Binary Codes

Gray Code

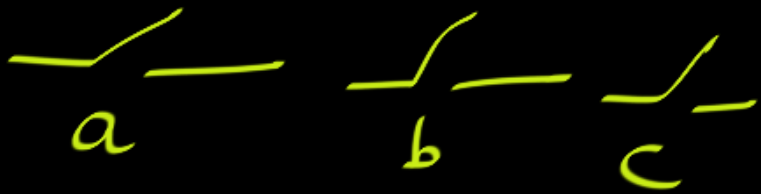
Many Applications ⇒ Counter

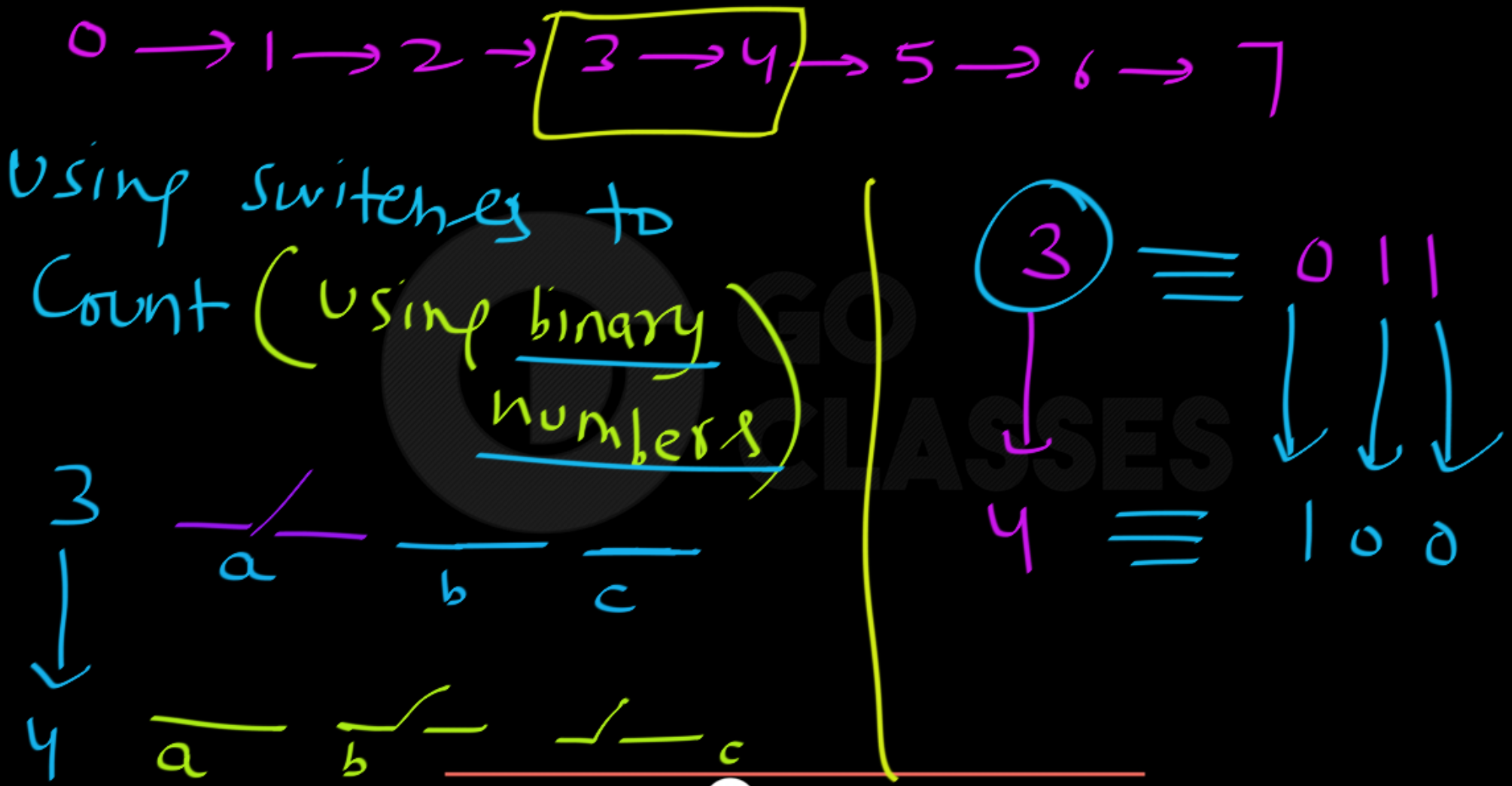
0 → 1 → 2 → 3 → 4 → 5 → 6 → 7

Use Switches to Count

Open = 0
Closed = 1

3 Switches



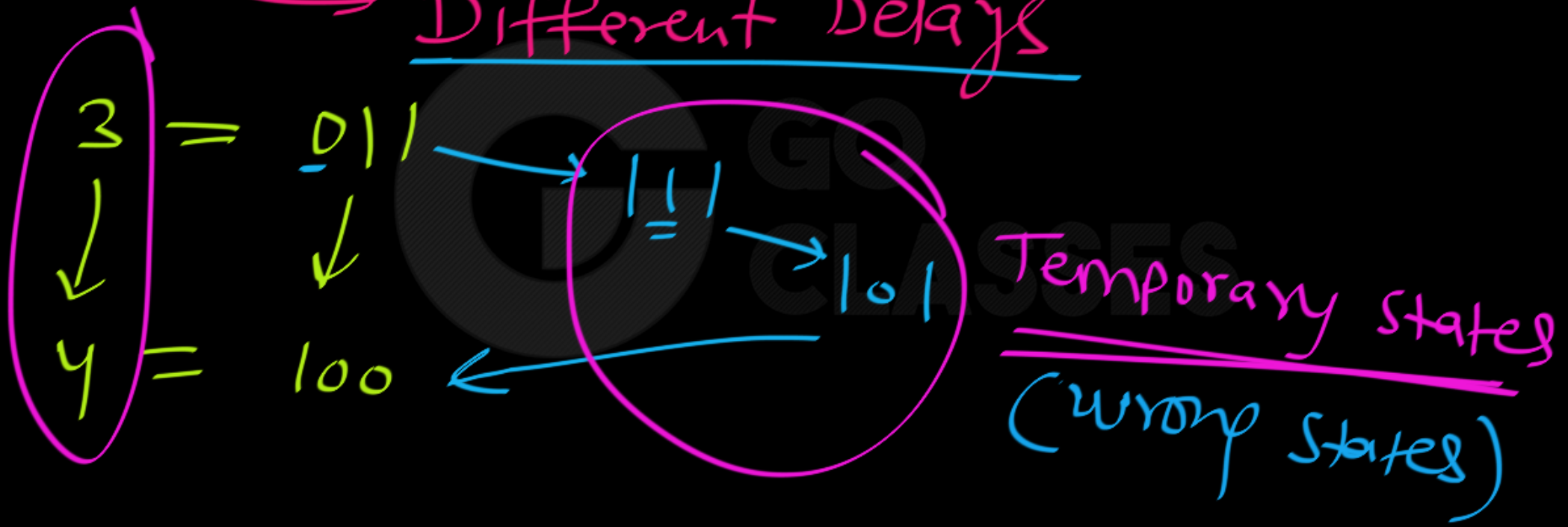

0 → 1 → 2 → 3 → 4 → 5 → 6 → 7
Using switches to Count (using binary numbers)
3
a
b
c
4
a
b
c
3 ≡ 0 1 1
4 ≡ 1 0 0

3 ⟶ 4

If we use Binary Numbers then all three Switches/bit needs to be Changed.

