
5	F	E	R	R	E	R	$	R	E
6	R	$	R	E	F	E	R	R	E
7	R	E	F	E	R	R	E	R	$
8	R	E	R	$	R	E	F	E	R
9	R	R	E	R	$	R	E	F	E

Occ

1	0
2	1
3	2
4	0
5	0
6	1
7	0
8	3
9	2

What are the values in the Occ array?

Outline

1. Compression
2. Burrows-Wheeler Transform (BWT)
3. Why BWT is effective for compression
4. Decompressing BWT
   A. Naïve Approach
   B. Efficient Approach
5. Substring search using BWT

Substring search using BWT

1	$	M	I	S	S	I	S	S	I	P	P	I_1
2	I_1	$	M	I	S	S	I	S	S	I	P	P_1
3	I_2	P	P	I	$	M	I	S	S	I	S	S_1
4	I_3	S	S	I	P	P	I	$	M	I	S	S_2
5	I_4	S	S	I	S	S	I	P	P	I	$	M_1
6	M_1	I	S	S	I	S	S	I	P	P	I	$
7	P_1	I	$	M	I	S	S	I	S	S	I	P_2
8	P_2	P	I	$	M	I	S	S	I	S	S	I_2
9	S_1	I	P	P	I	$	M	I	S	S	I	S_3
10	S_2	I	S	S	I	P	P	I	$	M	I	S_4
11	S_3	S	I	P	P	I	$	M	I	S	S	I_3
12	S_4	S	I	S	S	I	P	P	I	$	M	I_4

range

Suppose we want to search **SIS** in the string.

- Initially the range contains all rows of BWT
- Start from the last character **S** of SIS.
- Find first **S** in the range and the last **S** in the range in the Last column
- Find the corresponding **S**s in the first column and update the range
- Now, find the first **I** in the range and the last **I** in the range in the Last column
- Find the corresponding **I**s in the first column and update the range.
- Now, find the first **S** in the range and the last S in the range
- Find the corresponding **S**s in first column and update the range

At any stage, if the character is not found in the range then the substring is not present and false can be returned.

S I S

Locating the substring

i	SA[i]	suffix
1	12	$
2	11	i$
3	8	ippi$
4	5	issippi$
5	2	ississippi$
6	1	mississippi$
7	10	pi$
8	9	ppi$
9	7	sippi$
10	4	sissippi$
11	6	ssippi$
12	3	ssissippi$

- Where does this substring occur in the original string?
- To answer, we need a suffix array
- We ended on index 10 in the BWT

Locating the substring

i	SA[i]	suffix
1	12	$
2	11	i$
3	8	ippi$
4	5	issippi$
5	2	ississippi$
6	1	mississippi$
7	10	pi$
8	9	ppi$
9	7	sippi$
10	4	sissippi$
11	6	ssippi$
12	3	ssissippi$

- Where does this substring occur in the original string?
- To answer, we need a suffix array
- We ended on index 10 in the BWT
- Since SA[10] = 4, the 4th character of the original string is the start of our SIS substring
- MISSISSIPPI$

Substring search using BWT

1	\$	M I S S I S S I P P	I_1
2	I_1	\$ M I S S I S S I P	P_1
3	I_2	P P I \$ M I S S I S	S_1
4	I_3	S S I P P I \$ M I S	S_2
5	I_4	S S I S S I P P I \$	M_1
6	M_1	I S S I S S I P P I	\$
7	P_1	I \$ M I S S I S S I	P_2
8	P_2	P I \$ M I S S I S S	I_2
9	S_1	I P P I \$ M I S S I	S_3
10	S_2	I S S I P P I \$ M I	S_4
11	S_3	S I P P I \$ M I S S	I_3
12	S_4	S I S S I P P I \$ M	I_4

How to efficiently compute first and last occurrence of a character c in the range.

- **For each character, create a sorted array of their positions in the last column – this can be done in linear time**

To search a character c in range(i,j), use binary search.

- **to search the first S in the range (5,11), binary search for the smallest position equal to or larger than 5 in the array of S**
- **to search the last S in the range (5,11), binary search for the largest position smaller than or equal to 11**

I	**1, 8, 11, 12**
M	**5**
P	**2, 7**
S	**3, 4, 9, 10**

Time Complexity:

O(M log N) where M is length of substring.

