

# **FIT2004**

## **Algorithms and Data Structures**

Ian Wern Han Lim  
[lim.wern.han@monash.edu](mailto:lim.wern.han@monash.edu)

Referencing materials by  
Nathan Compane, Aamir Cheema, Arun Konagurthu and Lloyd Allison



# Faculty of Information Technology, Monash University

---

## COMMONWEALTH OF AUSTRALIA

### *Copyright Regulations 1969*

This material has been reproduced and communicated to you by or on behalf of Monash University pursuant to Part VB of the Copyright Act 1968 (the Act). The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act. Do not remove this notice

Ready?

# Agenda

- String retrieval

# Agenda

- String retrieval
- Suffix array

# Agenda

- String retrieval
- Suffix array
  - Very useful for a number of tasks...
  - Very popular question

Let us begin...

# Suffix Array

## Sorted suffixes

- Let say you have a word



# Suffix Array

## Sorted suffixes

- Let say you have a word
  - List out the suffixes
  - Sort the suffixes

# Suffix Array

## Sorted suffixes

String

<b>M</b>	<b>I</b>	<b>S</b>	<b>S</b>	<b>I</b>	<b>S</b>	<b>S</b>	<b>I</b>	<b>P</b>	<b>P</b>	<b>I</b>	<b>\$</b>
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	-----------

# Suffix Array

## Sorted suffixes

String

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

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

# Suffix Array

## Sorted suffixes

String

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

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



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

# Suffix Array

## Sorted suffixes

- Let say you have a word
  - List out the suffixes
  - Sort the suffixes
- What is the complexity of this?
  - $N$  is the number of character

# Suffix Array

## Sorted suffixes

- Let say you have a word
  - List out the suffixes
  - Sort the suffixes
- What is the complexity of this?
  - $N$  is the number of character
  - $O(N^2)$  to generate the suffixes
  - $O(N^2 \log N)$  to sort the suffixes
    - Note  $O(N)$  for string comparison

# Suffix Array

## Sorted suffixes

- Let say you have a word
  - List out the suffixes
  - Sort the suffixes
- What is the complexity of this?
  - $N$  is the number of character
  - $O(N^2)$  to generate the suffixes
  - $O(N^2 \log N)$  to sort the suffixes with **merge sort**
    - Note  $O(N)$  for string comparison

# Suffix Array

## Sorted suffixes

- Let say you have a word
  - List out the suffixes
  - Sort the suffixes
- What is the complexity of this?
  - $N$  is the number of character
  - $O(N^2)$  to generate the suffixes
  - $O(N^2 \log N)$  to sort the suffixes with **merge sort**
    - Note  $O(N)$  for string comparison
    - We can reduce this to  $O(N^2)$  with **radix sort**
      - $N$  passes (columns)



Questions?

# Suffix Array

## Sorted suffixes

- Usage?

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP

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

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern

```
$
I $
IPP I $
ISS IPP I $
ISSISSIPP I $
MISSISSIPP I $
PI $
PPI $
SIPP I $
SIS SIPP I $
SSIPPI $
SSISSIPP I $
```

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$

```
$
I $
IPP I $
ISSIPP I $
ISSISSIPP I $
MISSISSIPP I $
PI $
PPI $
SIPP I $
SIS SIPP I $
SSIPPI $
SSISSIPP I $
```

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next



# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next

```
$
I $ 0
I P P I $
I S S I P P I $
I S S I S S I P P I $
M I S S I S S I P P I $
P I $
P P I $
S I P P I $
S I S S I P P I $
S S I P P I $
S S I S S I P P I $
```

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next

\$

I \$

I P P I \$

I S S I P P I \$

I S S I S S I P P I \$

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

P I \$

P P I \$

S I P P I \$

S I S S I P P I \$

S S I P P I \$

S S I S S I P P I \$

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next

```
$
I $
IPP I $
ISS IPP I $
ISSISSIPP I $
MISSISSIPP I $
PI $
PP I $
SIPP I $
SISSIPPI $
SSIPPI $
SISSIPPI $
```

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next

\$  
I \$  
I P P I \$  
I S S I P P I \$  
I S S I S S I P P I \$ 4  
M I S S I S S I P P I \$  
P I \$  
P P I \$  
S I P P I \$  
S I S S I P P I \$  
S S I P P I \$  
S S I S S I P P I \$

