Alex Josué Flórez Farfán
Magister en Ciencias de la Computación

PROGRAMACIÓN COMPETITIVA

Ciencia de la Computación – UNSA
Segundo semestre 2021

TERCERA UNIDAD

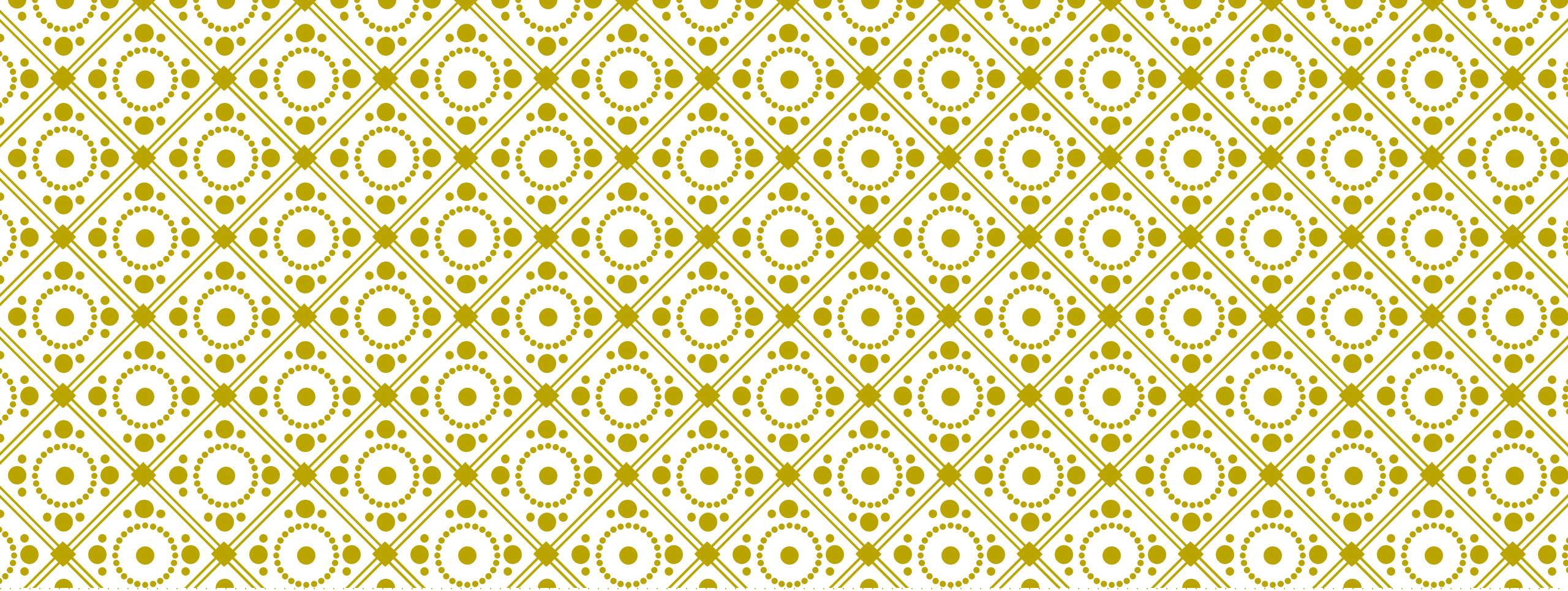
Nota Evaluación Continua

Repositorio

- Directorios para cada aula
- Nombres de archivos

Participaciones en clase

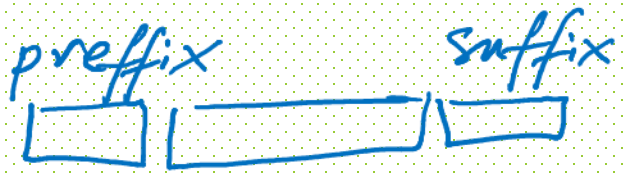
Examen



SUFFIX ARRAY

SUFFIX

A suffix is a substring at the end of a string of characters.



HORSE

01234

0 HORSE
1 ORSE
2 RSE
3 SE
4 E

SUFFIX ARRAY

A Suffix Array is an array which contains all the sorted suffixes of a string

```
camel
0  camel
1   amel
2    mel
3     el
4      l
```

