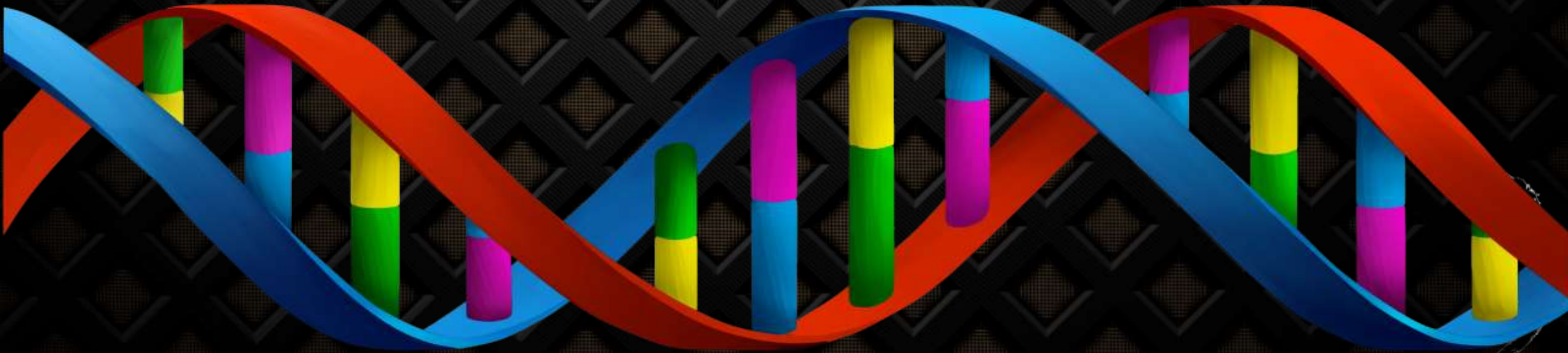
LONGEST COMMON SUBSEQUENCE (LCS)



LONGEST COMMON SUBSEQUENCE

What is Longest common subsequence ?

The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences).

WHAT IS SUBSEQUENCES ?

Suppose you have a sequence

X = < A,B,C,D,E,F,G>

of elements over a finite set S.

A sequence Y = <B,C,E,G >

over S is called a subsequence of X if and only if it can be obtained from X by deleting elements.

WHAT IS **COMMON SUBSEQUENCES** ?

Suppose that X and Y are two sequences over a set S.

If , A=<A,B,C,E,D,G,F,H,K>

B=<A,B,D,F,H,K>

then a common subsequence of *X* and *Y* could be

Z=<A,F,K>

We say that Z is a common subsequence of X and Y if and only if

Z is a subsequence of X

Z is a subsequence of Y

THE LONGEST COMMON SUBSEQUENCE PROBLEM

Given two sequences X and Y over a set S, the longest common subsequence problem asks to find a common subsequence of X and Y that is of maximal length.

NAÏVE SOLUTION

Let X be a sequence of length m,

and Y a sequence of length n.

Check for every subsequence of X whether it is a subsequence of Y, and return the longest common subsequence found.

There are 2^m subsequences of X. Testing a sequences whether or not it is a subsequence of Y takes O(n) time. Thus, the naïve algorithm would take $O(n2^m)$ time.

FACTS OF LCS

INPUT: two strings

OUTPUT: longest common subsequence

ACTGAACTCTGTGCACT

TGACTCAGCACAAAAAC

FACTS OF LCS

INPUT: two strings

OUTPUT: longest common subsequence

ACTGAACTCTGTGCACT

TGACTCAGCACAAAAAC

FACTS OF LCS

Brute Force

X= ABCBDAB
Y= BDCABA

Elements of X is m=7
Elements of Y is n=6
So, the complexity will calculate by O (n2^m)

FACTS OF LCS

Brute Force

Strength

- **Wide applicability, simplicity**
- **Reasonable algorithms for some important problems such as searching, string matching, and matrix multiplication**
- **Standard algorithms for simple computational tasks such as sum and product of n numbers, and finding maximum or minimum in a list**

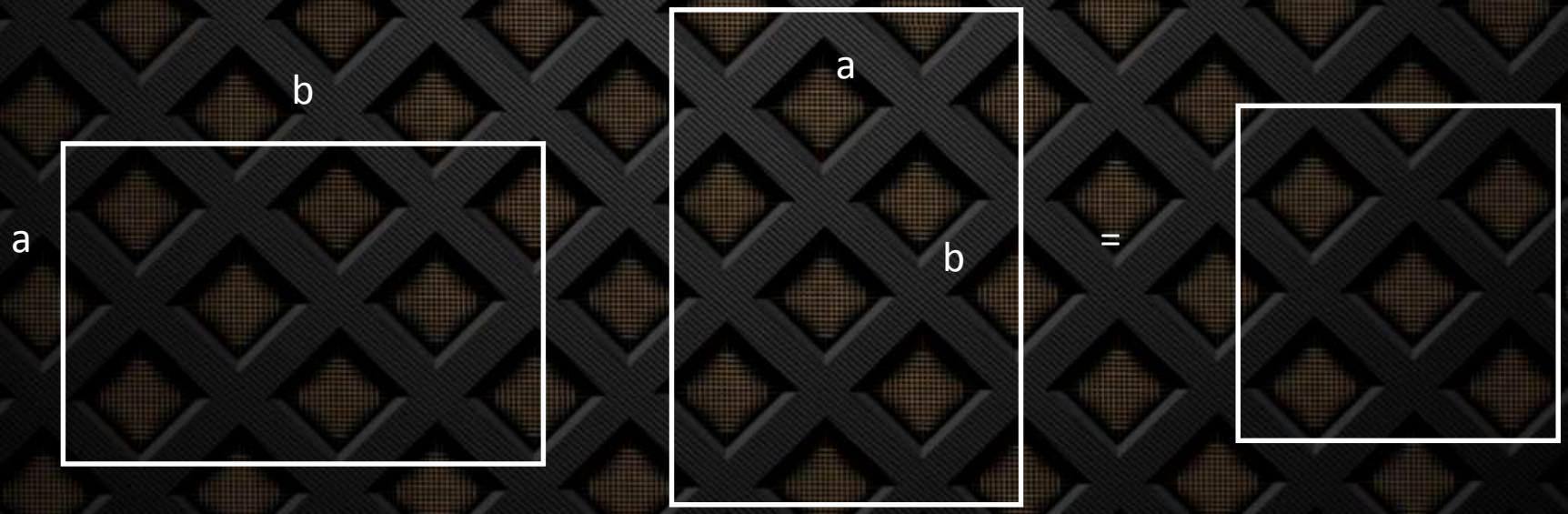
FACTS OF LCS

Brute Force

Weakness

- **Brute Force approach rarely yields efficient algorithms**
- **Some brute force algorithms are unacceptably slow**
- **Brute Force approach is neither as constructive nor creative as some other design techniques**