Could be improved to O(M) by maintaining an occ array of size alphabet * M (example to follow)

Substring search in linear O(M) time

- In the following slides, we use a larger version of "occ"
- Occ[row,char] is the number of times we have seen character "char" **before** this row
- We have two pointers to keep track of our range, first and last

Substring search in linear O(M) time

- Occ[row,char] is the number of times we have seen character “char” **before** this row
- Update rules for first and last
- Next_char refers to the next character in the pattern that we are searching

- First = rank[next_char] + occ[first, next_char]
- Last = rank[next_char] + occ[last + 1, next_char] - 1

Substring search using BWT

1	$	M	I	S	S	I	S	S	I	P	P	I_1
2	I_1	$	M	I	S	S	I	S	S	I	P	P_1
3	I_2	P	P	I	$	M	I	S	S	I	S	S_1
4	I_3	S	S	I	P	P	I	$	M	I	S	S_2
5	I_4	S	S	I	S	S	I	P	P	I	$	M_1
6	M_1	I	S	S	I	S	S	I	P	P	I	$
7	P_1	I	$	M	I	S	S	I	S	S	I	P_2
8	P_2	P	I	$	M	I	S	S	I	S	S	I_2
9	S_1	I	P	P	I	$	M	I	S	S	I	S_3
10	S_2	I	S	S	I	P	P	I	$	M	I	S_4
11	S_3	S	I	P	P	I	$	M	I	S	S	I_3
12	S_4	S	I	S	S	I	P	P	I	$	M	I_4

S I S

Occ

	$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

Rank

1	2	6	7	9
$	I	M	P	S

Count

1	4	1	2	4
$	I	M	P	S

Substring search using BWT

1	$	M I S S I S S I P P	I_1 ←
2	I_1	$ M I S S I S S I P	P_1
3	I_2	P P I $ M I S S I S	S_1
4	I_3	S S I P P I $ M I S	S_2
5	I_4	S S I S S I P P I $	M_1
6	M_1	I S S I S S I P P I	$
7	P_1	I $ M I S S I S S I	P_2
8	P_2	P I $ M I S S I S S	I_2
9	S_1	I P P I $ M I S S I	S_3
10	S_2	I S S I P P I $ M I	S_4
11	S_3	S I P P I $ M I S S	I_3
12	S_4	S I S S I P P I $ M	I_4 ←

S I S

Occ

	$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

Rank

1	2	6	7	9
$	I	M	P	S

Count

1	4	1	2	4
$	I	M	P	S

Substring search using BWT

1	\$	M I S S I S S I P P	I_1
2	I_1	\$ M I S S I S S I P	P_1
3	I_2	P P I \$ M I S S I S	S_1
4	I_3	S S I P P I \$ M I S	S_2
5	I_4	S S I S S I P P I \$	M_1
6	M_1	I S S I S S I P P I	\$
7	P_1	I \$ M I S S I S S I	P_2
8	P_2	P I \$ M I S S I S S	I_2
9	S_1	I P P I \$ M I S S I	S_3
10	S_2	I S S I P P I \$ M I	S_4
11	S_3	S I P P I \$ M I S S	I_3
12	S_4	S I S S I P P I \$ M	I_4

S I S

First S: rank[S] + occ[1,S] = 9 + 0 = 9

Last S: rank[S] + occ[12+1,S] – 1 = 9 + 4 – 1 = 12

Occ

	\$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

Rank

1	2	6	7	9
\$	I	M	P	S

Count

1	4	1	2	4
\$	I	M	P	S

Substring search using BWT

	F		L
1	\$	M I S S I S S I P P	I_1
2	I_1	\$ M I S S I S S I P	P_1
3	I_2	P P I \$ M I S S I S	S_1
4	I_3	S S I P P I \$ M I S	S_2
5	I_4	S S I S S I P P I \$	M_1
6	M_1	I S S I S S I P P I	\$
7	P_1	I \$ M I S S I S S I	P_2
8	P_2	P I \$ M I S S I S S	I_2
9	S_1	I P P I \$ M I S S I	S_3
10	S_2	I S S I P P I \$ M I	S_4
11	S_3	S I P P I \$ M I S S	I_3
12	S_4	S I S S I P P I \$ M	I_4

