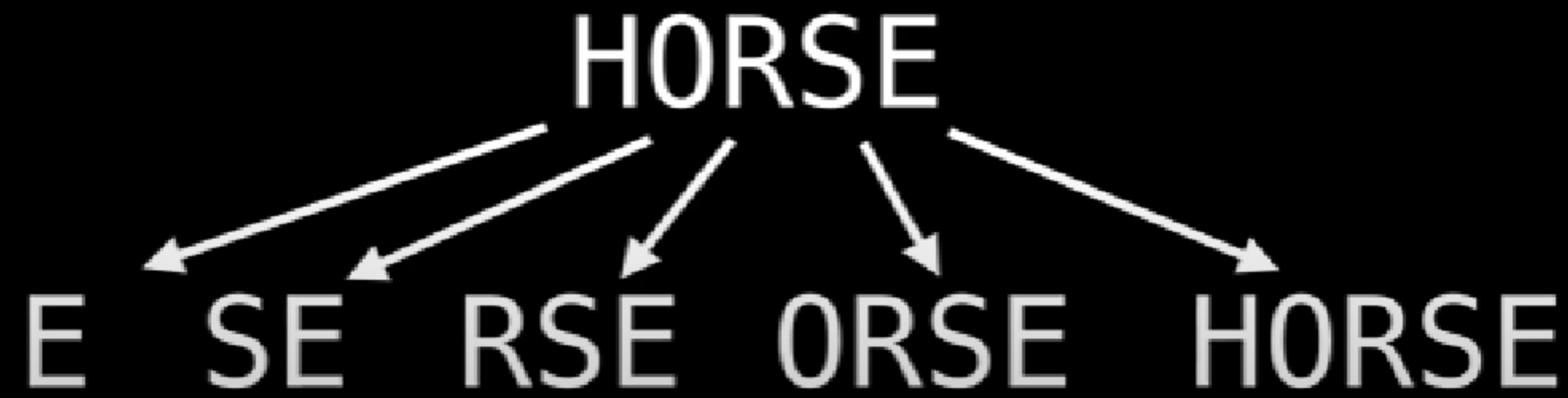
Suffix Array

What is a suffix?

A **suffix** is a substring at the end of a string of characters. For our purposes suffixes are non empty.



What is a SA?

A **suffix array** is an array which contains all the **sorted** suffixes of a string.

For example, the SA of “camel” is:

0	camel
1	amel
2	mel
3	el
4	l

1	amel
0	camel
3	el
4	l
2	mel

What is a SA?

The actual ‘**suffix array**’ is the array of sorted indices.

This provides a compressed representation of the sorted suffixes without actually needing to store the suffixes.

Longest Common Prefix (LCP) Array

What is a LCP array?

This is best shown with an example. Let's find the LCP array of the string '**ABABBAB**'
Begin by constructing the suffix array:

Sorted Index	LCP value	Suffix
5	0	AB
0		ABABBAB
2		ABBAB
6		B
4		BAB
1		BABBAB
3		BBAB

What is a LCP array?

This is best shown with an example. Let's find the LCP array of the string '**ABABBAB**'
Begin by constructing the suffix array:

Sorted Index	LCP value	Suffix
5	0	AB
0	2	ABABBAB
2	2	ABBAB
6	0	B
4	1	BAB
1	3	BABBAB
3	1	BBAB

Summary

The LCP array is an array in which every index tracks how **many characters two sorted adjacent suffixes have in common**.

By convention, LCP[0] is undefined, but for most purposes it's fine to set it to zero.

Finding a substring in a string

Q. Given a string **A** of length **N**, and multiple query string of the form B_i, such that $\sum|B_i| \leq M$, then for each query answer "Yes" or "No", if B_i is a **substring of A** or not?

Aim: $O(N \log^2(N) + M\log(N))$

String = ABABBAABAA

LCP	Suffix
0	A
1	AA
2	AABAA
1	ABAA
→ 3	ABABBAABAA
2	ABBAABAA
0	BAA
↘ 3	BAABAA
2	BABBAABAA
1	BBAABAA

Longest Repeated Substring (LRS)

Find the LRS of the string: ‘ABRACADABRA’

The substring ‘ABRA’ is the longest substring that repeats at least twice so it is the longest repeated substring

ABRACADABRA

Longest Repeated Substring (LRS)

Find the LRS of the string: ABRACADABRA

Find the maximum LCP value

LCP	Suffix
0	A
1	ABRA
4	ABRACADABRA
1	ACADABRA
1	ADABRA
0	BRA
3	BRACADABRA
0	CADABRA
0	DABRA
0	RA
2	RACADABRA

Example #2 with multiple LRSs

What about the LRS of “ABABBAABAA”?