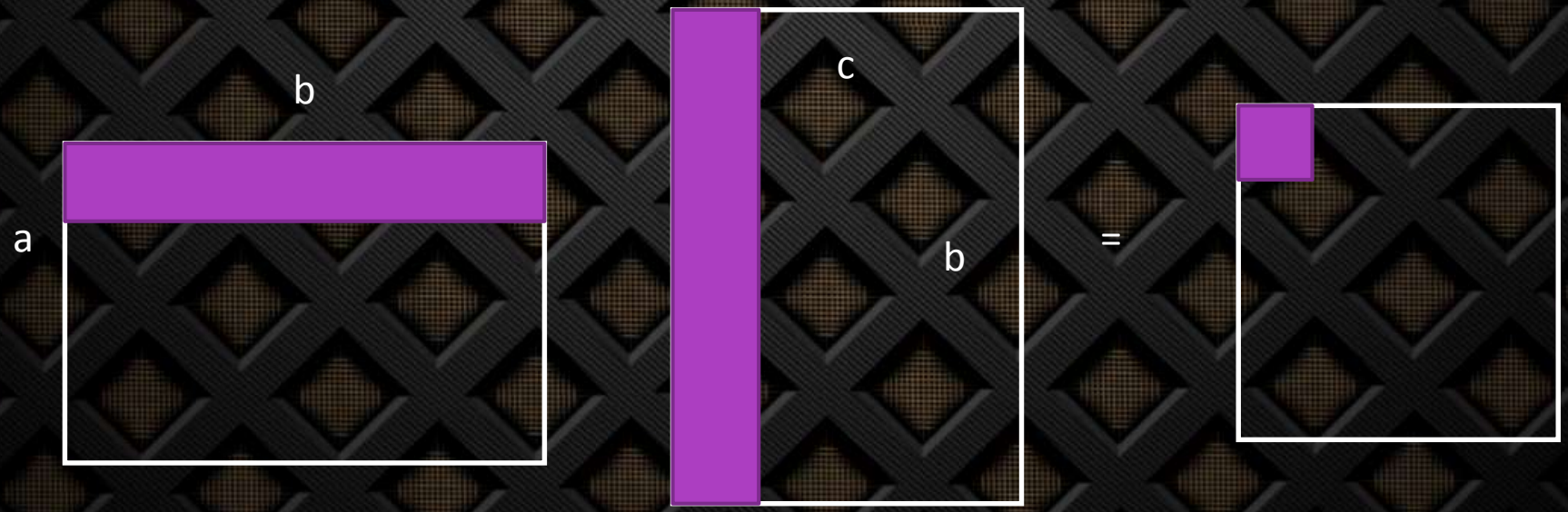
Facts OF LCS
Dynamic programming



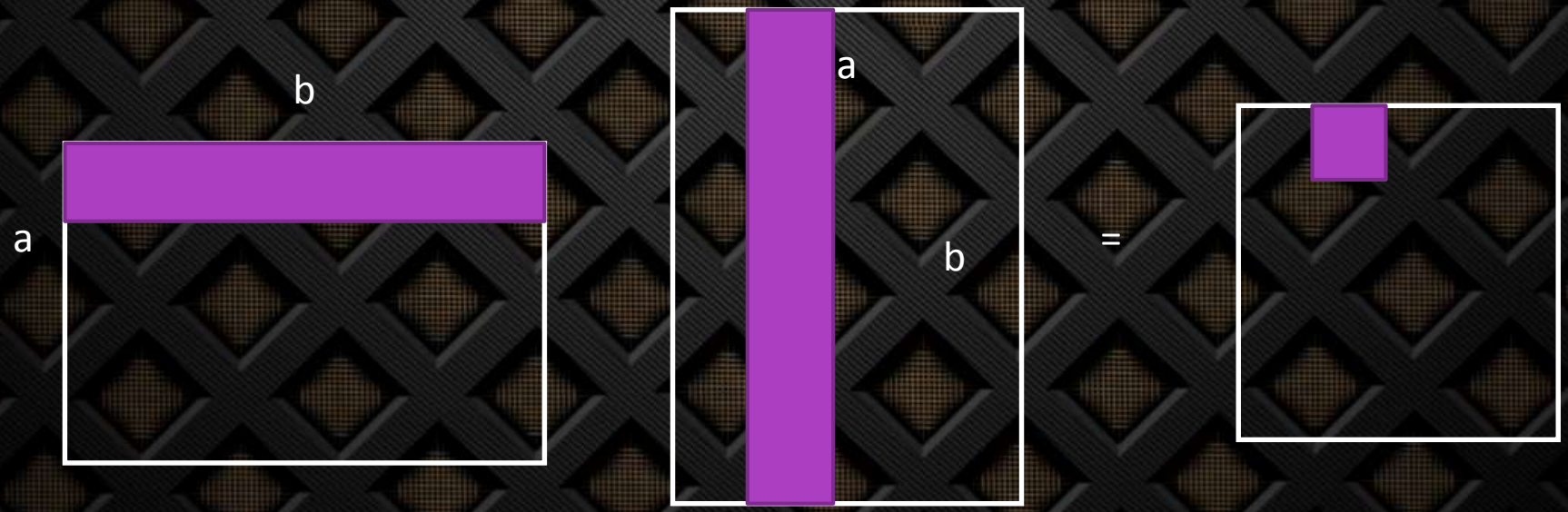
A = a x b matrix

How many operations to compute AB ?

Facts OF LCS
Dynamic programming

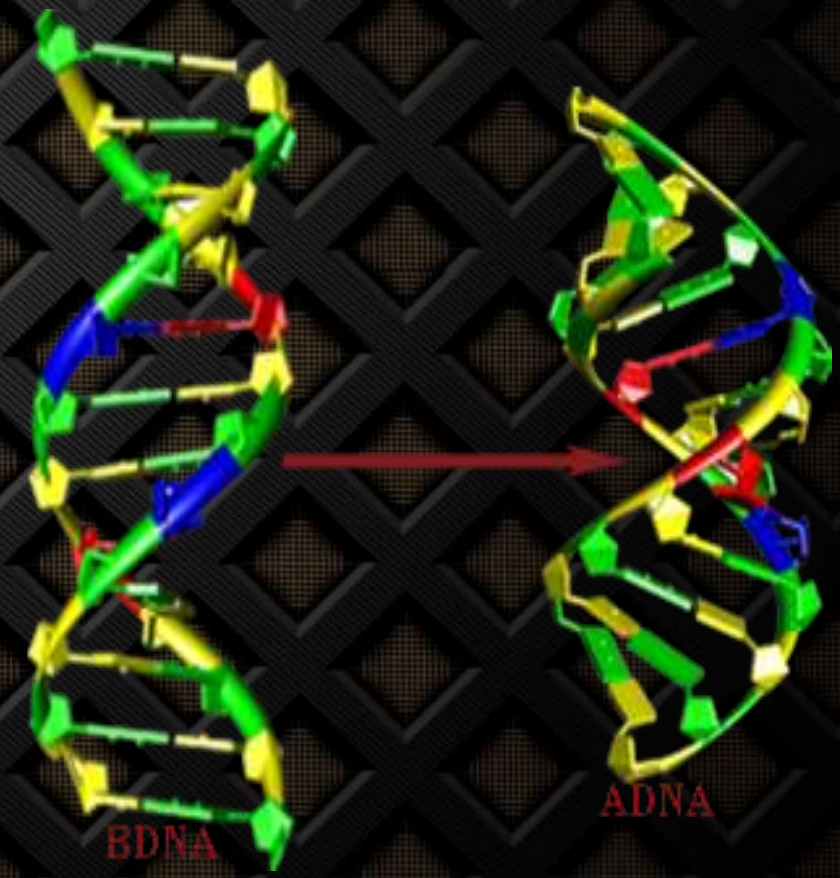


Facts OF LCS
Dynamic programming



Need to compute = O (a×b)

Work Examples



To Compare DNA of two (or more) Different organisms

EXAMPLE

Assume two DNA sequence

X = {ATGCTTC}

Y = {GCTCA}

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Xi → / Yj ↓		0	1	2	3	4	5	6	7
			A	T	G	C	T	T	C
0									
1	G								
2	C								
3	T								
4	C								
5	A								

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj →

Xi →

		0	1	2	3	4	5	6	7
			A	T	G	C	T	T	C
0		0	0	0	0	0	0	0	0
1	G	0							
2	C	0							
3	T	0							
4	C	0							
5	A	0							

Z[j,i]

Here I = 1, j = 1

Z[1,1]

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj ↓ / Xi →

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0							
2	C	0							
3	T	0							
4	C	0							
5	A	0							

Maximum of two box z[J-1, i] and [J, i-1]

X	Y
A	G

Not Match

Z[1,1]

Z[j-1, i]=Z[1-1, 1]= Z[0,1]

Z[j, i-1]=Z[1, 1-1]= Z[1,0]

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj

Xi

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0						
2	C	0							
3	T	0							
4	C	0							
5	A	0							

X	Y	
A	G	

Not Match

Lets Take from Upper one

Arrow indicate from where you Take the maximum.

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj ↓ , Xi →

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0					
2	C	0							
3	T	0							
4	C	0							
5	A	0							

X	Y	Max	arrow
T	G	0	←

Not Match

Lets Take from left one

Arrow indicate from where you Take the maximum.

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj

Xi

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0					
2	C	0							
3	T	0							
4	C	0							
5	A	0							

When match arrow will be diagonal because we will increment the value of this cell Z[i-1, j-1]

X	Y	Max	arrow
G	G		↖

Match

0 + 1 = 1

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj →

Xi →

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0	1				
2	C	0							
3	T	0							
4	C	0							
5	A	0							

X	Y	Max	arrow
G	G		↖

Match

Incremented value X[i-1] Y[j-1]

Z[I,j] = Z[3,1]

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj

Xi

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0	1	1			
2	C	0							
3	T	0							
4	C	0							
5	A	0							

X	Y	Max	arrow
C	G	1	←

Not Match

Lets Take from left one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj →

Xi ↓

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0	1	1	1		
2	C	0							
3	T	0							
4	C	0							
5	A	0							

X	Y	Max	arrow
T	G	1	←

Not Match

Lets Take from left one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj

Xi

		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0	1	1	1	1	
2	C	0							
3	T	0							
4	C	0							
5	A	0							