input string

output vector<int>

SA

1	amel
0	camel
3	el
4	l
2	mel

n

$n \log n$

$n^2 \log n$

SUFFIX ARRAY

trie

suffix tree

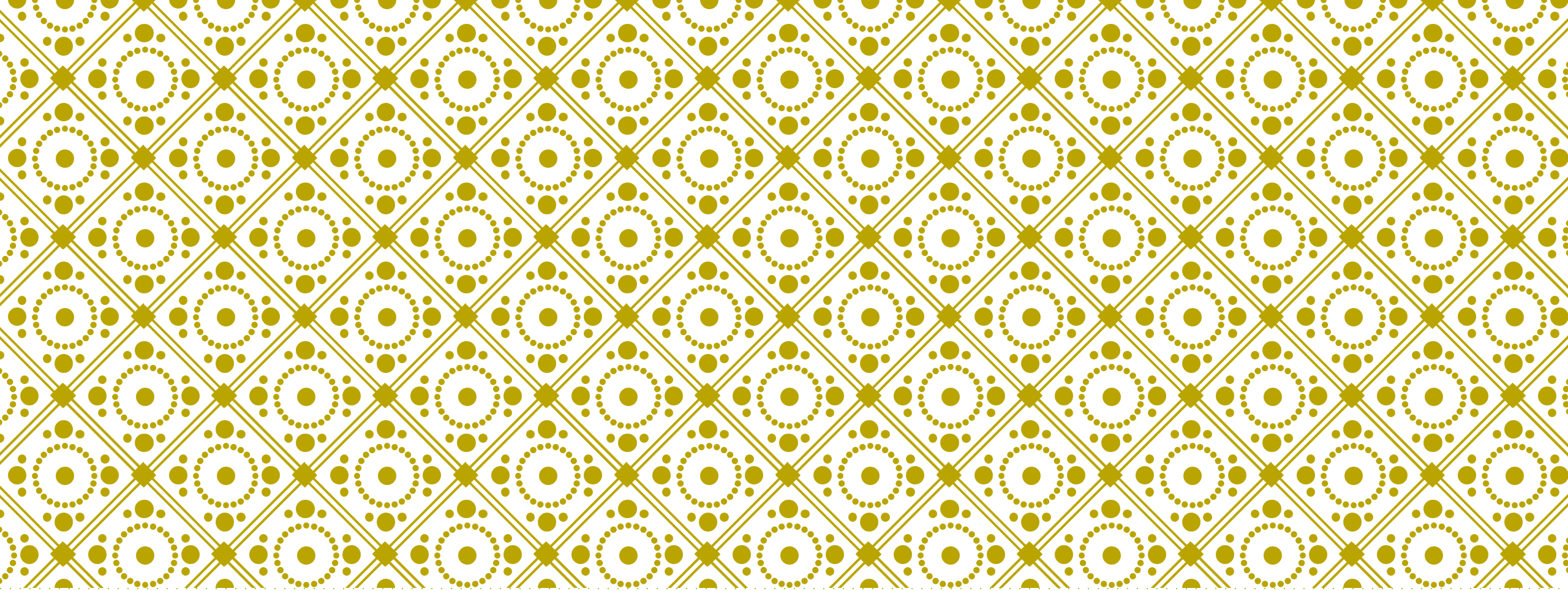
The actual “Suffix Array” is the array of sorted indices.

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

```
1    amel
0   camel
3      el
4       l
2     mel
```

camel

1 0 3 4 2

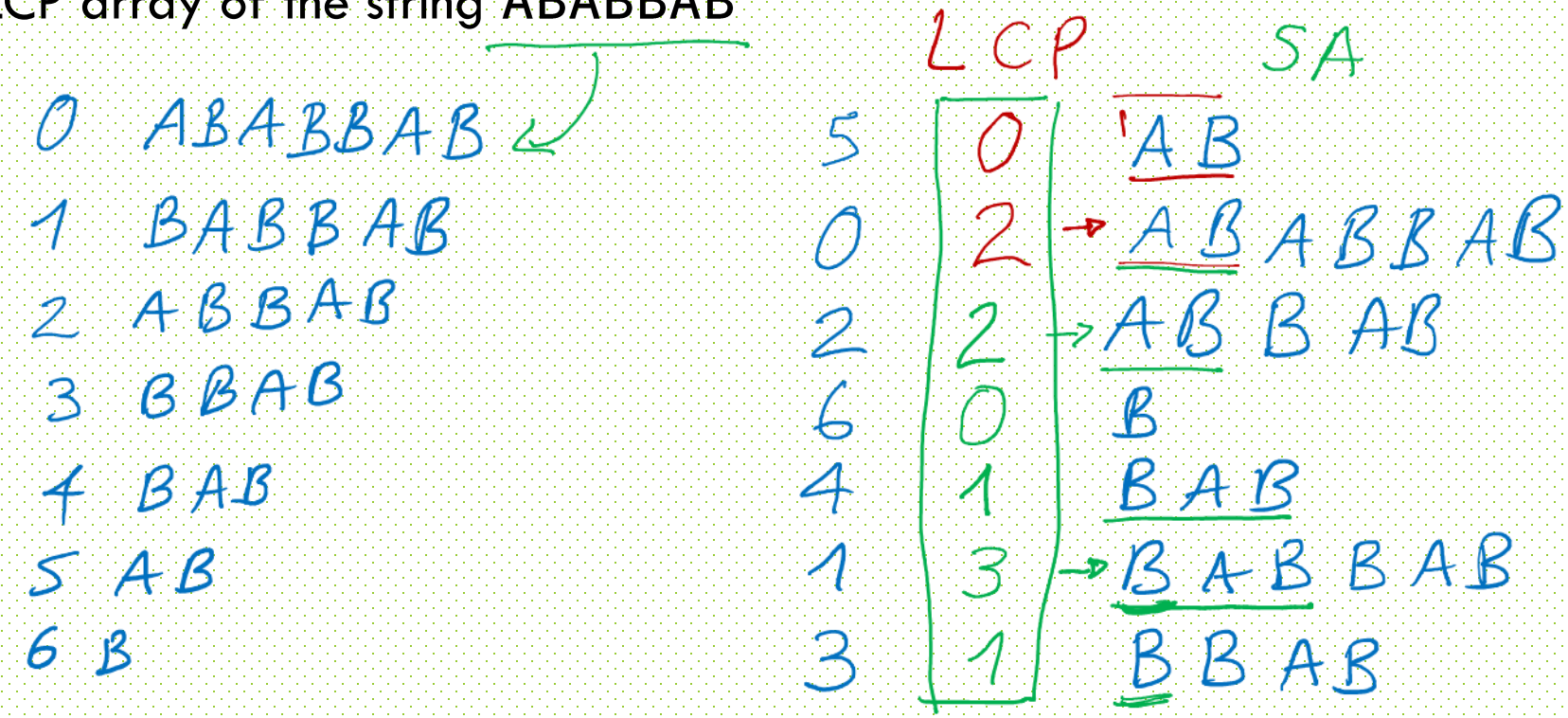


LONGEST COMMON PREFIX ARRAY

LONGEST COMMON PREFIX ARRAY

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

Find the LCP array of the string ABABBAB



LONGEST COMMON PREFIX ARRAY

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

Find the LCP array of the string ABABBAB

Suffix Array	Sorting
0 ABABBAB	5 ABABBAB
1 BABBAB	0 ABABBAB
2 ABBAB	2 ABBAB
3 BBAB	6 ABABBAB
4 BAB	4 BAB
5 AB	1 BABBAB
6 B	3 BBAB