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next

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

3

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next
    - This is the longest with length 4
      - ISSI
    - Complexity?

```
$
I $
I P P I $
ISSIPPI $
ISSISSIPPI $
MISSISSIPPI $
PI $
PPI $
SIPPI $
SIS S I P P I $
SSI P P I $
SSI S S I P P I $
```

4

# Suffix Array

## Sorted suffixes

- Usage?
  - Search for substring
    - With binary search!
    - Search for IPP
    - Complexity?
      - M is length of pattern
      - $O(M \log N)$
  - Finding longest repeated substring
    - Since it is in order, we can just compare one suffix with the next
    - This is the longest with length 4
      - ISSI
    - Complexity?
      - $O(N^2)$
      - Due to  $O(N)$  for char comparison

```
$
I $
I P P I $
ISSIPPI $
ISSISSIPPI $
MISSISSIPPI $
PI $
PPI $
SIPPI $
SIS S I P P I $
SSI P P I $
SSI S S I P P I $
```

4

Questions?



# Suffix Array

## Sorted suffixes

- Space complexity?

# Suffix Array

## Sorted suffixes

- Space complexity?
  - $O(N^2)$

# Suffix Array

## Sorted suffixes

- Space complexity?
  - $O(N^2)$
  - But can we do better?

# Suffix Array

## Sorted suffixes

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

# Suffix Array

## Sorted suffixes

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

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



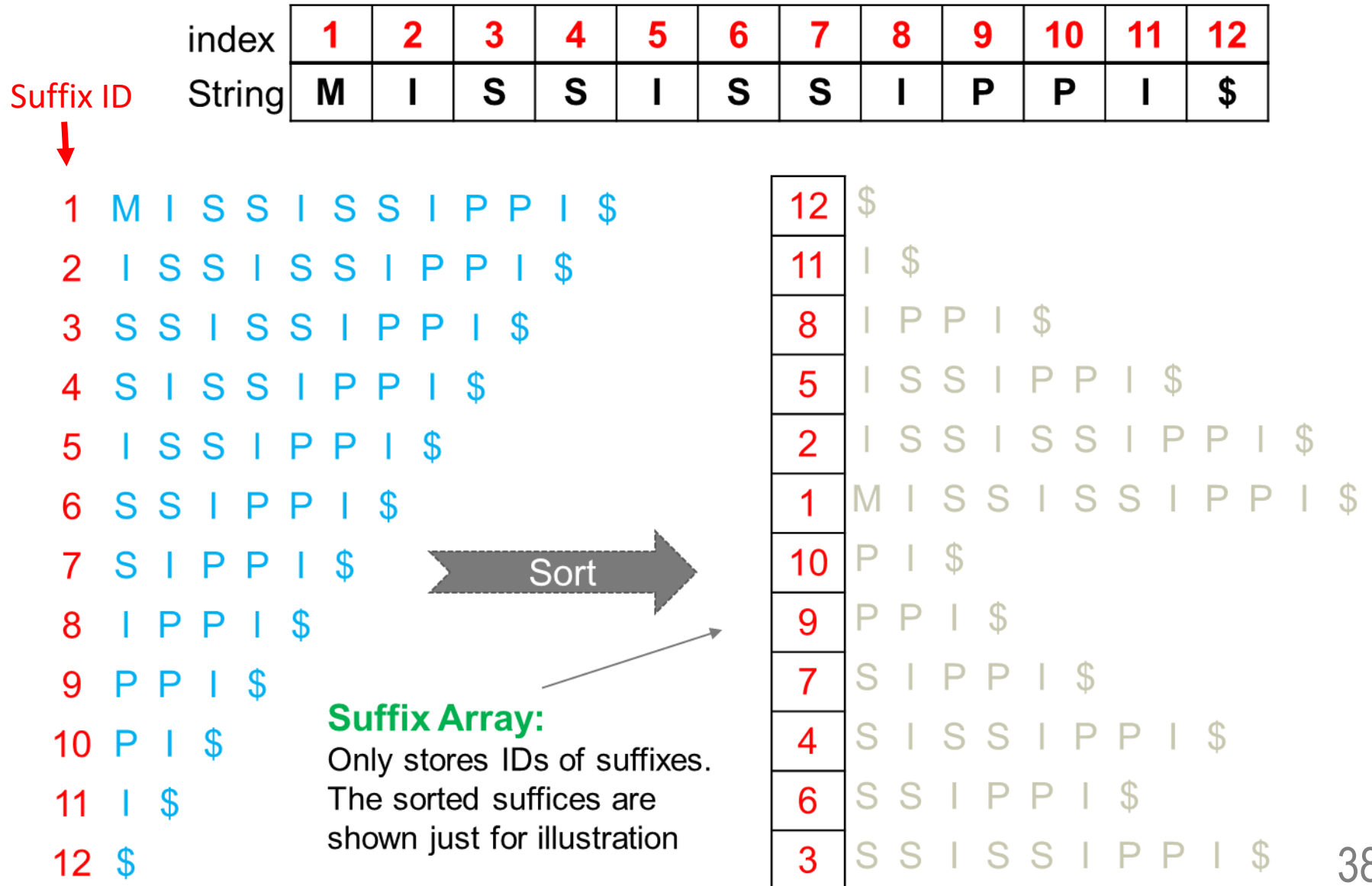
### Suffix Array:

Only stores IDs of suffixes.  
The sorted suffixes are shown just for illustration

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

# Suffix Array

## Sorted suffixes



# Suffix Array

## Sorted suffixes

- Space complexity?
  - $O(N^2)$
  - But can we do better?
  - $O(N)$  if we use **suffix ID** and store the suffix ID (in order) only

# Suffix Array

## Sorted suffixes

- Space complexity?
  - $O(N^2)$
  - But can we do better?
  - $O(N)$  if we use **suffix ID** and store the suffix ID (in order) only
    - With suffix ID, we know where the suffix start
    - We can get the suffixes by doing `suffix = string[id:]`



Questions?

# Suffix Array

## Sorted suffixes

- Time complexity is  $O(N^2)$  with radix sort
- Space complexity is  $O(N)$  with suffix ID

# Suffix Array

## Sorted suffixes

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
- Space complexity is  $O(N)$  with suffix ID

# Suffix Array

## Sorted suffixes

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
- Space complexity is  $O(N)$  with suffix ID

# Suffix Array

With prefix doubling

- Let us see the example now

# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters

## Rank ID

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

# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters
- Sort suffixes on first 2 characters

## Rank ID

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

# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters
- Sort suffixes on first 2 characters
- Sort suffixes on first 4 characters

## Rank ID

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



# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters
- Sort suffixes on first 2 characters
- Sort suffixes on first 4 characters
- ...

## Rank ID

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

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

# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters
- Sort suffixes on first 2 characters
- Sort suffixes on first 4 characters
- ...
- Sort suffixes on all characters

## Rank ID

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

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

# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters
- Sort suffixes on first 2 characters
- Sort suffixes on first 4 characters
- ...
- Sort suffixes on all N characters

## Time complexity:

- Cost of first sort (1 character)
  - $1.N \log N$
- Cost of second sort (2 characters)
  - $2.N \log N$
- Cost of i-th sort ( $2^{i-1}$  characters)
  - $2^{i-1} N \log N$
- **Total cost:**
- $N \log N + 2N \log N + 4N \log N + \dots + N.N \log N$
- $(1 + 2 + 4 + \dots + N/2 + N) * N \log N$ 
  - $(N + N/2 + N/4 + \dots + 1) \rightarrow O(N)$
- Total cost is still  $O(N^2 \log N)$

## Rank ID

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

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

# Suffix Array

## With prefix doubling

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
- Space complexity is  $O(N)$  with suffix ID

# Suffix Array

## With prefix doubling

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
- Space complexity is  $O(N)$  with suffix ID

# Suffix Array

## With prefix doubling

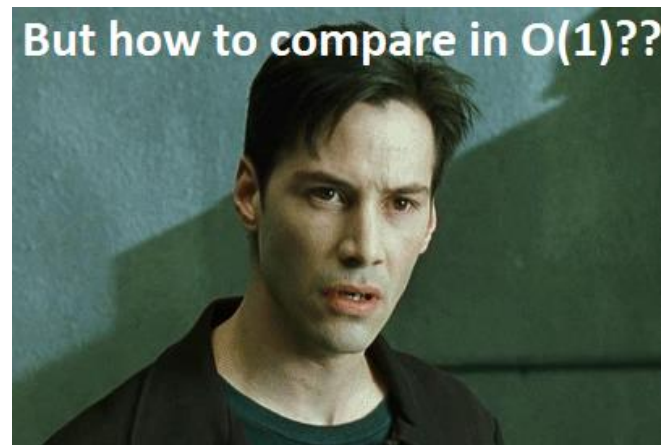
- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
- Space complexity is  $O(N)$  with suffix ID



# Suffix Array

## With prefix doubling

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
- Space complexity is  $O(N)$  with suffix ID



Questions?



# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
    - We use **the rank table to get  $O(1)$  comparison!**
- Space complexity is  $O(N)$  with suffix ID

# Constructing Suffix Array: Prefix Doubling

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

## Basic Idea:

- Generate suffixes
- Sort suffixes on their 1st characters
- Sort suffixes on first 2 characters
- Sort suffixes on first 4 characters
- ...
- Sort suffixes on all N characters
- **Suppose we could compare in  $O(1)$**
- $\log N$  sorts
- $O(N \log N)$  for each
- $O(N \log N \log N) = O(N \log^2 N)$

## Rank ID

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

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

# O(1) Comparison

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

## Comparing suffixes in O(1):

- Suppose already sorted on first  $k$  characters (2 in this example)
- Now sorting on  $2k$  characters (4 in this example)

## Observation 1:

- If current ranks are different, suffix with smaller rank is smaller (because its first  $k$  characters are smaller)
- E.g., PPI\$ < SSIP
- Note comparison cost is  $O(1)$

## Rank ID

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

# O(1) Comparison

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

## Observation 2:

If current ranks are the same

- First k characters must be the same
- The tie is to be broken on the next k characters, e.g.,

## Rank ID

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

# O(1) Comparison

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

## Observation 2:

If current ranks are the same

- First k characters must be the same
- The tie is to be broken on the next k characters, e.g.,
  - We need to compare “SSIPPI\$” and “PPI\$” on the first 2 characters

## Rank ID

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

# O(1) Comparison

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

## Observation 2:

If current ranks are the same

- First k characters must be the same
- The tie is to be broken on the next k characters, e.g.,
  - We need to compare “SSIPPI\$” and “PPI\$” on the first 2 characters
- SSIPPI\$ and PPI\$ are suffixes and are already ranked on first 2 characters
  - E.g., PPI\$ < SSIPPI\$ because its rank is smaller
  - Therefore, suffix #7 < suffix #4

## Rank ID

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

# Practice

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

- BUT WAIT!
- How did we do that quickly?  
Surely looking up the “second half” suffixes is  $O(N)$ ?

Rank ID

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

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

# Practice

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

- BUT WAIT!
- How did we do that quickly?  
Surely looking up the “second half” suffixes is  $O(N)$ ?
- **USE SUFFIX IDs!**

## Rank ID

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

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



# Practice

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

Suppose we are comparing suffix  
with ID 2 and 5:

- We need to compare SSIPPI\$ and PPI\$

Rank ID

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

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

# Practice

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

Suppose we are comparing suffix with ID 2 and 5:

- We need to compare SSIPPI\$ and PPI\$
- How do we find their ranks quickly?

Rank ID

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

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

# Practice

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

Suppose we are comparing suffix with ID 2 and 5:

- We need to compare SSIPPI\$ and PPI\$
- How do we find their ranks quickly?
- We want the ranks of suffixes:  $2+k$  and  $5+k$
- I.e. suffixes 6 and 9
- This means we can calculate the IDs of the suffixes we want in  $O(1)$
- Now we need to get from IDs to ranks in  $O(1)$

Rank ID

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

# Practice

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

Suppose we are comparing suffix with ID 2 and 5:

- We need to compare SSIPPI\$ and PPI\$
- How do we find their ranks quickly?
- We want the ranks of suffixes:  $2+k$  and  $5+k$
- I.e. suffixes 6 and 9
- To have  $O(1)$  access to their ranks, we need an array indexed by ID which contains the ranks!
- In other words, the way the ranks are arranged on this slide is useless

Rank ID

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

# O(1) Comparison

Index/ID	1	2	3	4	5	6	7	8	9	10	11	12
Rank	6	4	12	10	4	11	9	3	8	7	2	1
String	M	I	S	S	I	S	S	I	P	P	I	\$

**Note:** The greyed out oldRank array has been left for reference, but does not exist in implementation

- If we want the rank of ID i, look at Rank[i]
- Going back to our example...