Switches → physical Devices

→ Different Delays

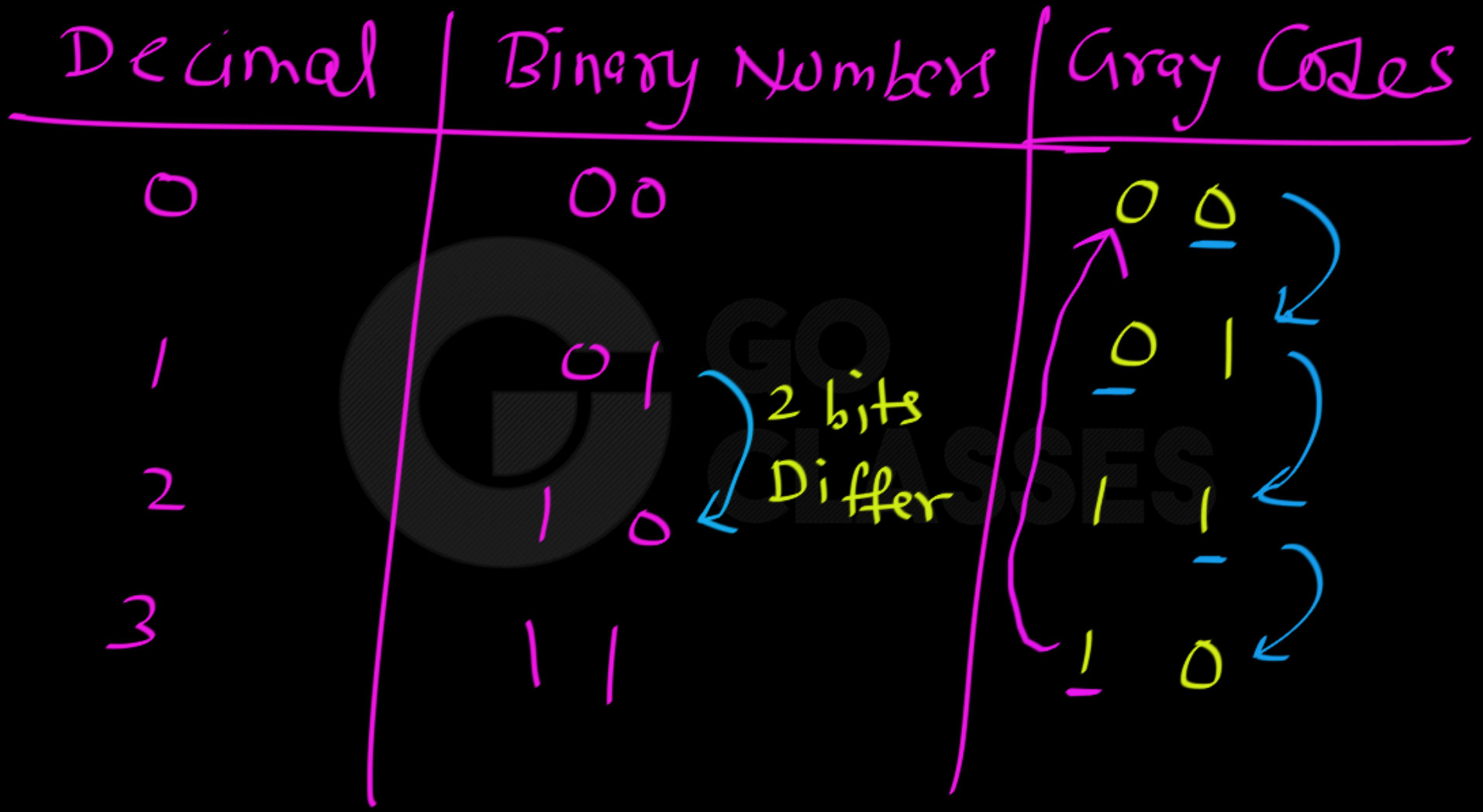


Solution:

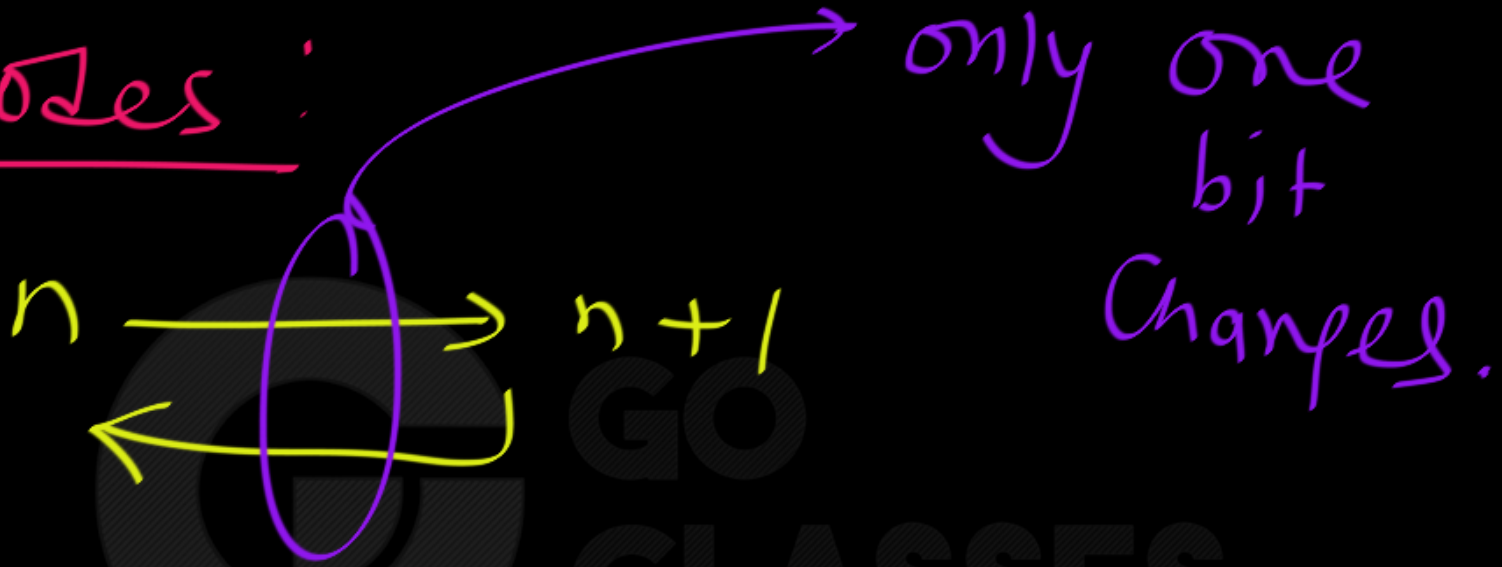
$n \longrightarrow n+1$

only one bit / switch changes.

use Gray Codes


Decimal
Binary Numbers
Gray Codes
0
1
2
3
00
01
10
11
2 bits Differ
0 0
0 1
1 1
1 0

Gray Codes :



The **reflected binary code** (**RBC**), also known as **reflected binary** (**RB**) or **Gray code** after Frank Gray, is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).

For example, the representation of the decimal value "1" in binary would normally be "`001`" and "2" would be "`010`". In Gray code, these values are represented as "`001`" and "`011`". That way, incrementing a value from 1 to 2 requires only one bit to change, instead of two.

advantage of the Gray code over the straight binary number sequence is that only one bit in the code group changes in going from one number to the next. For example, in going from 7 to 8, the Gray code changes from 0100 to 1100. Only the first bit changes, from 0 to 1; the other three bits remain the same. By contrast, with binary numbers the change from 7 to 8 will be from 0111 to 1000, which causes all four bits to change values.

Motivation and name [edit]

Many devices indicate position by closing and opening switches. If that device uses natural binary codes, positions 3 and 4 are next to each other but all three bits of the binary representation differ:

Decimal	Binary
...	...
3	011
4	100
...	...