X	Y	Max	arrow
T	G	1	←

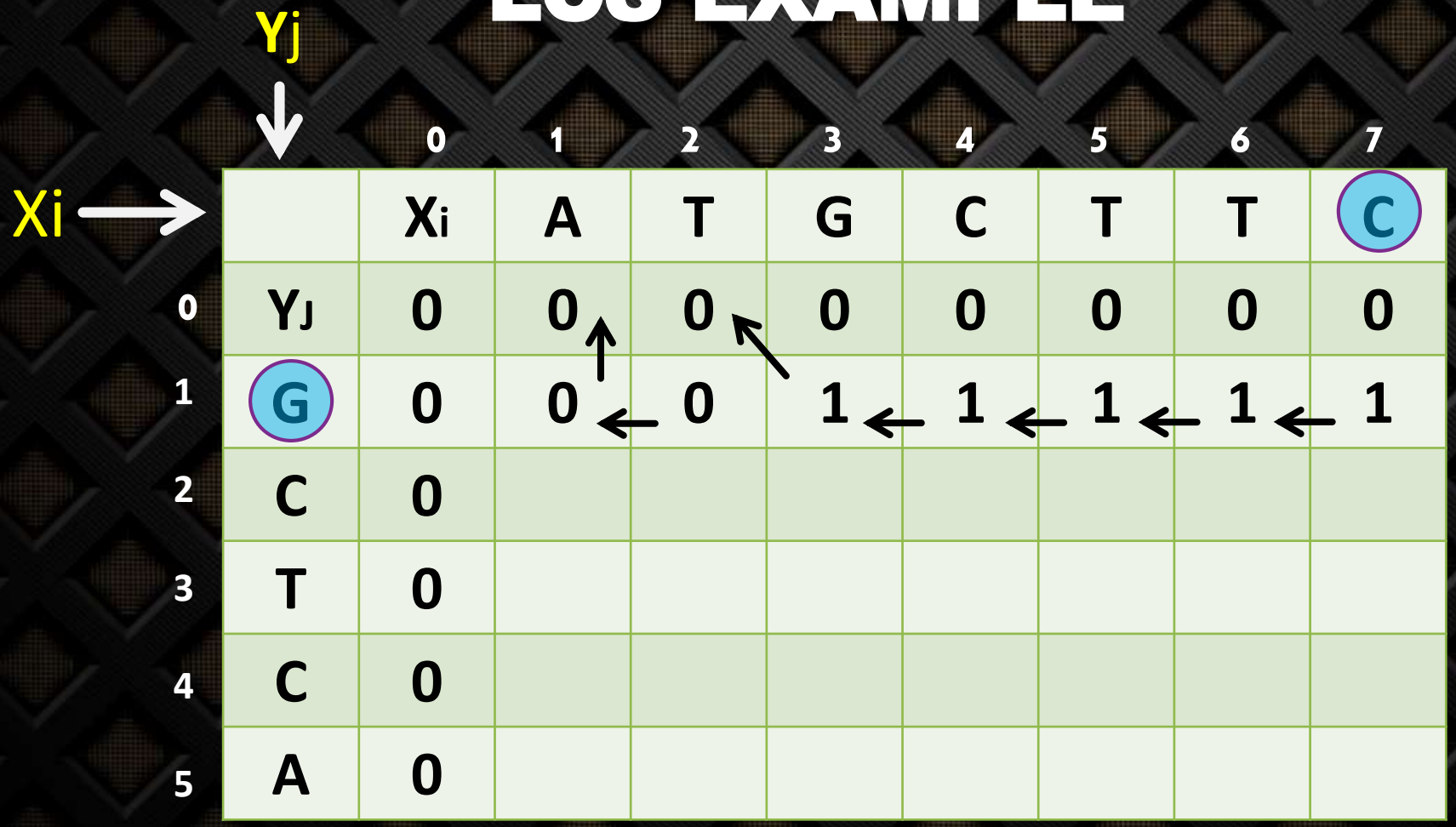
Not Match

Lets Take from left one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}



X	Y	Max	arrow
C	G	1	←

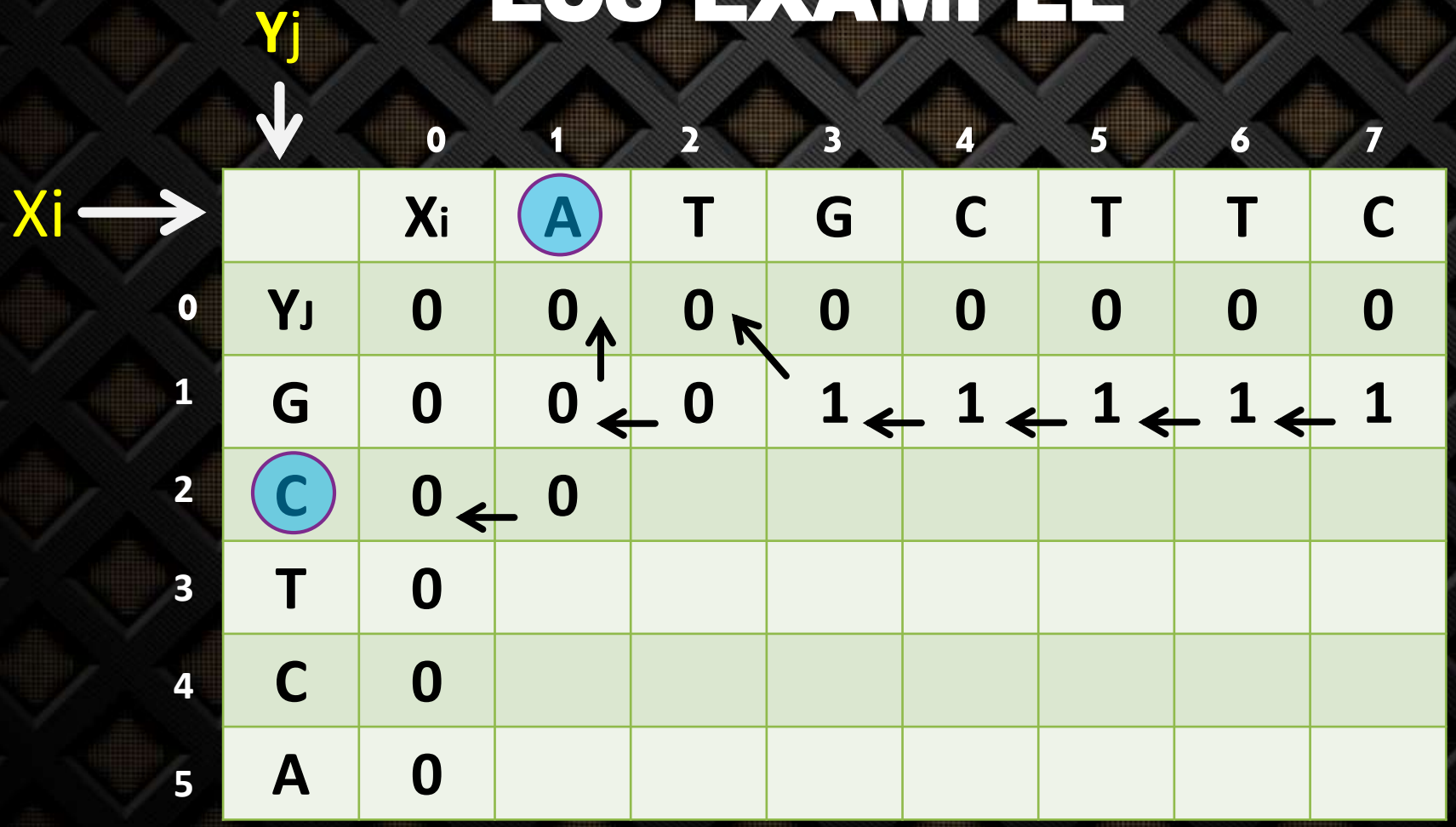
Not Match

Lets Take from left one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}



X	Y	Max	arrow
A	C	0	←

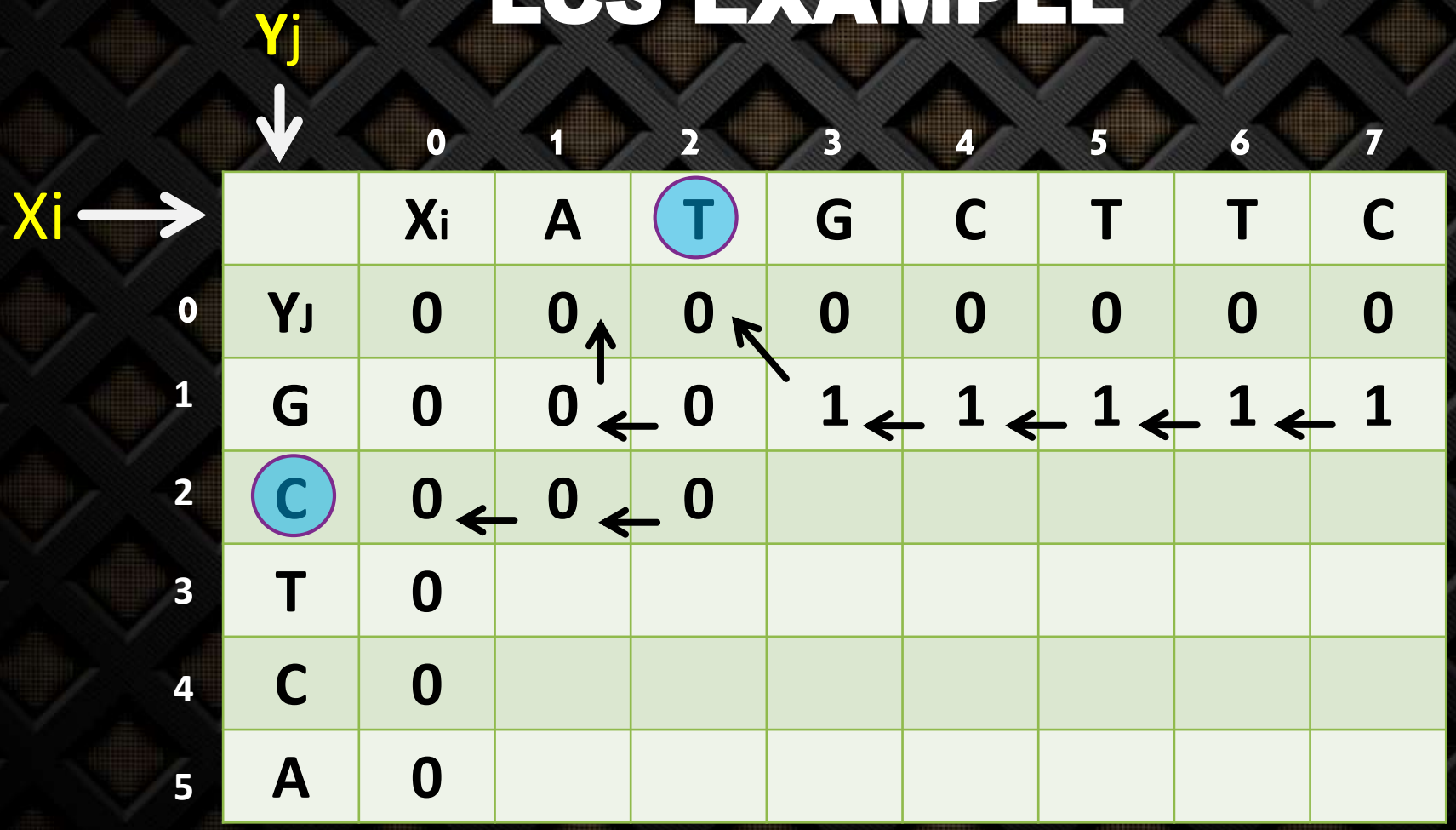
Not Match

Lets Take from left one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}



X	Y	Max	arrow
A	C	0	←

Not Match

Lets Take from Upper one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

Yj →

Xi ↓

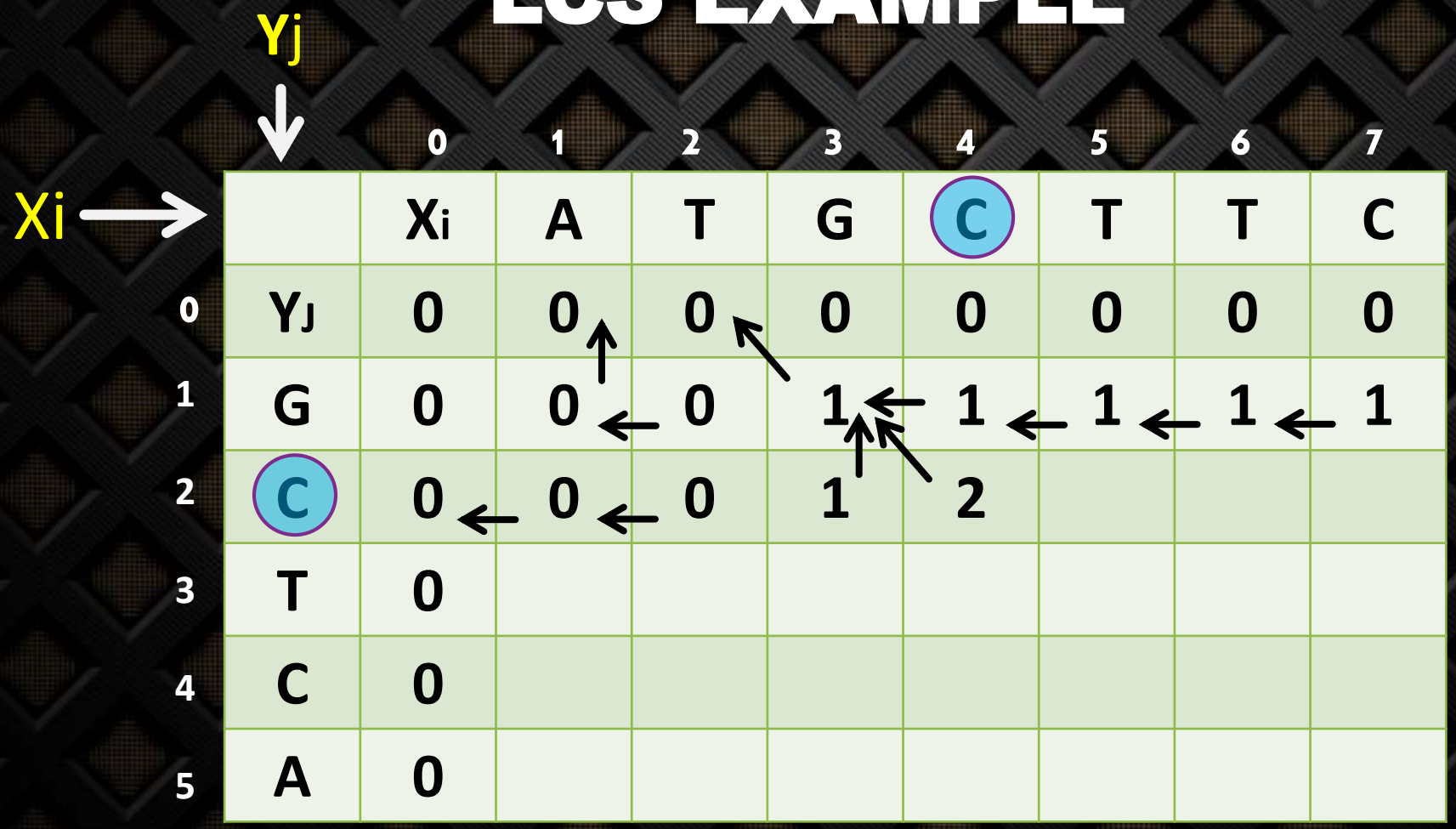
		0	1	2	3	4	5	6	7