LONGEST COMMON PREFIX ARRAY

Find the LCP array of the string ABABBAB

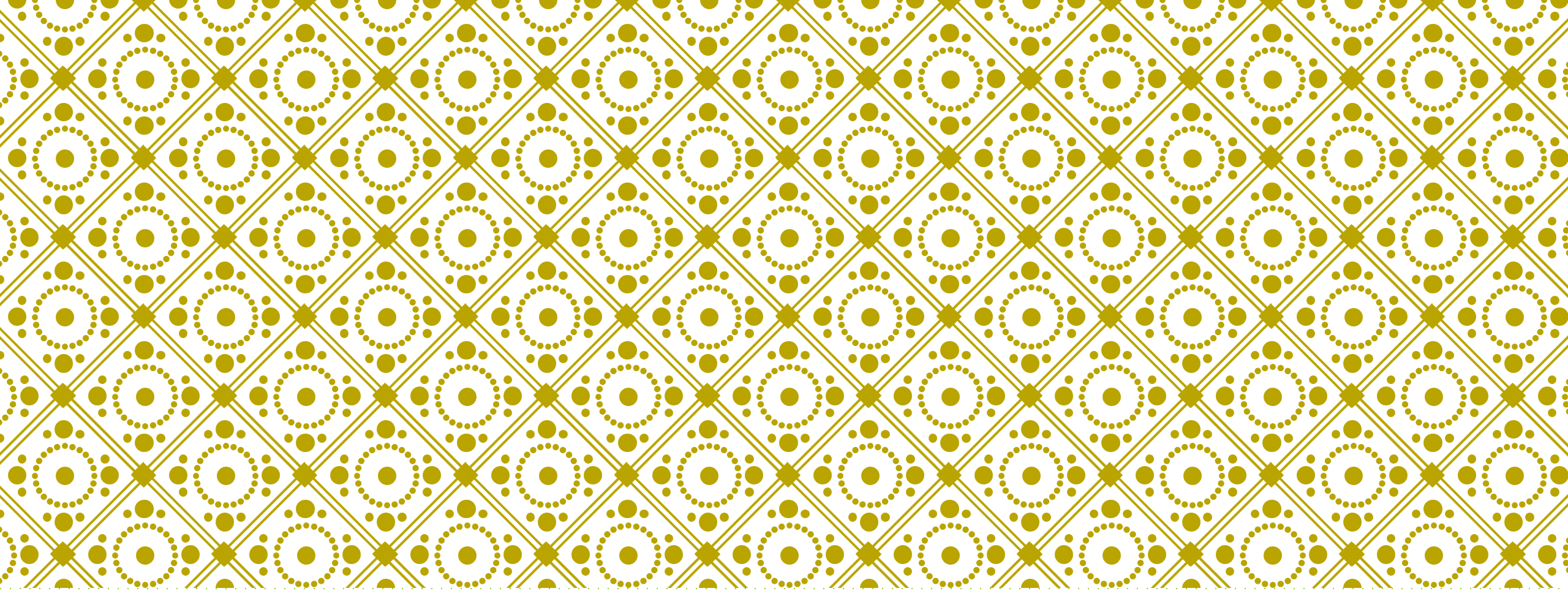
	Sorted	LCP
5	ABABBAB	0
0	ABABBAB	
2	ABABBAB	
6	ABABBAB	
4	ABABBAB	
1	ABABBAB	
3	ABABBAB	

LCP

0

	LCP	SA
5	0	AB
0	2	ABABBAB
2	2	ABBAB
6	0	B
4	1	BAB
1	3	BABBAB
3	1	BBAB

SA



APPLICATIONS SUFFIX ARRAY & LONGEST COMMON PREFIX ARRAY

FINDING UNIQUE SUBSTRINGS

Find/count all the unique substrings of a string.

Naïve approach: generate all substrings and put them in a set. This is O(n^2) time complexity

Better approach: Using LCP array. It provides a quick and space efficient solution

Other approaches: Rabin–Karp algorithm or Bloom filters

FINDING UNIQUE SUBSTRINGS

Find all the unique substrings of AZAZA

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

0 AZAZA
1 ZAZA
2 AZA
3 ZA
4 A

LCP

4	0	A
2	1	AZA
0	3	AZAZA
3	0	ZA
1	2	ZAZA

6

Number of all substrings

n(n+1)/2 = 15

Number of unique substrings: 9

FINDING UNIQUE SUBSTRINGS

String: AZAZA

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

AZAZA
AZAZA
AZAZA
AZAZA
AZAZA

LCP	Sorted Suffixes
0	AZAZA
1	AZAZA
3	AZAZA
0	AZAZA
2	AZAZA

FINDING UNIQUE SUBSTRINGS

String: AZAZA

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

Repeated	LCP	Sorted Suffixes (SA)
A	0	A
A	1	AZA
AZ	3	AZAZA
AZA	0	ZA
Z	2	ZAZA
ZA		

FINDING UNIQUE SUBSTRINGS

Number of unique substrings in a string:

$$\frac{n(n+1)}{2} - \sum_{i=1}^{n} LCP[i]$$

Number of substrings — Duplicates

FINDING UNIQUE SUBSTRINGS

Number of unique substrings in a string:

$$\frac{n(n+1)}{2} - \sum_{i=1}^{n} LCP[i]$$

Number of substrings — Duplicates

String: AZAZA

n = 5

$$\frac{5(5+1)}{2} - (1+3+0+2) = 9$$

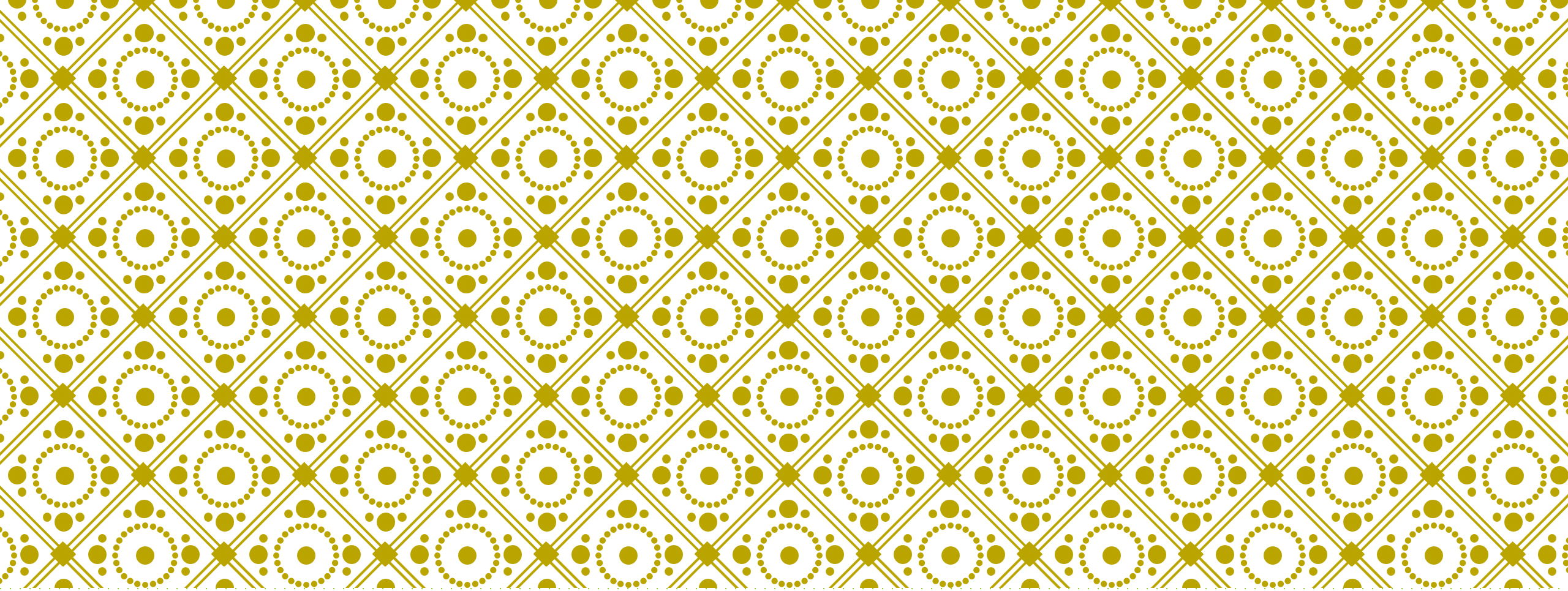
REPEATED SUBSTRINGS

String analysis often arises in applications from biology and chemistry, such as the study of DNA and protein molecules. One interesting problem is to find how many substrings are repeated (at least twice) in a long string.

Find the total number of repeated substrings in a string of at most 100000 alphabetic characters. Any unique substring that occurs more than once is counted.

As an example, if the string is “aabaab”, there are 5 repeated substrings: “a”, “aa”, “aab”, “ab”, “b”. If the string is “aaaaa”, the repeated substrings are “a”, “aa”, “aaa”, “aaaa”. Note that repeated occurrences of a substring may overlap (e.g. “aaaa” in the second case).

https://open.kattis.com/problems/substrings



LONGEST COMMON SUBSTRING

LONGEST COMMON SUBSTRING

k-Common Substring Problem

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?

Example: Consider n = 3, k = 2 with

S_1 = 'abca'

S_2 = 'bcad'

S_3 = 'daca'

S_1 = 'abca'

S_2 = 'bcad'

S_3 = 'daca'

LONGEST COMMON SUBSTRING

k-Common Substring Problem

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?

One approach is to use dynamic programming running in $O(n_1 * n_2 * \ldots * n_m)$, where n_i is the length of the string S_i.

Another approach is to use a suffix array which can find the solution in $O(n_1 + n_2 + \ldots + n_m)$ time

LONGEST COMMON SUBSTRING

Consider S_1 = 'abca' S_2 = 'bcad' S_3 = 'daca'

To find the LCS create a new larger string T which is the concatenation of all the strings S_i separated by unique sentinels.

$T = S_1 + '\#' + S_2 + '\$' + S_3 + '\%'$

T = abca#bcad$daca%

The sentinels are unique and lexicographically less than any of the characters in the strings

LONGEST COMMON SUBSTRING

T = abca#bcad$daca%

abca#bcad$daca%
bca#bcad$daca%
ca#bcad$daca%
a#bcad$daca%
#bcad$daca%
bcad$daca%
cad$daca%
ad$daca%
d$daca%
$daca%
daca%
aca%
ca%
a%
%

LCP

0
0
0
0
1
1
1
1
0
3
0
2
2
0
1

SA

Suffixes

#bcad$daca%
$daca%
%
a#bcad$daca%
a%
abca#bcad$daca%
aca%
ad$daca%
bca#bcad$daca%
bcad$daca%
ca#bcad$daca%
ca%
cad$daca%
d$daca%
daca%

LONGEST COMMON SUBSTRING

T = abca#bcad$daca%

If k = 3 we need one string of each color

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%

LONGEST COMMON SUBSTRING

T = abca#bcad$daca%

If k = 2 what is the LCS?