The problem with natural binary codes is that physical switches are not ideal: it is very unlikely that physical switches will change states exactly in synchrony. In the transition between the two states shown above, all three switches change state. In the brief period while all are changing, the switches will read some spurious position. Even without keybounce, the transition might look like 011 — 001 — 101 — 100. When the switches appear to be in position 001, the observer cannot tell if that is the "real" position 1, or a transitional state between two other positions. If the output feeds into a sequential system, possibly via combinational logic, then the sequential system may store a false value.

Table 1.6
Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Q: Decimal $n = 17$

Binary Encoding / Binary Number : 1 0 0 0 1

→ Gray Code ?

Objective:

Decimal n ⟶ Gray Code of n !

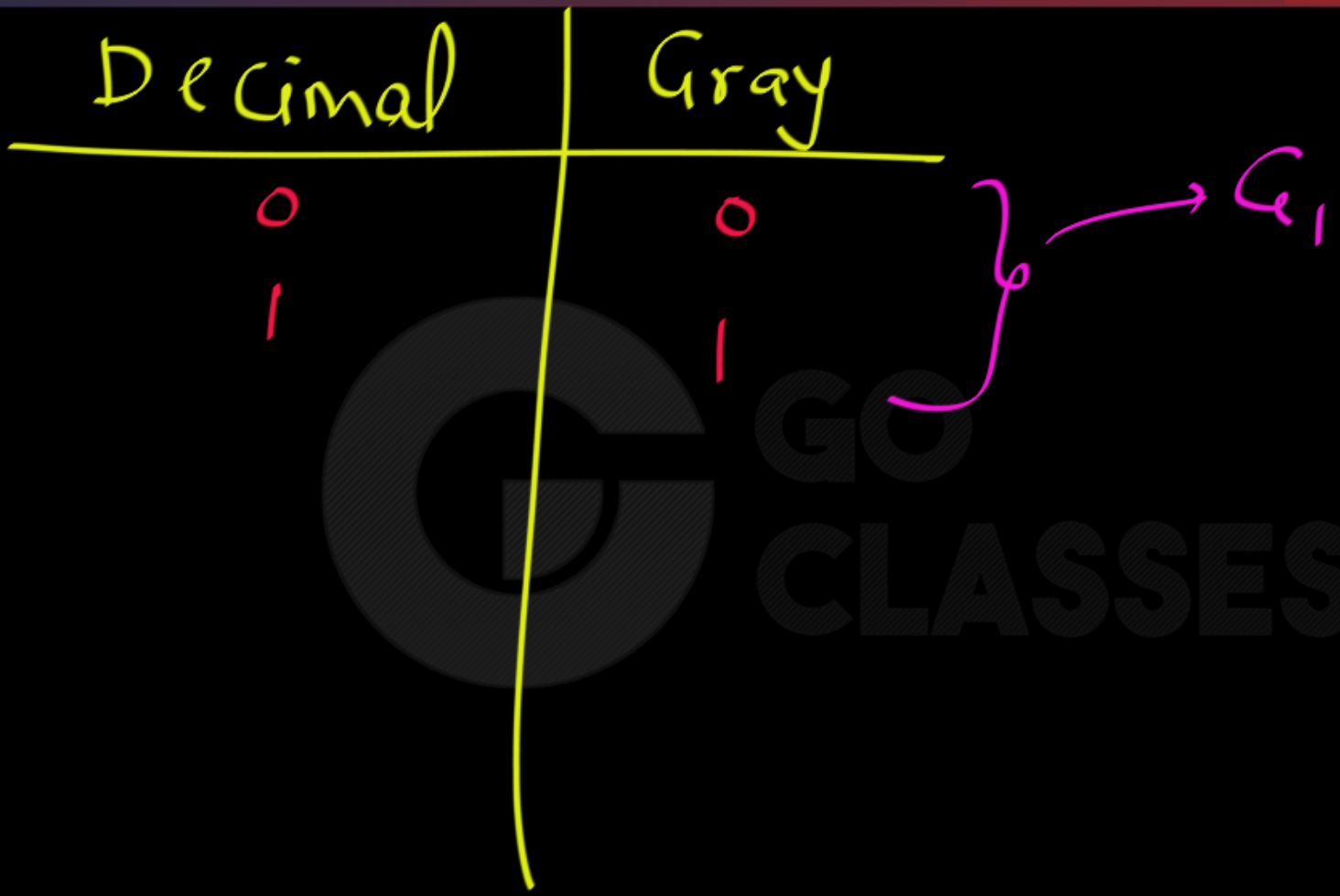
Gray Codes

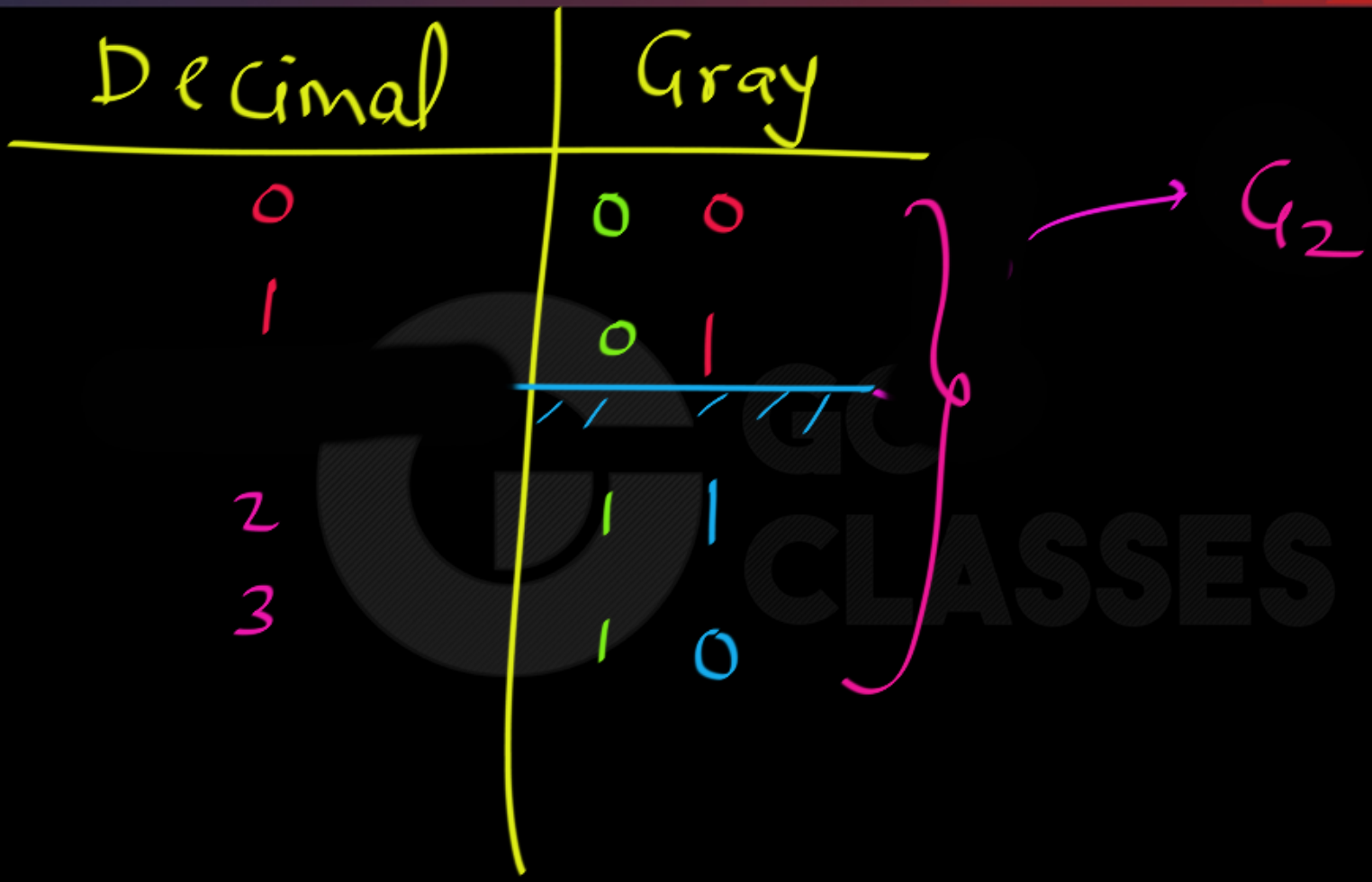
Recursive Construction Of Gray Codes

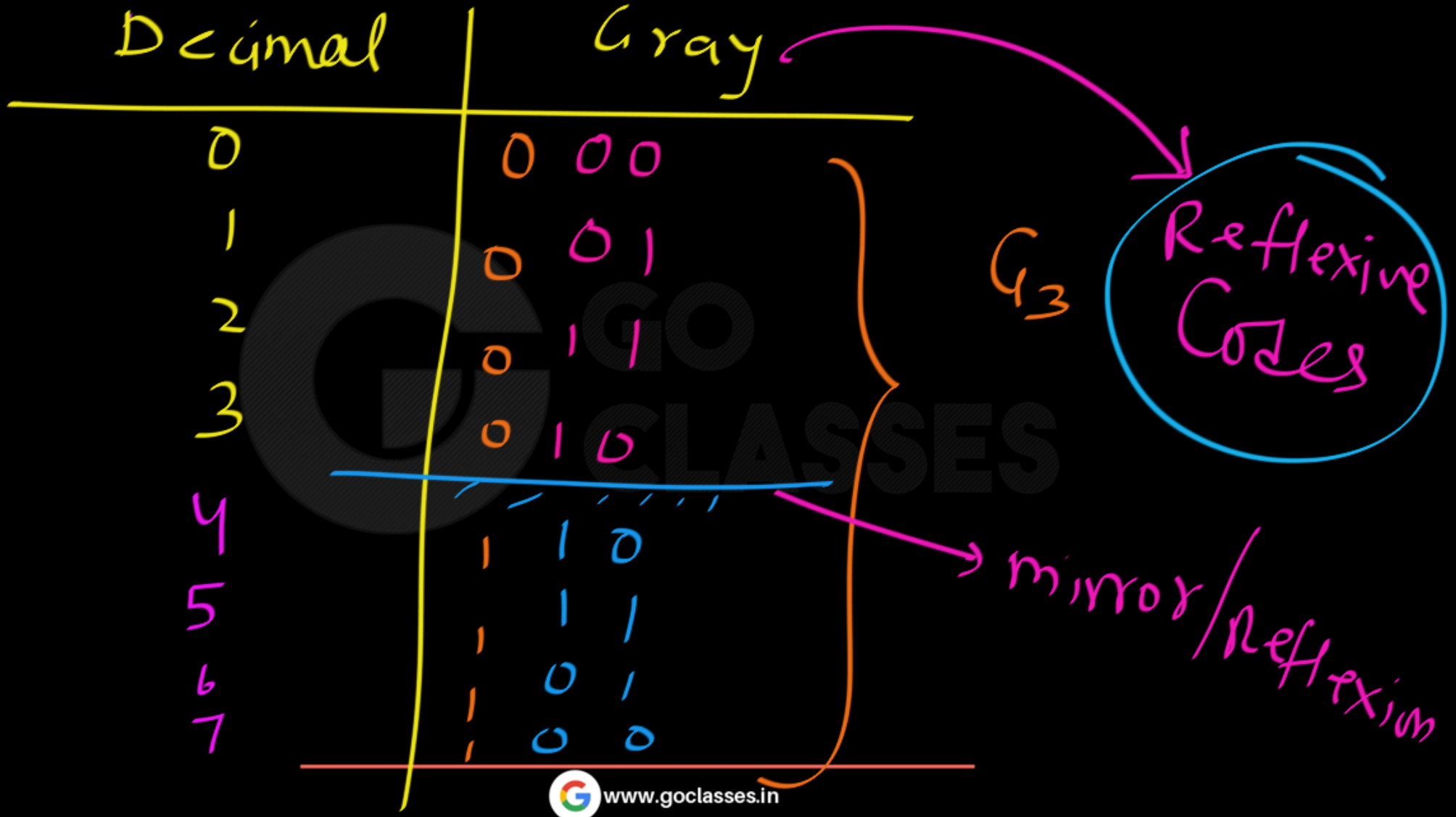
G_n : Gray Code with n bits.