		Xi	A	T	G	C	T	T	C
0	Yj	0	0	0	0	0	0	0	0
1	G	0	0	0	1	1	1	1	1
2	C	0	0	0	1				
3	T	0							
4	C	0							
5	A	0							

X	Y	Max	arrow
G	C	1	↑

Not Match

Lets Take from left one

LCS EXAMPLE



X = {ATGCTTC}

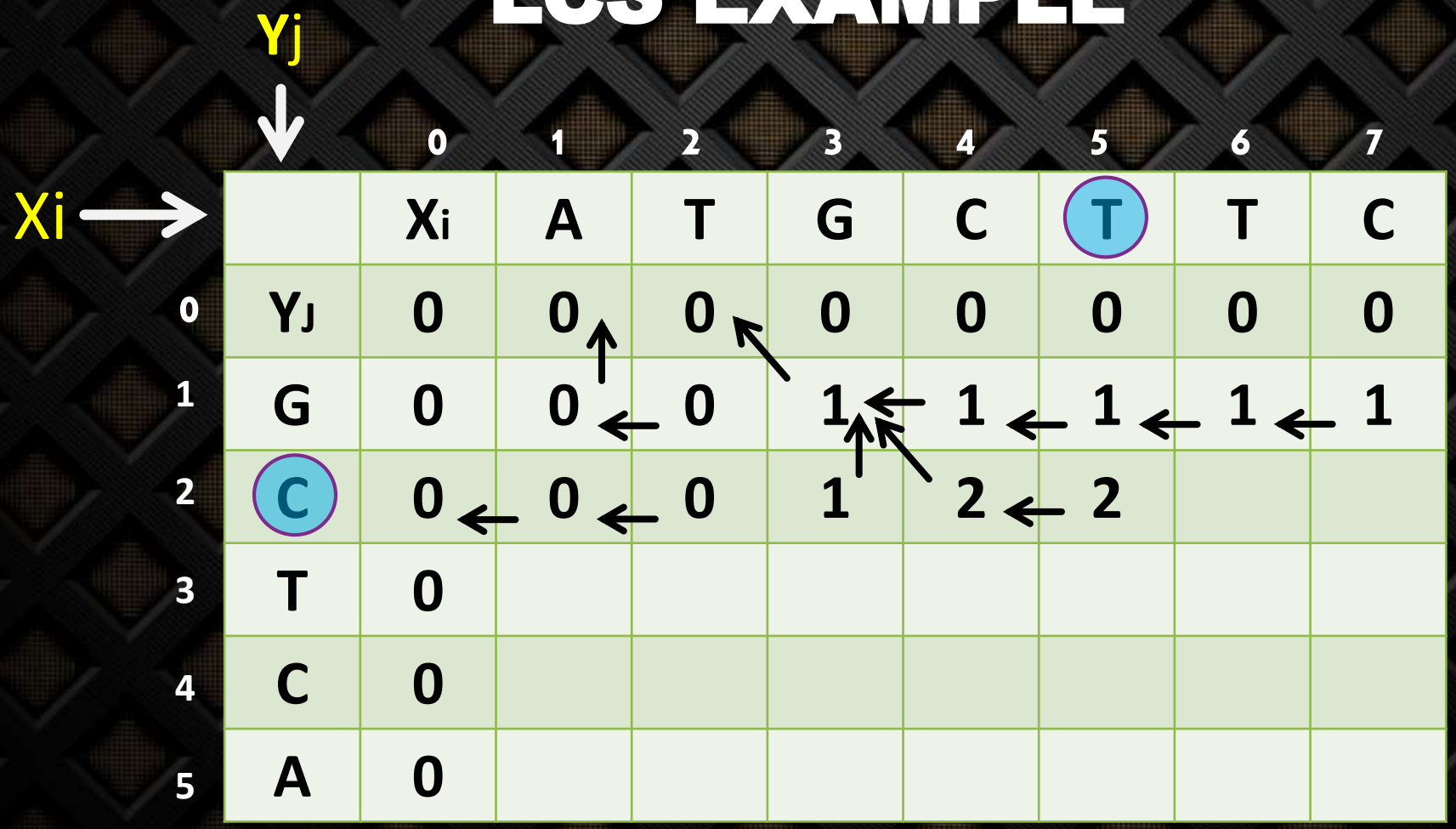
Y = {GCTCA}

X	Y	Max	arrow
C	C		↖

Match

Increment Z[i-1,j-1]

LCS EXAMPLE



X = {ATGCTTC}

Y = {GCTCA}

X	Y	Max	arrow
T	C	2	←

Not Match

Lets Take from left one

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

		Yj	0	1	2	3	4	5	6	7
			Xi	A	T	G	C	T	T	C
0		Yj	0	0	0	0	0	0	0	0
1		G	0	0	0	1	1	1	1	1
2		C	0	0	0	1	2	2	2	2
3		T	0	0	1	1	2	3	3	3
4		C	0	0	1	1	2	3	3	4
5		A	0	1	1	1	2	3	3	4