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%

LONGEST COMMON SUBSTRING

When different colors are not adjacent, we can use a sliding window to capture the correct amount of suffix colors. Adjusting the window in such a way that contains exactly k suffixes of different colors.

LCP	Suffixes
…	
0	AGT$CGAAGC%
2	AGC%
3	AGC#AGAAGT$CGAAGC%
2	AGAAGT$CGAAGC%
5	AGAAGC#AGAAGT$CGAAGC%
1	AAGT$CGAAGC%
…	

LONGEST COMMON SUBSTRING

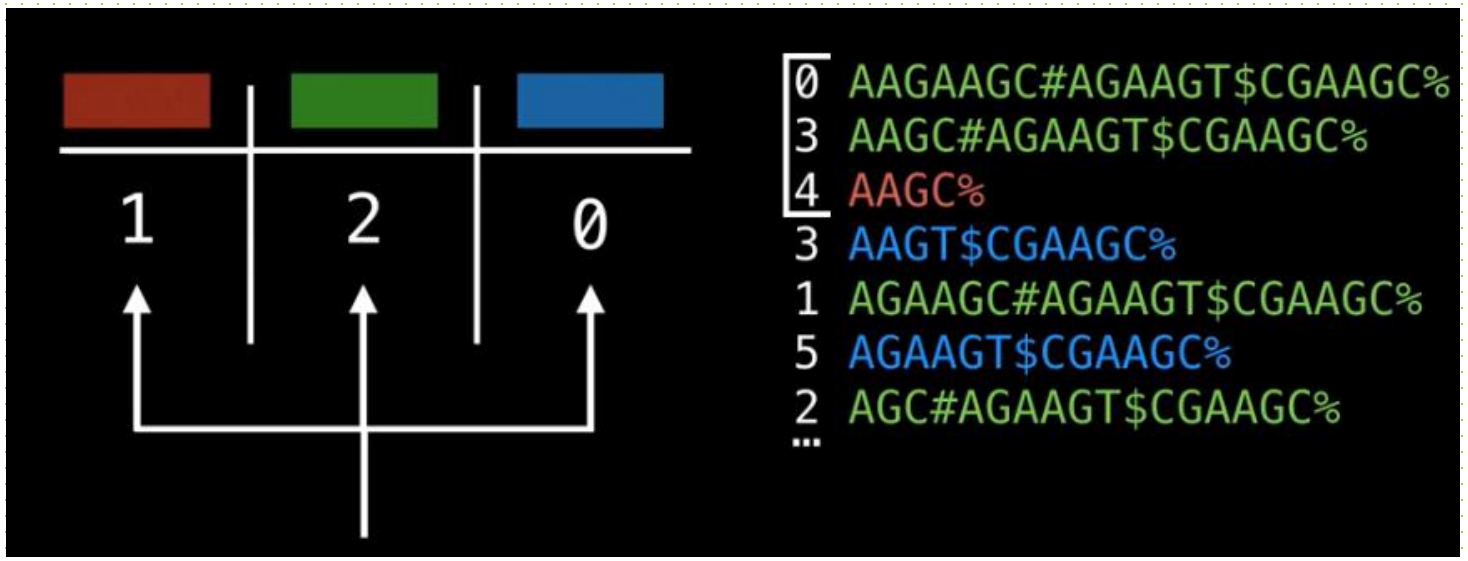
For each valid window perform a range query on the LCP array between the bottom and top endpoints. The LCS will be the maximum LCP value for all possible windows.

This can be solved in O(n) time for all windows.

We will need a DS (hashtable) to track the colors in the sliding window.

LONGEST COMMON SUBSTRING

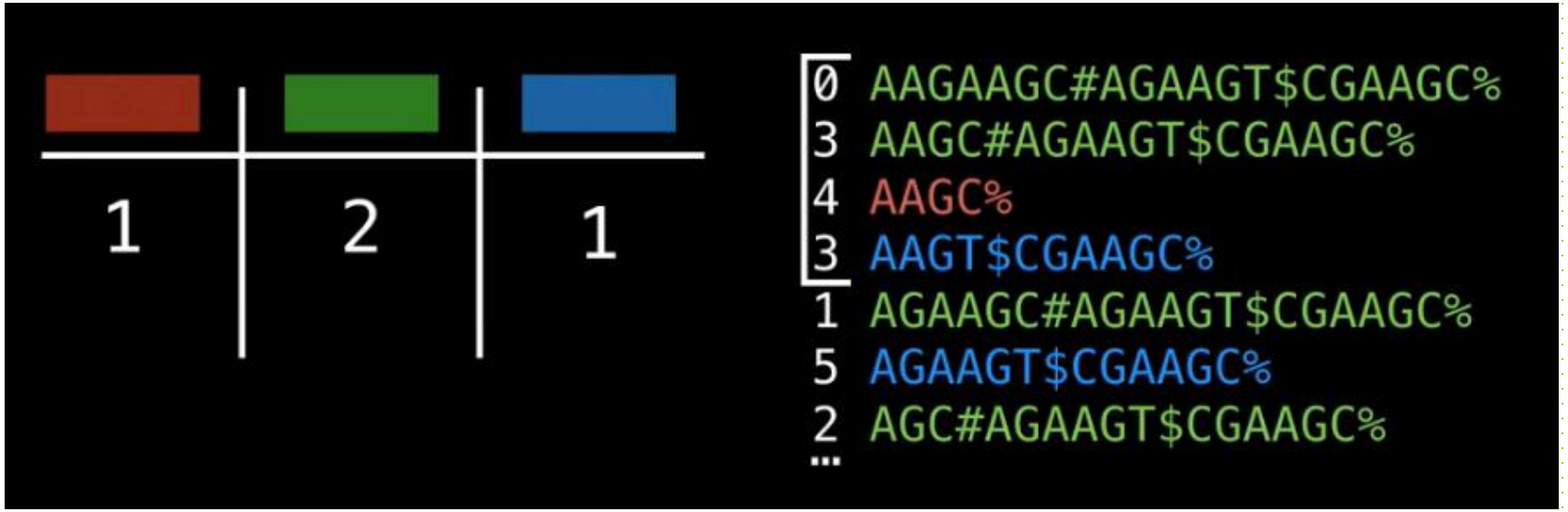
Using a hashtable to track the colors in the sliding window.