oldRank ID

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

# O(1) Comparison

Index/ID	1	2	3	4	5	6	7	8	9	10	11	12
Rank	6	4	12	10	4	11	9	3	8	7	2	1
String	M	I	S	S	I	S	S	I	P	P	I	\$

**Note:** The greyed out oldRank array has been left for reference, but does not exist in implementation

- If we want the rank of ID i, look at Rank[i]
- Going back to our example...
- We wanted to find the second parts of suffixes 2 and 5

oldRank ID

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

# O(1) Comparison

Index/ID	1	2	3	4	5	6	7	8	9	10	11	12
Rank	6	4	12	10	4	11	9	3	8	7	2	1
String	M	I	S	S	I	S	S	I	P	P	I	\$

**Note: The greyed out oldRank array has been left for reference, but does not exist in implementation**

- If we want the rank of ID i, look at Rank[i]
- Going back to our example...
- We wanted to find the second parts of suffixes 2 and 5
- I.e. ID 6 and 9
- Rank[9] < rank[6]
- So ID 5 should come before ID 2 in the suffix array

oldRank ID

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

# Lets do an example (by hand in lecture)

ID	String	Rank		Rank	SA	
1	M	-		-	1	M I S S I S S I P P I \$
2	I	-		-	2	I S S I S S I P P I \$
3	S	-		-	3	S S I S S I P P I \$
4	S	-		-	4	S I S S I P P I \$
5	I	-		-	5	I S S I P P I \$
6	S	-		-	6	S S I P P I \$
7	S	-		-	7	S I P P I \$
8	I	-		-	8	I P P I \$
9	P	-		-	9	P P I \$
10	P	-		-	10	P I \$
11	I	-		-	11	I \$
12	\$	-		-	12	\$

REMEMBER:  
This rank array does not exist

It contains the same values as the other rank array, its just to make the algorithm easier to follow



# Lets do an example (by hand in lecture)

ID	String	Rank		Rank	SA	
1	M	-	Rank the first characters of each suffix	-	1	M I S S I S S I P P I \$
2	I	-		-	2	I S S I S S I P P I \$
3	S	-		-	3	S S I S S I P P I \$
4	S	-		-	4	S I S S I P P I \$
5	I	-		-	5	I S S I P P I \$
6	S	-		-	6	S S I P P I \$
7	S	-		-	7	S I P P I \$
8	I	-		-	8	I P P I \$
9	P	-		-	9	P P I \$
10	P	-		-	10	P I \$
11	I	-		-	11	I \$
12	\$	-		-	12	\$

# Lets do an example (by hand in lecture)

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

Rank the first characters of each suffix

In practice we would do this using `ord()`, but since ranks are only comparative, the actual values don't matter, just their order. So we use numbers starting at 1

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

# Lets do an example (by hand in lecture)

ID	String	Rank		Rank	SA	
1	M	3	Sort SA by ranks	3	1	M I S S I S S I P P I \$
2	I	2		2	2	I S S I S S I P P I \$
3	S	5		5	3	S S I S S I P P I \$
4	S	5		5	4	S I S S I P P I \$
5	I	2		2	5	I S S I P P I \$
6	S	5		5	6	S S I P P I \$
7	S	5		5	7	S I P P I \$
8	I	2		2	8	I P P I \$
9	P	4		4	9	P P I \$
10	P	4		4	10	P I \$
11	I	2		2	11	I \$
12	\$	1		1	12	\$

# Lets do an example (by hand in lecture)

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

Sort SA by ranks

Note that this does not change the rank array, since IDs have kept the same ranks

We just rearranged the SA

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

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

# Lets do an example (by hand in lecture)

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

Now sort on first 2 characters

For each comparison, we use the trick outlined in the previous section

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

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

# Lets do an example (by hand in lecture)

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

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

Now we update the ranks

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

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	1	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

Make an array, "Temp" to hold the new ranks

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	1	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

For each pair of adjacent suffixes, compare them



# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	1	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

If they have different ranks already, then the second suffix is certainly larger

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	2	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

Set Temp[11] to Rank[12]+1

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	If they have the same rank											
1	M	3	1	1	12	\$											
2	I	2	1	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	1	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	1	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	1	4	9	P	P	I	\$								
9	P	4	1	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	1	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	2	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

We need to use the  $O(1)$  trick

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	2	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

Sometimes, one or both suffixes will not have 2k chars (like I\$)

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	1	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	2	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

In such a case, the shorter one comes earlier!