Xi →

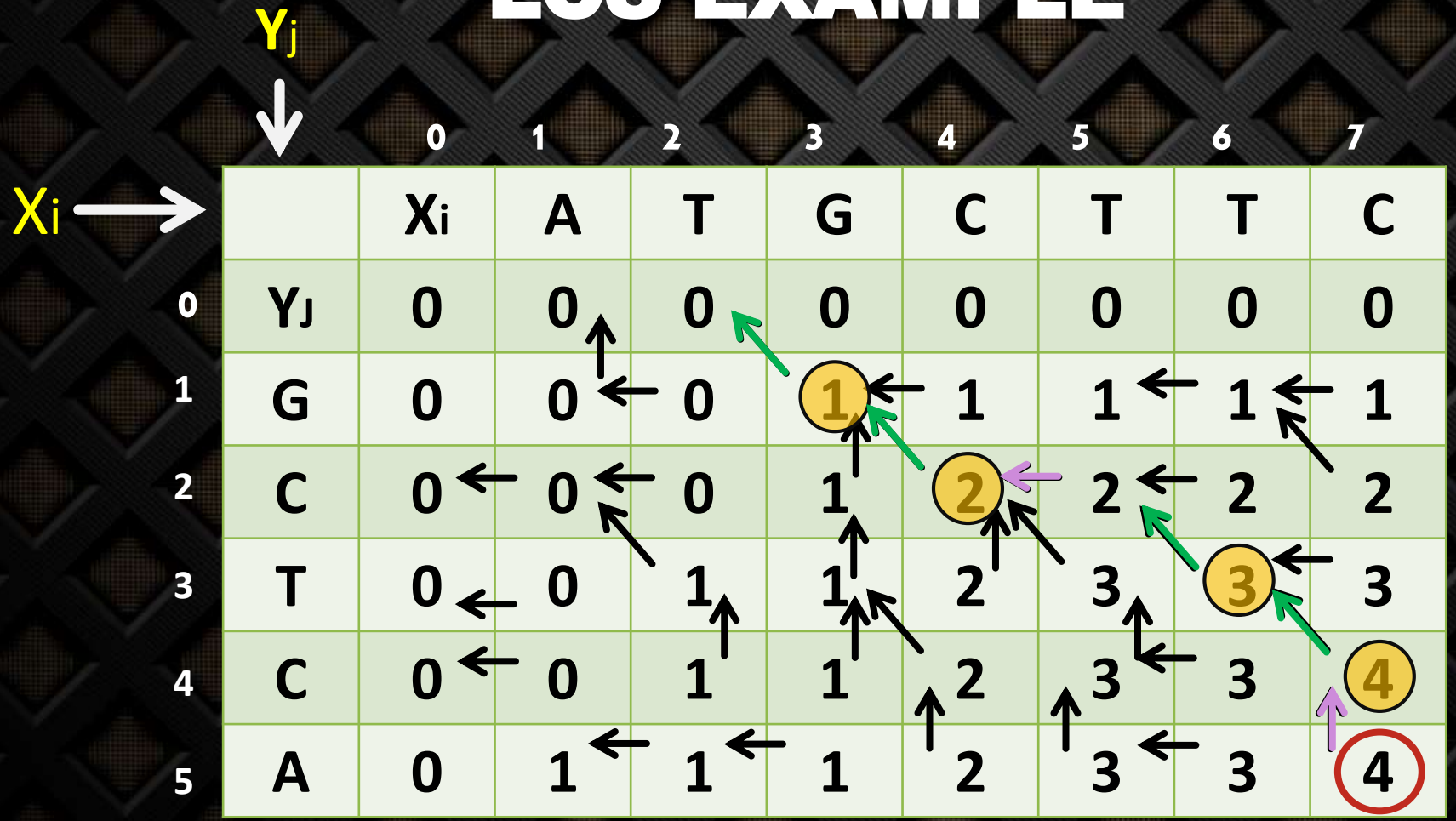
In the same way...

Traceback Approach

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}



Firstly have to point out highest value

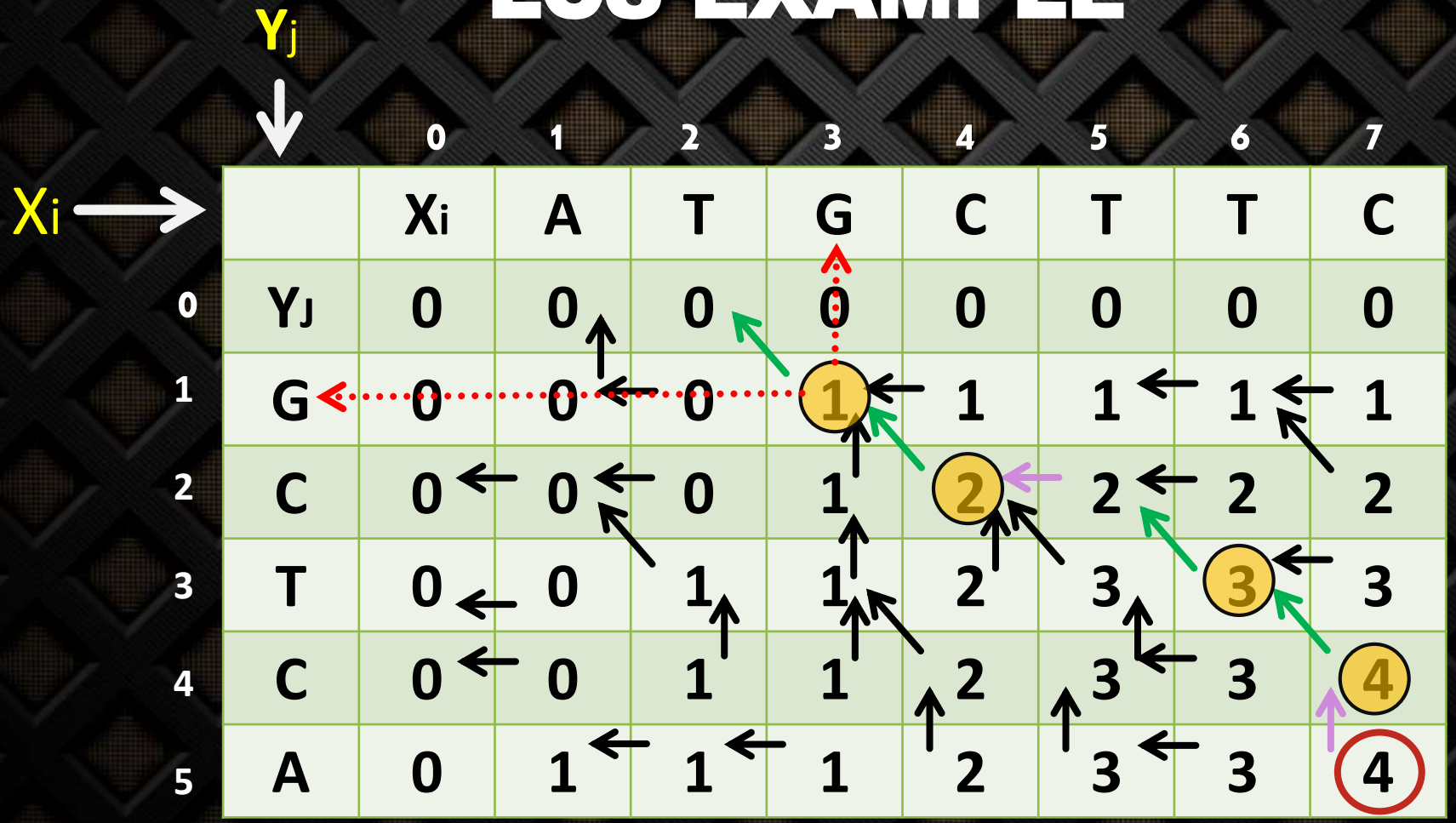
For left and upper arrow we will follow the direction

For diagonal arrow we will point out the character for this cell.