G_1 : 0, 1

G_2 : 00, 01, 11, 10



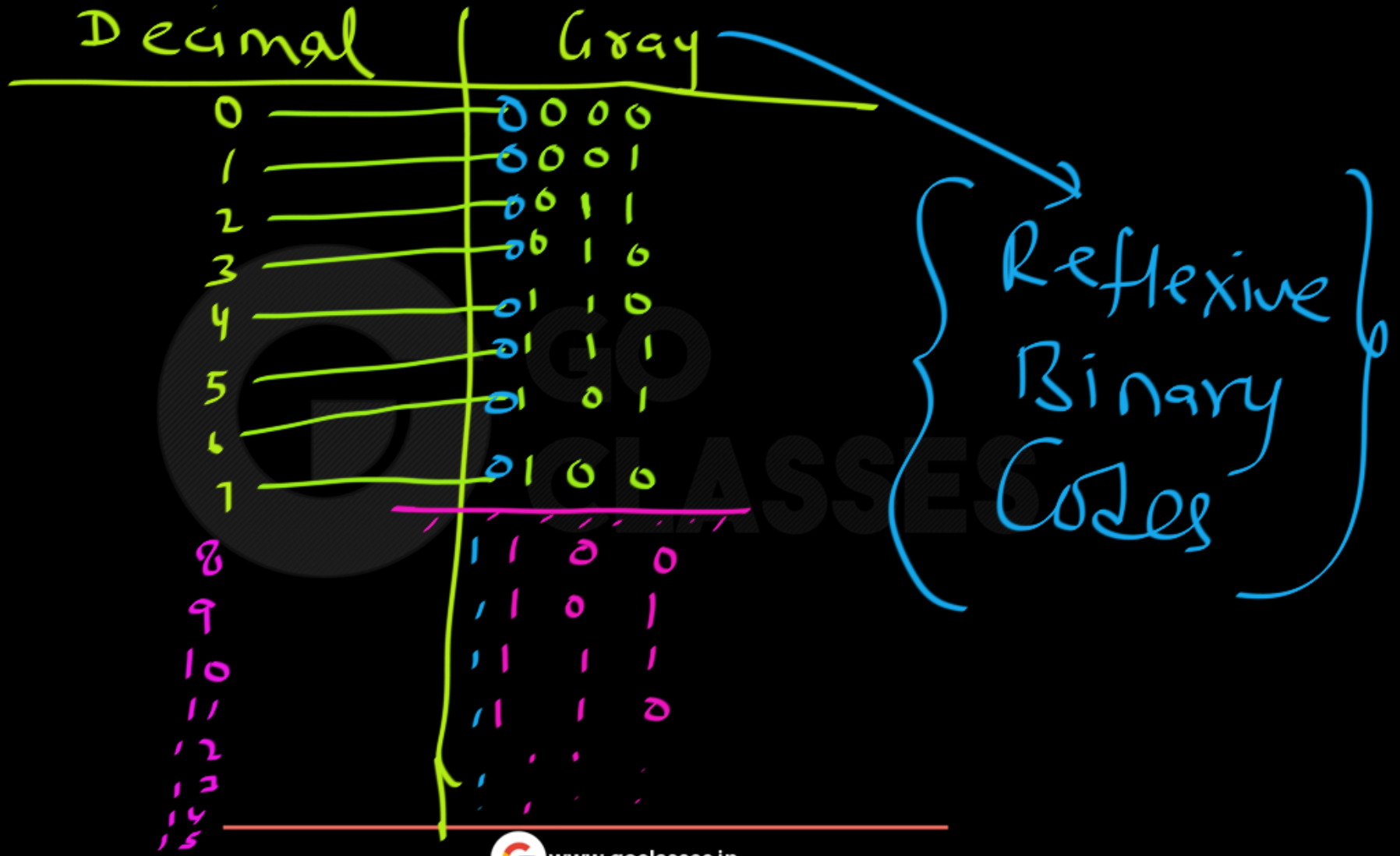

Decimal
Gray
0
1
2
3
0 0
0 1
1 1
1 0
G_2


Decimal
Gray
0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0
0
1
2
3
4
5
6
7
G_3
Reflexive Codes
mirror/Reflexive

G_2 ⟶ G_3

00, 01, 11, 10

0 00, 0 01, 0 11, 0 10, 1 10, 1 11, 1 01, 1 00

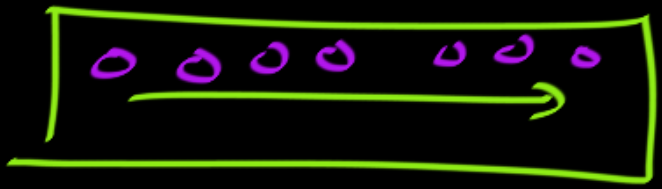

Decimal
Gray
0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
Reflexive
Binary
Codes

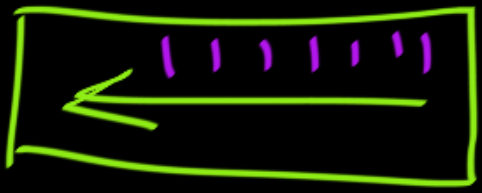
G_1 → G_2

0, 1

00, 01, 11, 10

G_n → G_{n+1}





Constructing an n-bit Gray code [edit]

The binary-reflected Gray code list for n bits can be generated recursively from the list for $n - 1$ bits by reflecting the list (i.e. listing the entries in reverse order), prefixing the entries in the original list with a binary `0`, prefixing the entries in the reflected list with a binary `1`, and then concatenating the original list with the reversed list.[12] For example, generating the $n = 3$ list from the $n = 2$ list:

2-bit list:	`00, 01, 11, 10`	
Reflected:		`10, 11, 01, 00`
Prefix old entries with `0`:	`000, 001, 011, 010,`	
Prefix new entries with `1`:		`110, 111, 101, 100`
Concatenated:	`000, 001, 011, 010,`	`110, 111, 101, 100`

Gray Codes Construction

Gray codes of any number of bits can be reconstructed recursively using the three rules:

- A 1-bit gray code has two codes 0 and 1
- The first 2^{n-1} code words of an n-bit gray code equal the code words of (n-1)-bit gray code, written in order with a leading zero appended.
- The last 2^{n-1} code words of an n-bit gray code equal the code words of (n-1)-bit gray code, but written in reverse order with a leading 1 appended.

Gray Codes ≡ Reflexive Binary Codes

Gray Code of m = 29

efficient method

binary of 29 → Gray of 29

Recursive method is time Consuming.

Gray Codes

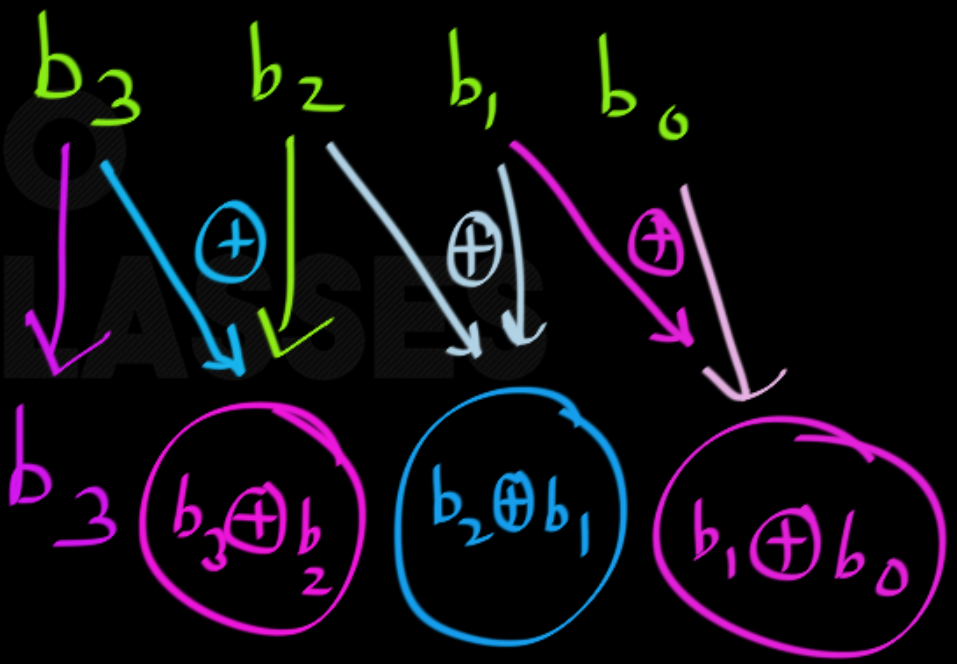
Binary to Gray Code

Decimal : m ⟶ Gray Code of m ?