S I S

First S: rank[S] + occ[1,S] = 9 + 0 = 9

Last S: rank[S] + occ[12+1,S] – 1 = 9 + 4 – 1 = 12

Occ

	\$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

Rank

1	2	6	7	9
\$	I	M	P	S

Count

1	4	1	2	4
\$	I	M	P	S

Substring search using BWT

1	\$	M I S S I S S I P P	I_1
2	I_1	\$ M I S S I S S I P	P_1
3	I_2	P P I \$ M I S S I S	S_1
4	I_3	S S I P P I \$ M I S	S_2
5	I_4	S S I S S I P P I \$	M_1
6	M_1	I S S I S S I P P I	\$
7	P_1	I \$ M I S S I S S I	P_2
8	P_2	P I \$ M I S S I S S	I_2
9	S_1	I P P I \$ M I S S I	S_3 ←
10	S_2	I S S I P P I \$ M I	S_4
11	S_3	S I P P I \$ M I S S	I_3
12	S_4	S I S S I P P I \$ M	I_4 ←

↓

S I S

Occ

	\$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

1	2	6	7	9
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Substring search using BWT

1	\$	M I S S I S S I P P	I_1
2	I_1	\$ M I S S I S S I P	P_1
3	I_2	P P I \$ M I S S I S	S_1
4	I_3	S S I P P I \$ M I S	S_2
5	I_4	S S I S S I P P I \$	M_1
6	M_1	I S S I S S I P P I	\$
7	P_1	I \$ M I S S I S S I	P_2
8	P_2	P I \$ M I S S I S S	I_2
9	S_1	I P P I \$ M I S S I	S_3 ←
10	S_2	I S S I P P I \$ M I	S_4
11	S_3	S I P P I \$ M I S S	I_3
12	S_4	S I S S I P P I \$ M	I_4 ←

↓

S I S

First I: rank[I] + occ[9,I] = 2 + 2 = 4

Last I: rank[I] + occ[12+1, I] – 1 = 2 + 4 – 1 = 5

Occ

	\$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

1	2	6	7	9
\$	I	M	P	S

Rank

1	4	1	2	4
\$	I	M	P	S

Count

Substring search using BWT

1	$	M I S S I S S I P P	I_1
2	I_1	$ M I S S I S S I P	P_1
3	I_2	P P I $ M I S S I S	S_1
4	I_3	S S I P P I $ M I S	S_2 ←
5	I_4	S S I S S I P P I $	M_1 ←
6	M_1	I S S I S S I P P I	$
7	P_1	I $ M I S S I S S I	P_2
8	P_2	P I $ M I S S I S S	I_2
9	S_1	I P P I $ M I S S I	S_3
10	S_2	I S S I P P I $ M I	S_4
11	S_3	S I P P I $ M I S S	I_3
12	S_4	S I S S I P P I $ M	I_4

↓

S I S

Occ

	$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

1	2	6	7	9
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Substring search using BWT

1	$	M I S S I S S I P P	I_1
2	I_1	$ M I S S I S S I P	P_1
3	I_2	P P I $ M I S S I S	S_1
4	I_3	S S I P P I $ M I S	S_2 ←
5	I_4	S S I S S I P P I $	M_1 ←
6	M_1	I S S I S S I P P I	$
7	P_1	I $ M I S S I S S I	P_2
8	P_2	P I $ M I S S I S S	I_2
9	S_1	I P P I $ M I S S I	S_3
10	S_2	I S S I P P I $ M I	S_4
11	S_3	S I P P I $ M I S S	I_3
12	S_4	S I S S I P P I $ M	I_4

S I S

First S: rank[S] + occ[4, S] = 9 + 1 = 10

Last S: rank[S] + occ[5+1, S] – 1 = 9 + 2 – 1 = 10

Occ

	$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

Rank

1	2	6	7	9
$	I	M	P	S

Count

1	4	1	2	4
$	I	M	P	S

Substring search using BWT

1	$	M	I	S	S	I	S	S	I	P	P	I_1
2	I_1	$	M	I	S	S	I	S	S	I	P	P_1
3	I_2	P	P	I	$	M	I	S	S	I	S	S_1
4	I_3	S	S	I	P	P	I	$	M	I	S	S_2
5	I_4	S	S	I	S	S	I	P	P	I	$	M_1
6	M_1	I	S	S	I	S	S	I	P	P	I	$
7	P_1	I	$	M	I	S	S	I	S	S	I	P_2
8	P_2	P	I	$	M	I	S	S	I	S	S	I_2
9	S_1	I	P	P	I	$	M	I	S	S	I	S_3
10	S_2	I	S	S	I	P	P	I	$	M	I	S_4 ⇐
11	S_3	S	I	P	P	I	$	M	I	S	S	I_3
12	S_4	S	I	S	S	I	P	P	I	$	M	I_4