A valid window requires at least k or more of these to be greater than one

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.

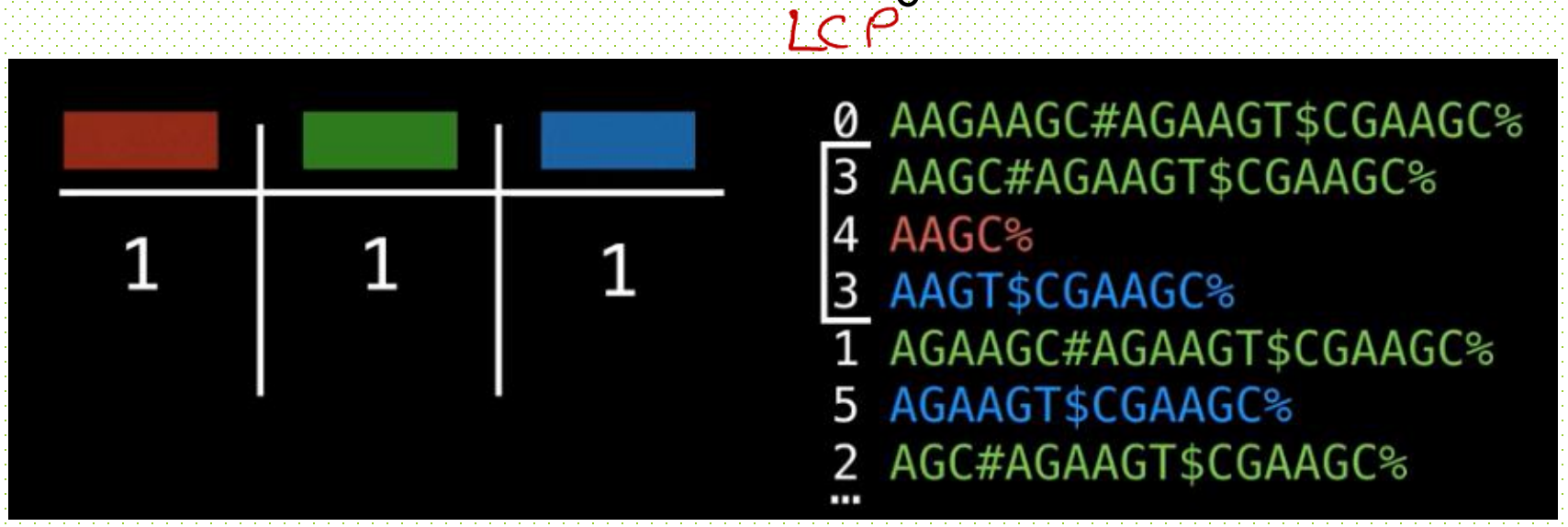


k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.

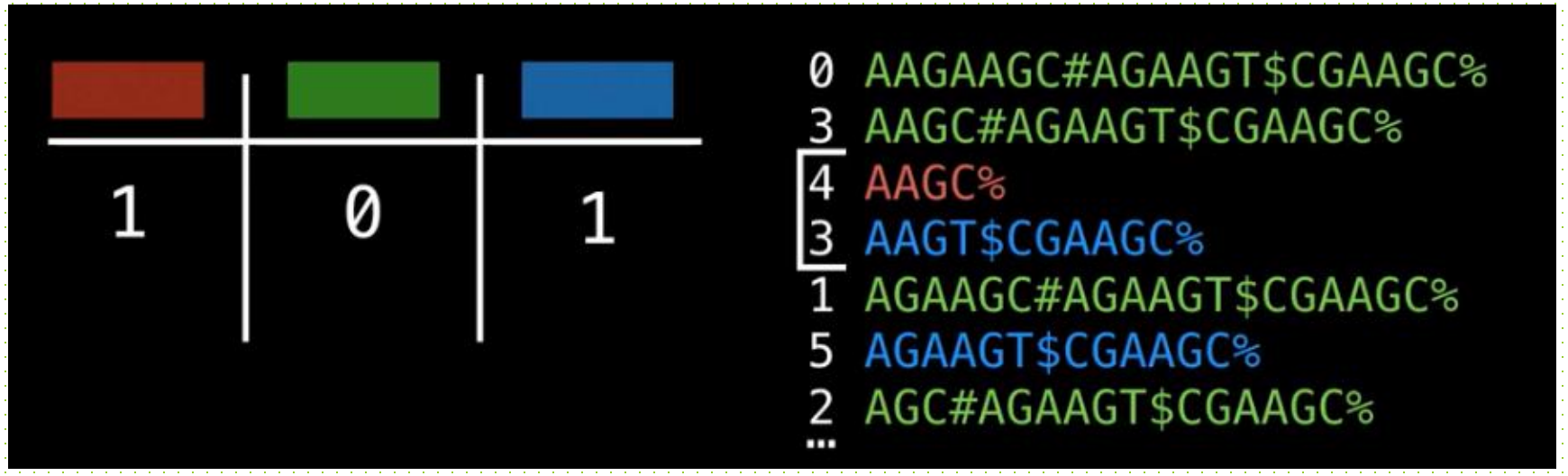


k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.