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

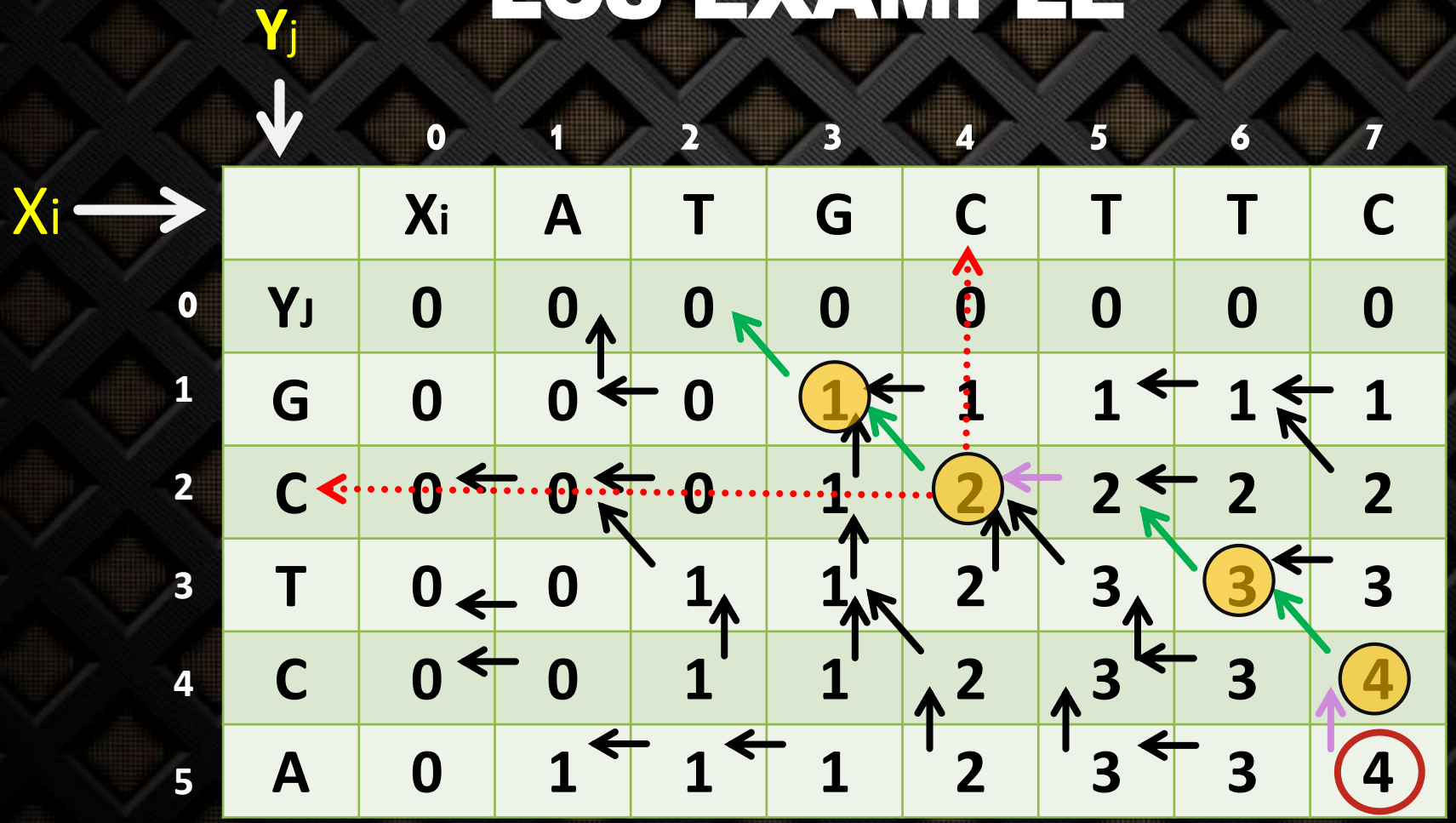


LCS Z= G

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

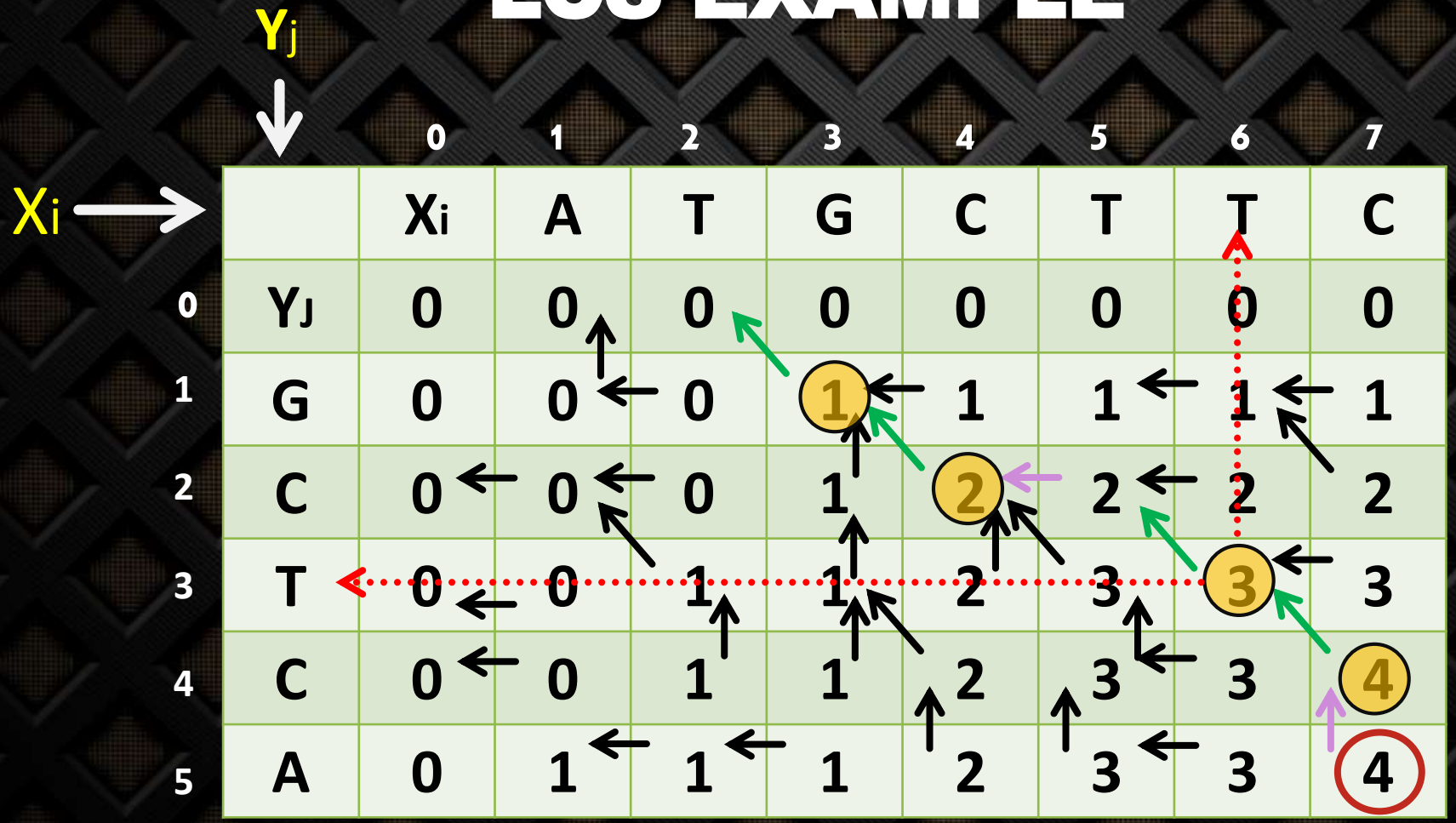


LCS Z= GC

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

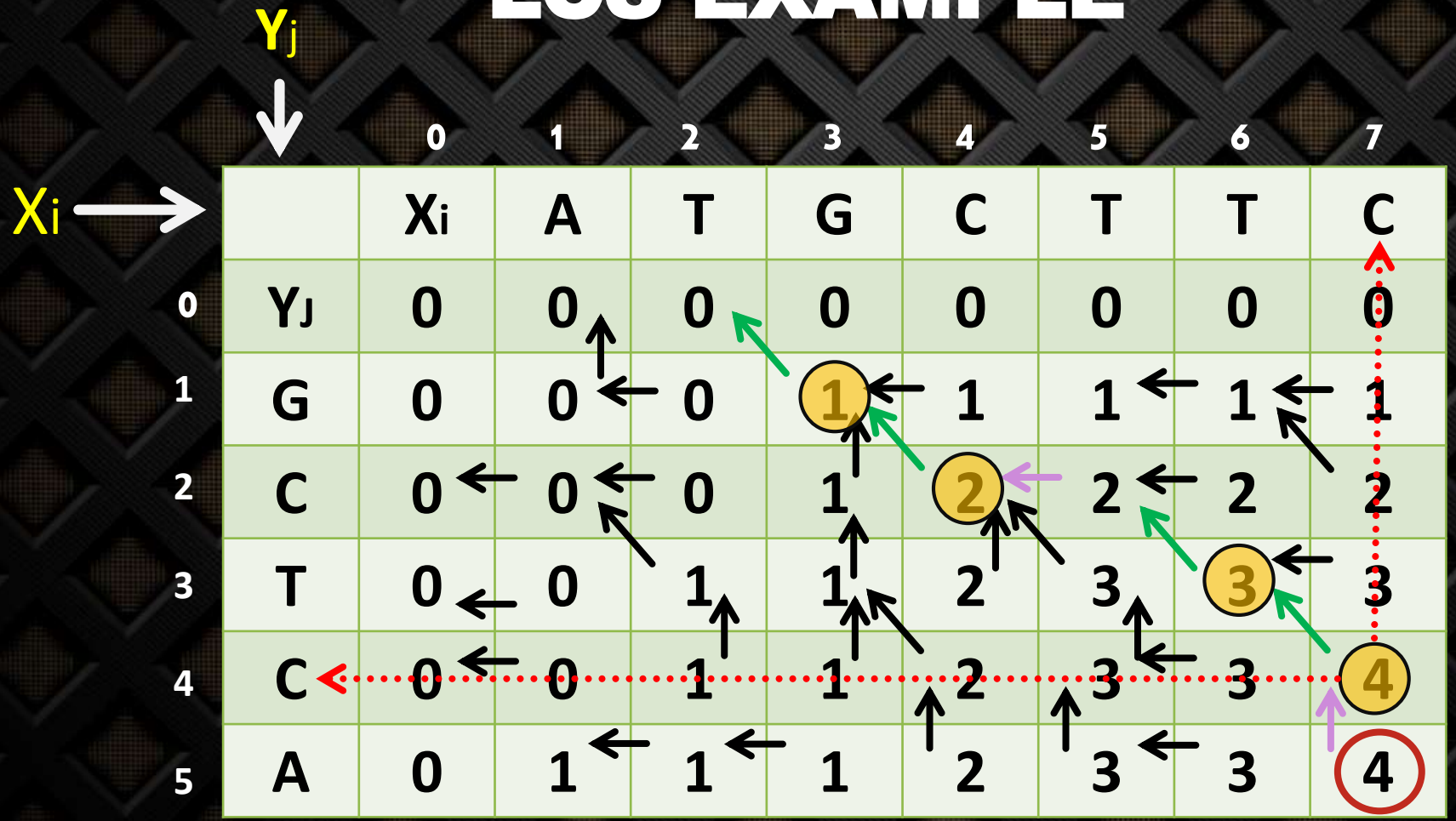


LCS Z= GCT