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	1	1	12	\$
2	I	2	1	2	11	I \$
3	S	5	1	2	8	I P P I \$
4	S	5	1	2	2	I S S I S S I P P I \$
5	I	2	1	2	5	I S S I P P I \$
6	S	5	1	3	1	M I S S I S S I P P I \$
7	S	5	1	4	10	P I \$
8	I	2	3	4	9	P P I \$
9	P	4	1	5	4	S I S S I P P I \$
10	P	4	1	5	7	S I P P I \$
11	I	2	2	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

Set Rank[8] = Rank[11] + 1

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	1	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	1	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	1	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	1	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	1	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					



# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	1	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	1	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	1	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	1	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	1	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	1	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	1	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	1	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	1	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	6	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	1	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	7	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	6	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	8	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	1	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	7	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	6	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	1	2	8	I	P	P	I	\$							
4	S	5	8	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	8	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	7	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	6	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	9	2	8	I	P	P	I	\$							
4	S	5	8	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	1	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	8	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	7	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	6	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	Continue in this way											
1	M	3	5	1	12	\$											
2	I	2	4	2	11	I	\$										
3	S	5	9	2	8	I	P	P	I	\$							
4	S	5	8	2	2	I	S	S	I	S	S	I	P	P	I	\$	
5	I	2	4	2	5	I	S	S	I	P	P	I	\$				
6	S	5	9	3	1	M	I	S	S	I	S	S	I	P	P	I	\$
7	S	5	8	4	10	P	I	\$									
8	I	2	3	4	9	P	P	I	\$								
9	P	4	7	5	4	S	I	S	S	I	P	P	I	\$			
10	P	4	6	5	7	S	I	P	P	I	\$						
11	I	2	2	5	3	S	S	I	S	S	I	P	P	I	\$		
12	\$	1	1	5	6	S	S	I	P	P	I	\$					



# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	3	5	1	12	\$
2	I	2	4	2	11	I \$
3	S	5	9	2	8	I P P I \$
4	S	5	8	2	2	I S S I S S I P P I \$
5	I	2	4	2	5	I S S I P P I \$
6	S	5	9	3	1	M I S S I S S I P P I \$
7	S	5	8	4	10	P I \$
8	I	2	3	4	9	P P I \$
9	P	4	7	5	4	S I S S I P P I \$
10	P	4	6	5	7	S I P P I \$
11	I	2	2	5	3	S S I S S I P P I \$
12	\$	1	1	5	6	S S I P P I \$

This is our new Rank array, so overwrite the old one

# Lets do an example (by hand in lecture)

ID	String	Rank	Temp	Rank	SA	
1	M	5	5	1	12	\$
2	I	4	4	2	11	I \$
3	S	9	9	3	8	I P P I \$
4	S	8	8	4	2	I S S I S S I P P I \$
5	I	4	4	4	5	I S S I P P I \$
6	S	9	9	5	1	M I S S I S S I P P I \$
7	S	8	8	6	10	P I \$
8	I	3	3	7	9	P P I \$
9	P	7	7	8	4	S I S S I P P I \$
10	P	6	6	8	7	S I P P I \$
11	I	2	2	9	3	S S I S S I P P I \$
12	\$	1	1	9	6	S S I P P I \$

This is our new Rank array, so overwrite the old one

# Lets do an example (by hand in lecture)

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

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

Now we have the suffixes sorted by the first 2. Go for 4!

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

# Lets do an example (by hand in lecture)

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

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

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

## Lets do an example (by hand in lecture)

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

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

[illegible]