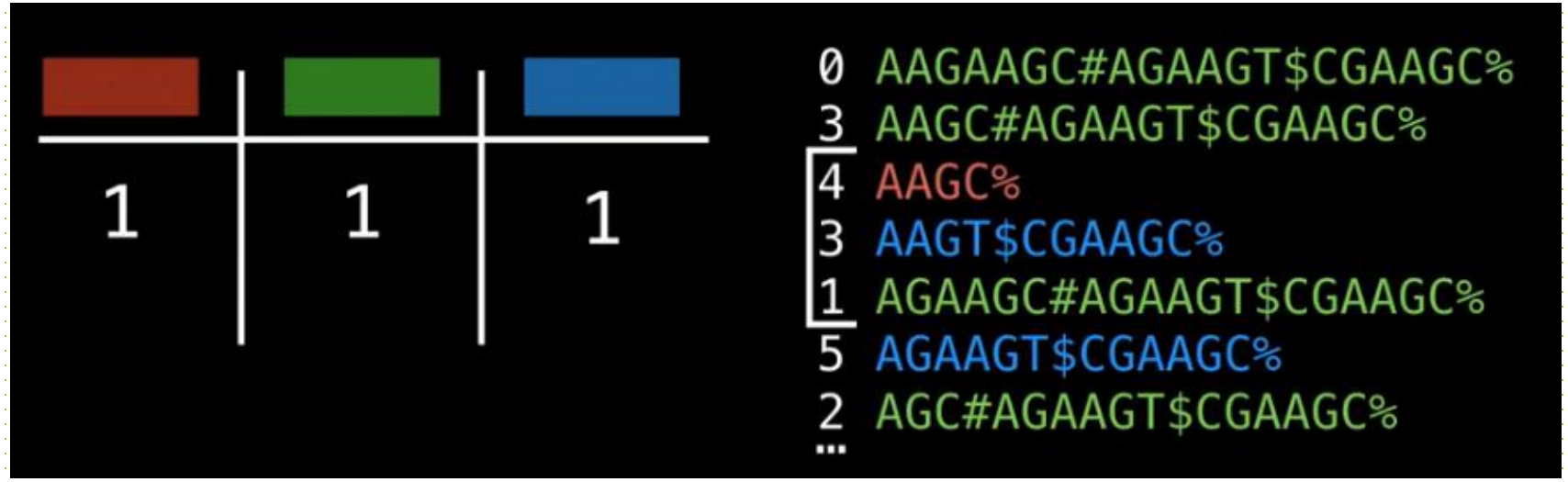


k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.

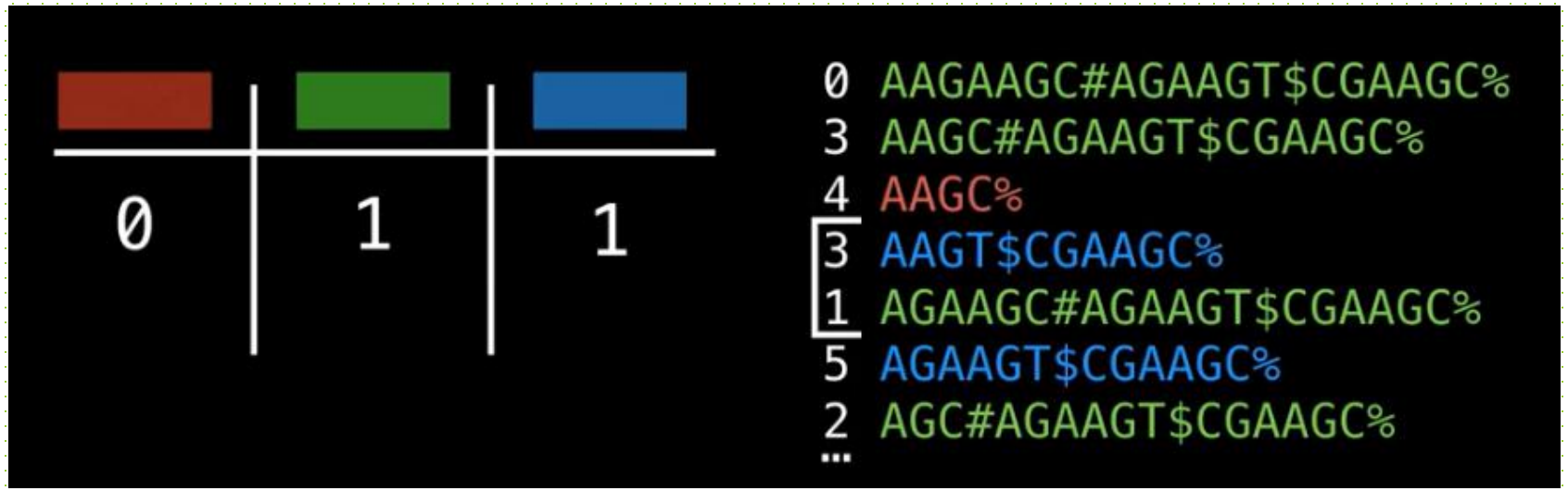


k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.