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

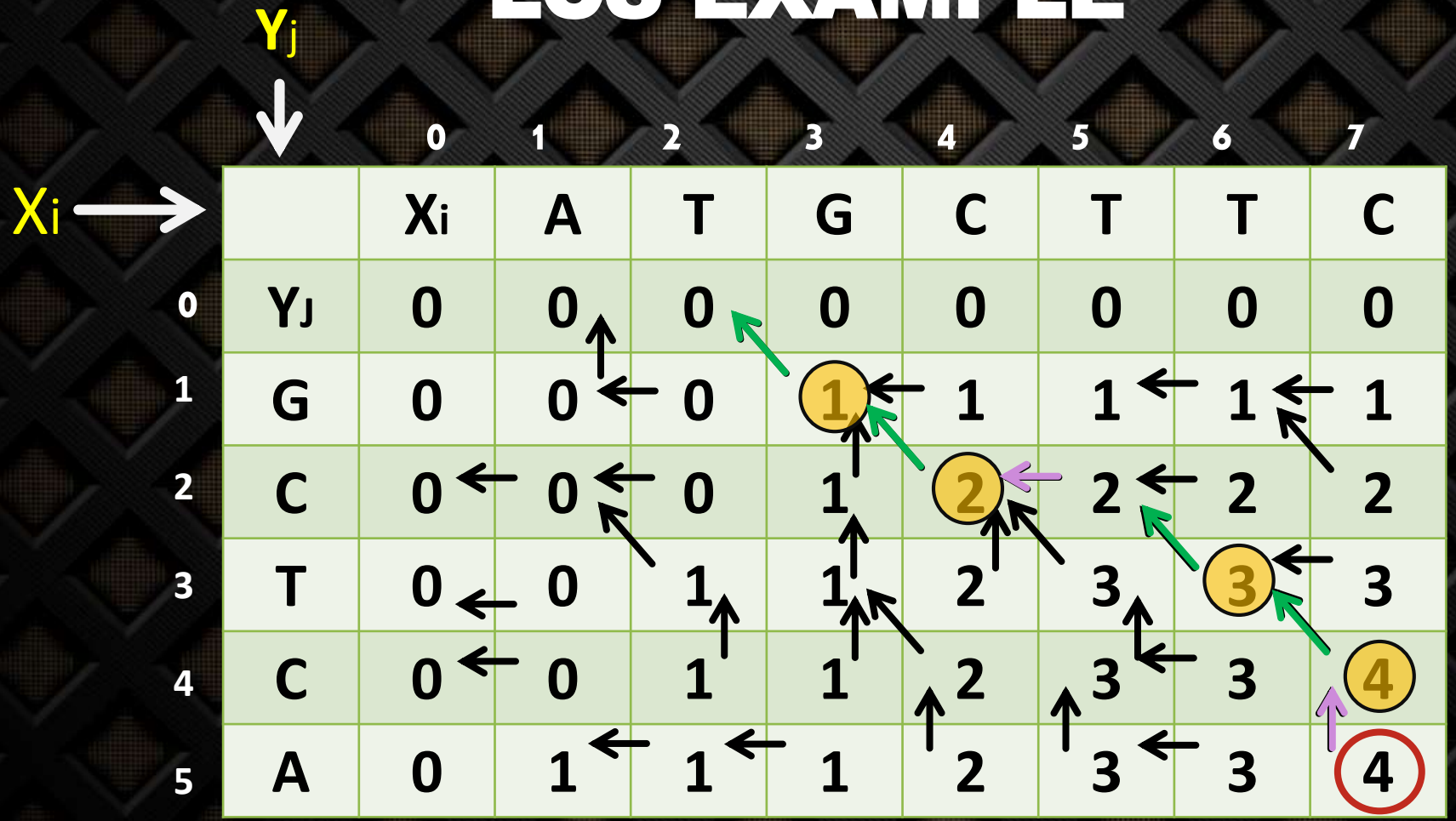


LCS Z= {GCTC}

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}



Firstly have to point out highest value

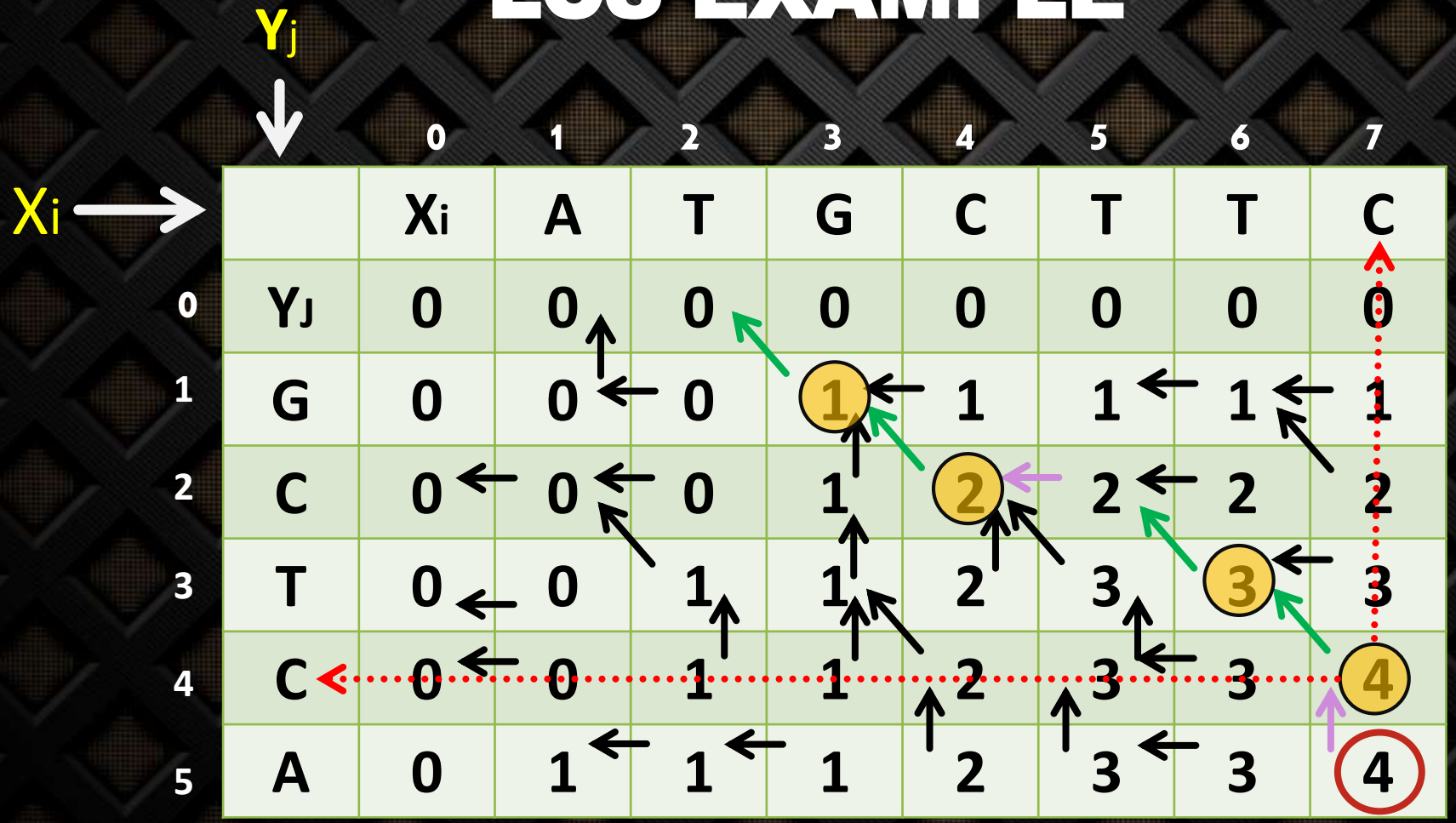
For left and upper arrow we will follow the direction

For diagonal arrow we will point out the character for this cell.