# Lets do an example (by hand in lecture)

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

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

```

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

Questions?

# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Let say we have a suffix “SSISSIPPI\$”



# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Let say we have a suffix “SSISSIPPI\$”
  - It has the suffix ID of 3
  - What is the suffix ID of “PPI\$”?

# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Let say we have a suffix “SSISSI**PPI\$**”
  - It has the suffix ID of 3
  - What is the suffix ID of “**PPI\$**”?

# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Let say we have a suffix “SSISSI**PPI\$**”
  - It has the suffix ID of 3
  - What is the suffix ID of “**PPI\$**”?
    - $3 + 6 = 9$
    - #ez

# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Let say we have a suffix “SSISSI**PPI**\$”
  - It has the suffix ID of 3
  - What is the suffix ID of “**PPI**\$”?
    - $3 + 6 = 9$ 
      - 3 is the original ID
      - 6 is the number of characters from start
    - #ez

Questions?

## With prefix doubling with $O(1)$ comparison

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
    - We use **the rank table to get  $O(1)$  comparison!**
    - So we can use this to sort very quickly within the same rank
- Space complexity is  $O(N)$  with suffix ID

# O(1) Comparison

Index/ID	1	2	3	4	5	6	7	8	9	10	11	12
Rank	6	4	11	9	4	11	9	3	8	7	2	1
String	M	I	S	S	I	S	S	I	P	P	I	\$

Note: We don't need to store the suffixes (shown grey) – we only need Suffix IDs.

## Comparing Suffix with IDs 4 and 7:

- Their ranks are equal.
  - $\text{Rank}[4] = \text{Rank}[7]$ 
    - E.g., they are the same on **first 2** characters
- We need to compare them on the **next 2** characters
- Compare ranks of suffixes  $4+2=6$  and  $7+2=9$ 
  - $\text{Rank}[6] > \text{Rank}[9]$ 
    - So, Suffix #7 is smaller than #4

**Note:** Comparison takes  $O(1)$  and we do not need to store all suffixes – space used is  $O(N)$

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

# O(1) Comparison

Index/ID	1	2	3	4	5	6	7	8	9	10	11	12
Rank	6	4	12	10	4	11	9	3	8	7	2	1
String	M	I	S	S	I	S	S	I	P	P	I	\$

Note: We don't need to store the suffixes (shown grey) – we only need Suffix IDs.

Suppose array has been sorted on first 4 characters.

**Comparing Suffix with IDs 2 and 5:**

- Their ranks are equal.
  - **Rank[2] = Rank[5]**
  - E.g., they are the same on **first 4** characters
- We need to compare them on the **next 4** characters
  - Should we instead compare them on all remaining characters???
  - No, array is sorted on first 4 only
- Compare ranks of suffixes 2+4=6 and 5+4=9
  - **Rank[6] > Rank[9]**
  - So, Suffix #5 is smaller than #2

**Note:** Each comparison takes O(1) and we do not need to store all suffixes – space used is O(N)

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



# Construction Cost of Suffix Array

Index/ID	1	2	3	4	5	6	7	8	9	10	11	12
Rank	6	5	12	10	4	11	9	3	8	7	2	1
String	M	I	S	S	I	S	S	I	P	P	I	\$

## Time Complexity (prefix doubling):

- We need to sort  $O(\log N)$  times
  - Sort on 1 characters
  - Sort on 2 characters
  - ...
  - Sort on  $N/2$  characters
  - Sort on  $N$  characters
- Each sorting requires  $O(N \log N)$  comparisons
- Each comparison takes  $O(1)$
- Total cost:  $O(N \log^2 N)$

## Space Complexity:

- $O(N)$

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

# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
    - K character is sorted
    - So when we calculate the next K character (total of  $2K$ )
      - We **reuse the sorted** first K
      - This is **possible since they are suffixes of the same string!**
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
    - We use **the rank table to get  $O(1)$  comparison!**
    - So we can use this to sort very quickly within the same rank
- Space complexity is  $O(N)$  with suffix ID

# Suffix Array

With prefix doubling with  $O(1)$  comparison

- Time complexity is  $O(N^2)$  with radix sort
  - Can we do better?
  - Yes with prefix doubling!
    - K character is sorted
    - So when we calculate the next K character (total of  $2K$ )
      - We **reuse the sorted** first K
      - This is **possible since they are suffixes of the same string!**
  - But complexity still the same  $O(N^2 \log N)$ , even slower than radix sort
    - Due to the  $O(N)$  comparison
    - We use **the rank table to get  $O(1)$  comparison!**
    - So we can use this to sort very quickly within the same rank
    - So complexity now is  $O(N \log^2 N)$
- Space complexity is  $O(N)$  with suffix ID

Questions?

# Summary

## String search

- We can build
  - Suffix trie
  - Suffix tree
  - Suffix array

- We can build
  - Suffix trie
  - Suffix tree
  - Suffix array
  
- For anything we can treat as a string
  - Including long text (as a single string!)

- We can build
  - Suffix trie
  - Suffix tree
  - Suffix array
- For anything we can treat as a string
  - Including long text (as a single string!)
- Next week, we learn Burrows-Wheeler Transform (BWT)
  - Which is even better!

Questions?



Thank You