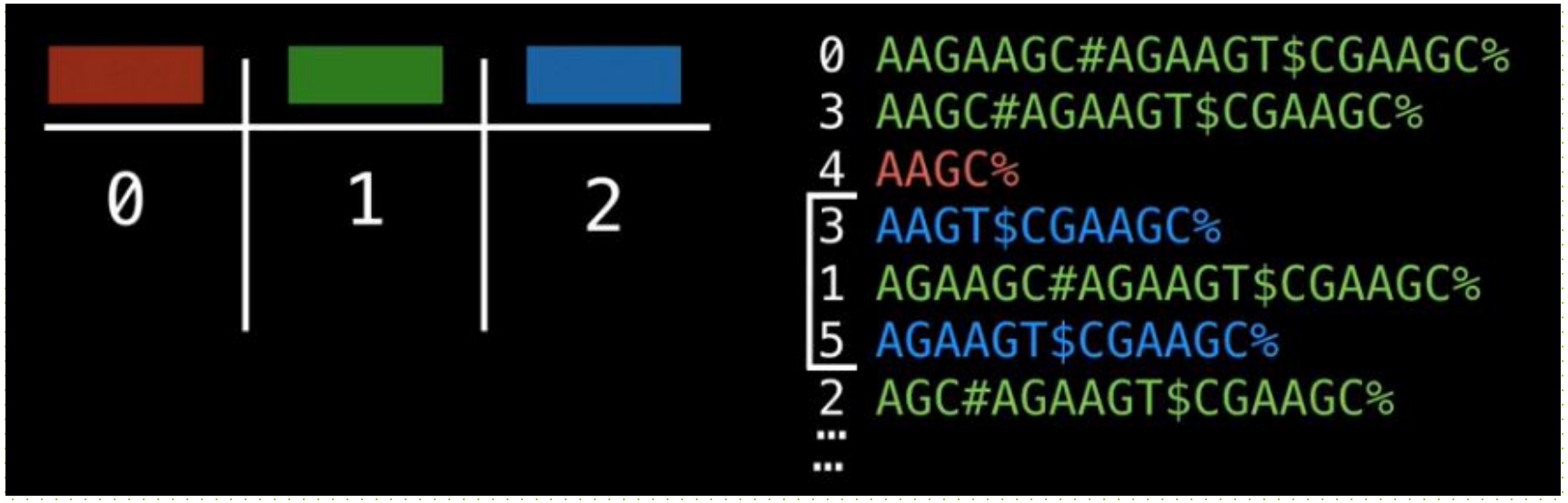


k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.

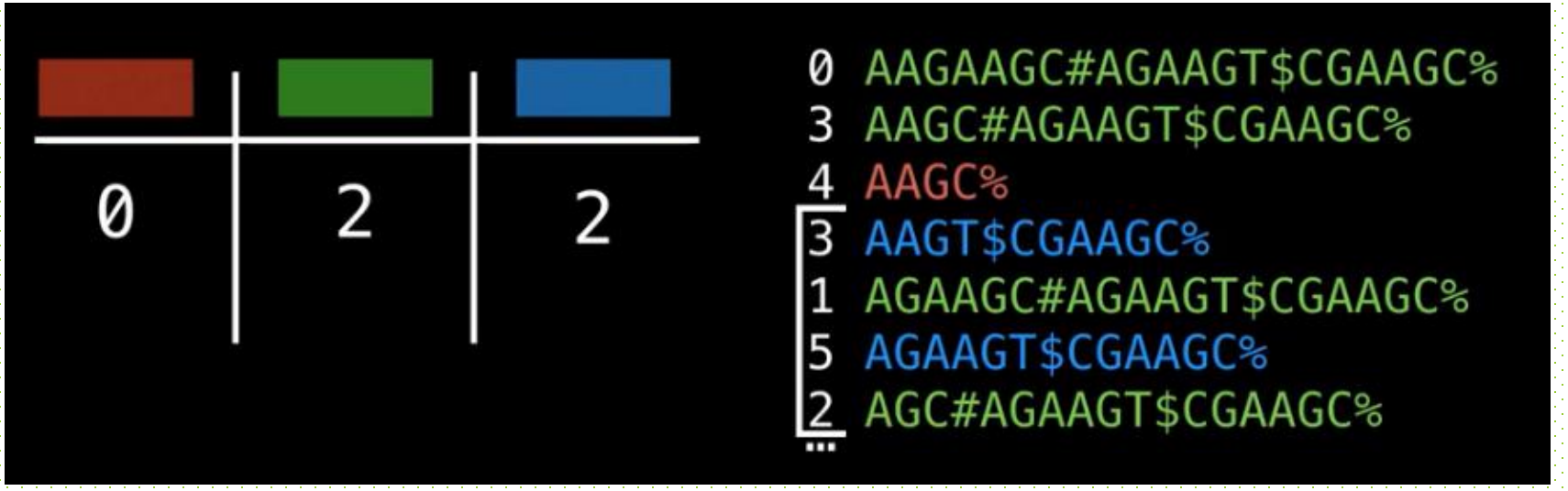


k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LONGEST COMMON SUBSTRING

Using a hashtable to track the colors in the sliding window.



k = 3

Expand window to acquire the correct amount of each color and retract window when the correct amount of colors is met

LIFE FORMS

Given the DNA sequences of several life forms represented as strings of letters, you are to find the longest substring that is shared by more than half of them.

https://open.kattis.com/problems/lifeforms

LCS EXAMPLE

Consider four strings S_1, S_2, S_3, S_4. Find the LCS that appear in at least two of the strings k = 2

S_1 = AABC, S_2 = BCDC

S_3 = BCDE, S_4 = CDED

T = AABC#BCDC$BCDE%CDED&

LCS(S1, S2, S3, S4) = {BCD, CDE}