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

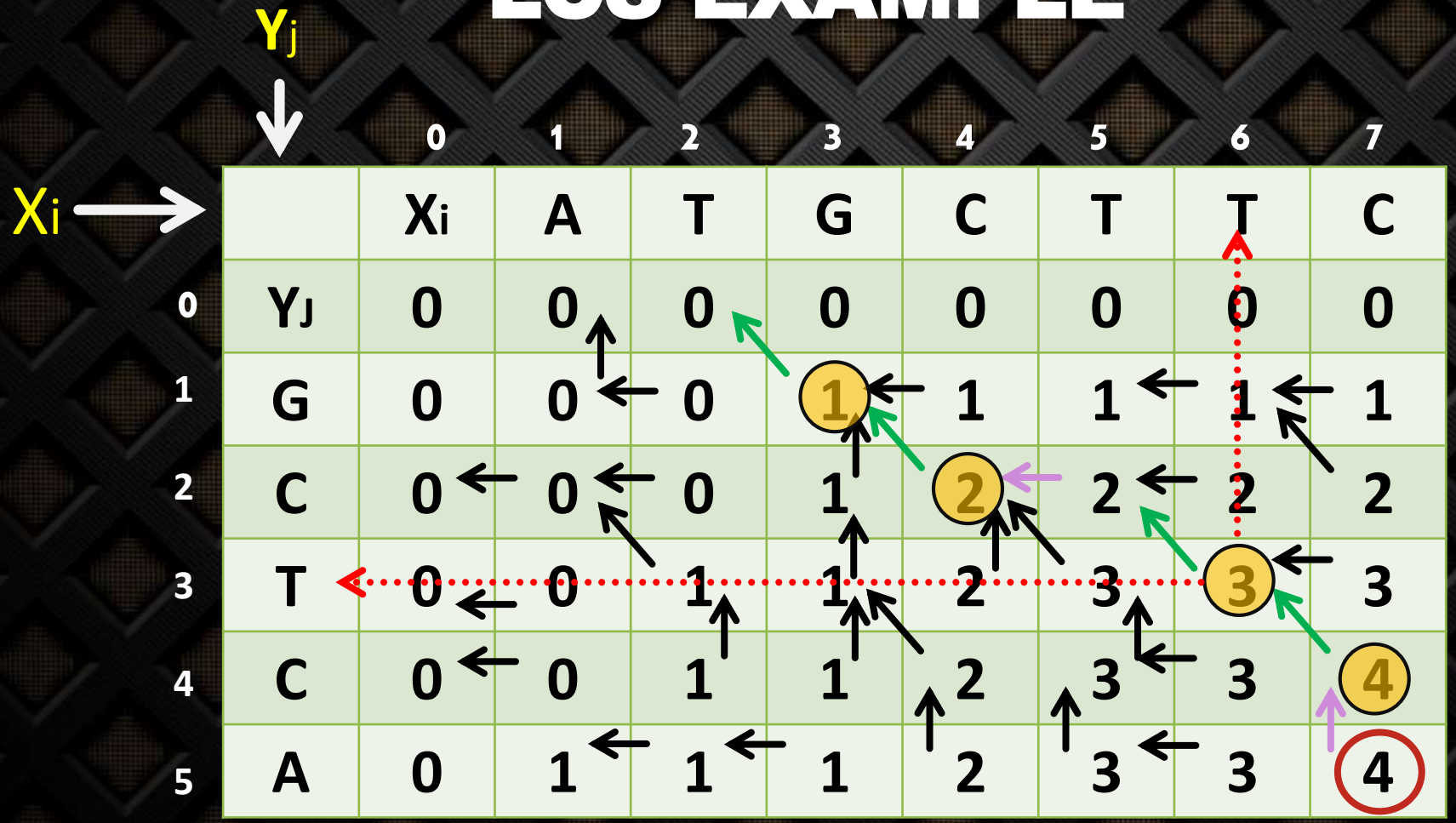


LCS Z= C

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

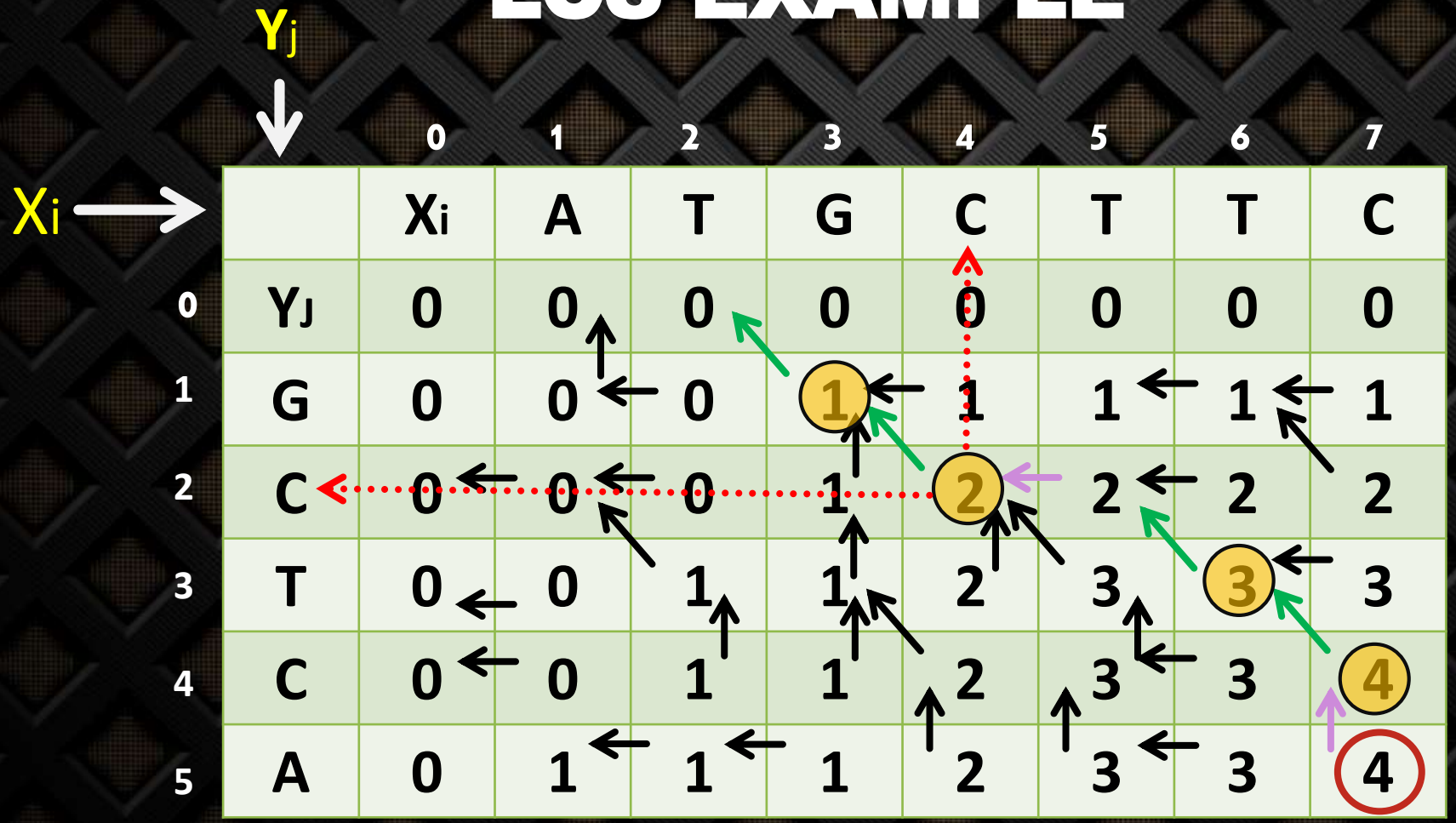


LCS Z= TC

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}

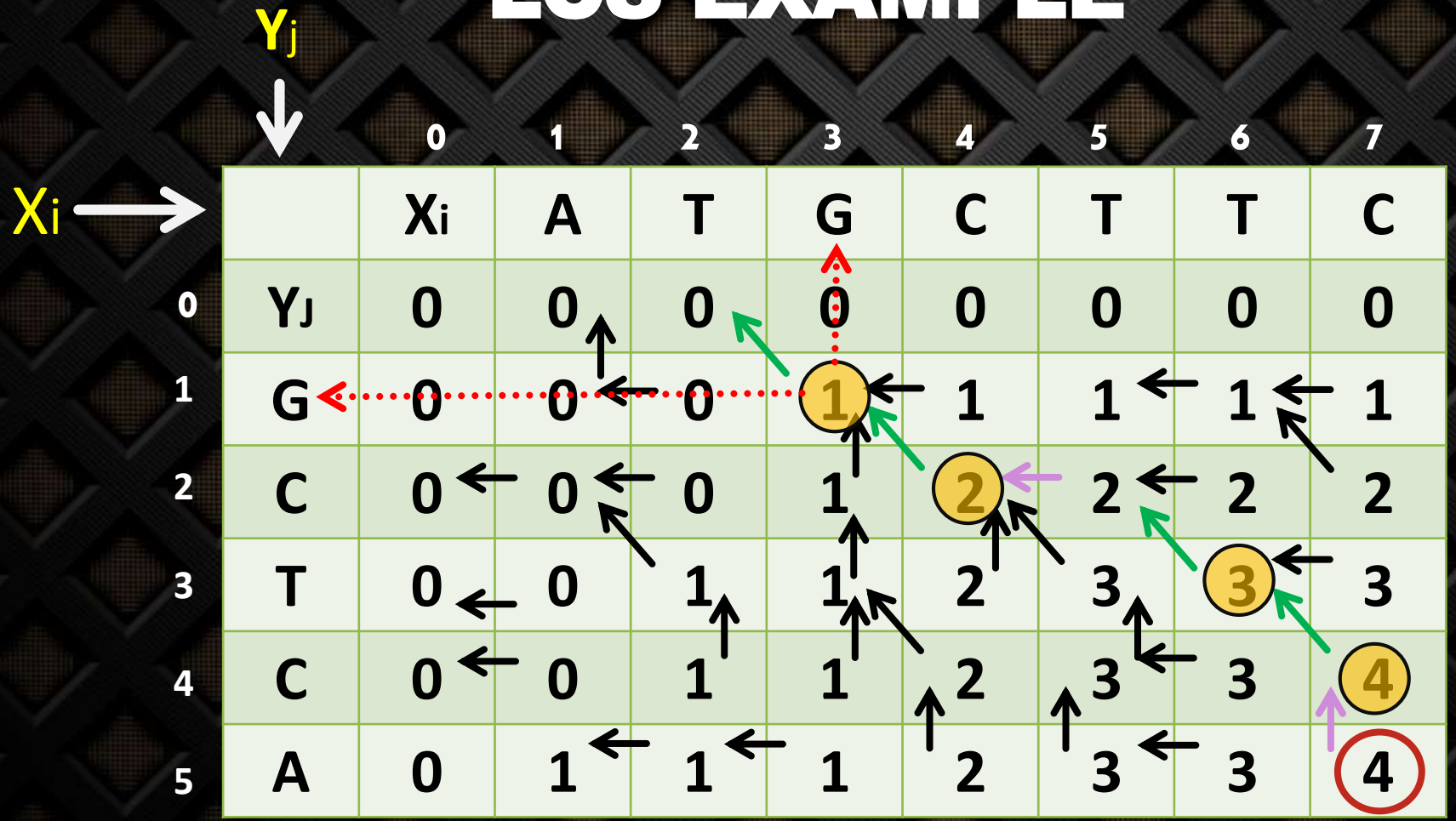


LCS Z= CTC

LCS EXAMPLE

X = {ATGCTTC}

Y = {GCTCA}



LCS Z= {GCTC}