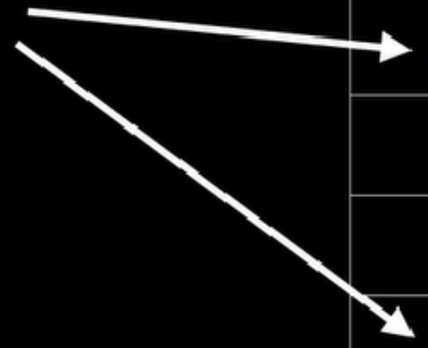
Turns our that this string has
multiple longest repeated substrings
since there can be ties!

ABABBAABAA **ABABBAABAA**

Longest Repeated Substring (LRS)

Find the LRS of the string: ABABBAABAA

Find the maximum LCP value(s)



LCP	Suffix
0	A
1	AA
2	AABAA
1	ABAA
3	ABABBAABAA
2	ABBAABAA
0	BAA
3	BAABAA
2	BABBAABAA
1	BBAABAA

Finding longest substring that occurs ≥ K times?

Q. Given a string **A** of length **N** and an integer **K**. Find the length of the longest string, that occurs in **A** at-least **K** times. (These occurrences are allowed to overlap with one another)

Aim: $O(N\log^2 N)$ or $O(N\log N)$

LCP	Suffix
0	A
1	AA
2	AABAA
1	ABAA
→ 3	ABABBAABAA
2	ABBAABAA
0	BAA
↘ 3	BAABAA
2	BABBAABAA
1	BBAABAA

Finding Unique Substrings

The problem of finding/counting all the unique substrings of a string is a commonplace problem in computer science.

The naive algorithm generates all substrings and places them in a set resulting in a $O(n^2)$ algorithm.

A better approach is to use the **LCP array**. This provides not only a quick but also a space efficient solution.

Finding Unique Substrings

Suppose we wish to find all the unique substrings of: 'AZAZA'

All n(n+1)/2 substrings:

A, AZ, AZA, AZAZ,
AZAZA, Z, ZA, ZAZ,
ZAZA, A, AZ, AZA,
Z, ZA, A

Number of unique substrings: 9

Finding Unique Substrings

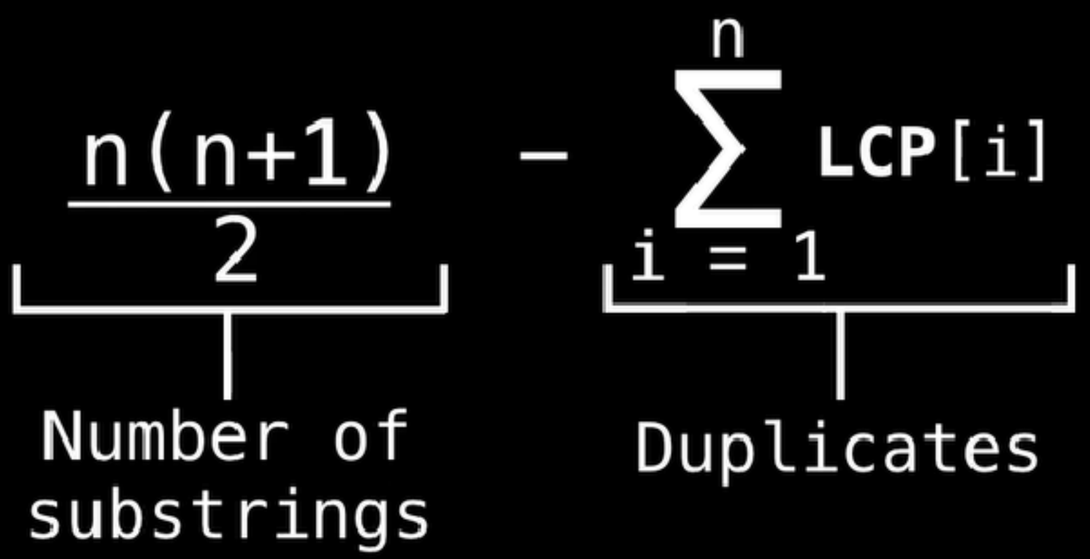
Text: 'AZAZA'

A, AZ, AZA, AZAZ,
AZAZA, Z, ZA, ZAZ,
ZAZA, A, AZ, AZA,
Z, ZA, A

LCP	Sorted Suffixes
0	A
1	AZA
3	AZAZA
0	ZA
2	ZAZA

Finding Unique Substrings

The number of unique substrings in a string is given by:



Suppose we have n strings, how do we find the **longest common substring** that appears in at least $2 \leq k \leq n$ of the strings?

Consider n = 3, k = 2 with:

S_1 = ‘a**bca**’

S_2 = ‘**bca**d’

S_3 = ‘daca’

T = abca#bcad$daca%

Suppose k = 3 what is the LCS?

We can achieve a LCP value of two using these three strings

LCP	Suffixes
0	a#bcad$daca%
1	a%
1	abca#bcad$daca%
1	aca%
1	ad$daca%
0	bca#bcad$daca%
3	bcad$daca%
0	ca#bcad$daca%
2	ca%
2	cad$daca%
0	d$daca%
1	daca%