↓

S I S

Occ

	$	I	M	P	S
1	0	0	0	0	0
2	0	1	0	0	0
3	0	1	0	1	0
4	0	1	0	1	1
5	0	1	0	1	2
6	0	1	1	1	2
7	1	1	1	1	2
8	1	1	1	2	2
9	1	2	1	2	2
10	1	2	1	2	3
11	1	2	1	2	4
12	1	3	1	2	4
13	1	4	1	2	4

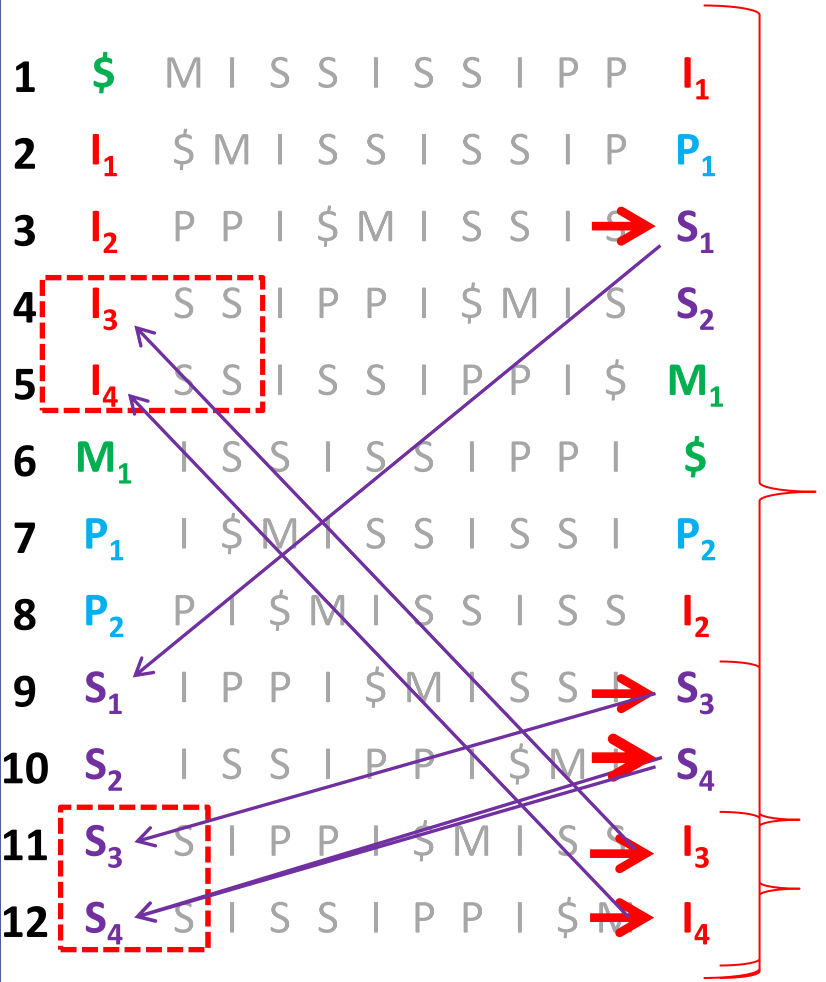
1	2	6	7	9
$	I	M	P	S

Rank

1	4	1	2	4
$	I	M	P	S

Count

Substring search using BWT



Another example:

Suppose we want to search **ISS** in the string.

- Initially the range contains all rows of BWT
- Start from the last character **S** of SIS.
- Find first **S** in the range and the last **S** in the range in the Last column
- Find the corresponding **S**s in the first column and update the range
- Now, find the first **S** in the range and the last S in the range in the Last column
- Find the corresponding **S**s in the first column and update the range.
- Now, find the first **I** in the range and the last **I** in the range
- Find the corresponding **I**s in first column and update the range

I S S

Practice: Substring matching

1	$ R E F E R R E R
2	E F E R R E R $ R
3	E R $ R E F E R R
4	E R R E R $ R E F
5	F E R R E R $ R E
6	R $ R E F E R R E
7	R E F E R R E R $
8	R E R $ R E F E R
9	R R E R $ R E F E

- Search ER
- Search RE
- Search FEF

Summary

Take home message

- Burrows-Wheeler Transform is an elegant algorithm that allows efficient and effective compression and substring matching

Things to do (this list is not exhaustive)

- Read more about Burrows-Wheeler Transform and understand how and why it works
- Implement it in Python

Coming Up Next

- Introduction to Graphs and Path problems on Graphs