Binary of m ?

↓

Gray of m ?



$n = 29$ ⟶ Gray of 29 ?

↓

binary : 1 1 1 0 1

↓ (Gray code)

Gray code : 1 0 0 1 1

Decimal : 2 ⟶ Gray Code of 2?

binary : 10

Gray : 11

11

binary : $b_3\ b_2\ b_1\ b_0$

Gray : $g_3\ g_2\ g_1\ g_0$

$g_3 = b_3$; $g_2 = b_3 \oplus b_2$; $g_1 = b_2 \oplus b_1$

$g_0 = b_1 \oplus b_0$


b3
b2
b1
b0
g3
g2
g1
g0

find Gray Codes of 0 to 15 ?

Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010

find Gray Codes of 0 to 15 ?

Decimal	Binary	Gray Code
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0

find Gray Codes of 0 to 15 ?

Decimal	Binary	Gray Code
8		
9		H W
10		
11		

find Gray Codes of 0 to 15 ?

Decimal	Binary	Gray Code
12		
13		
14		H W
15		

Gray Codes

Gray to Binary Code

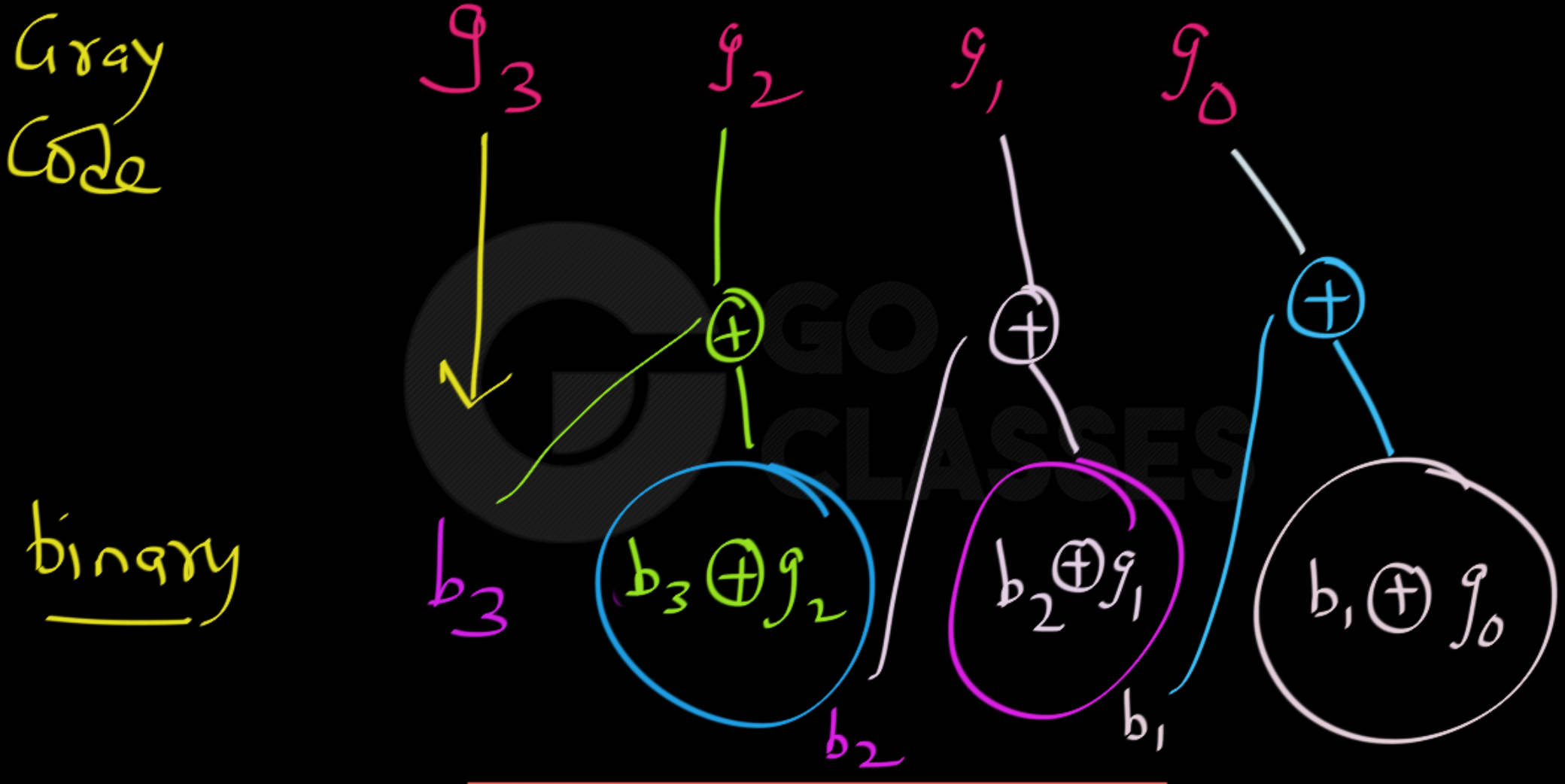
Q: Decimal number m, whose Binary Number is 1101.

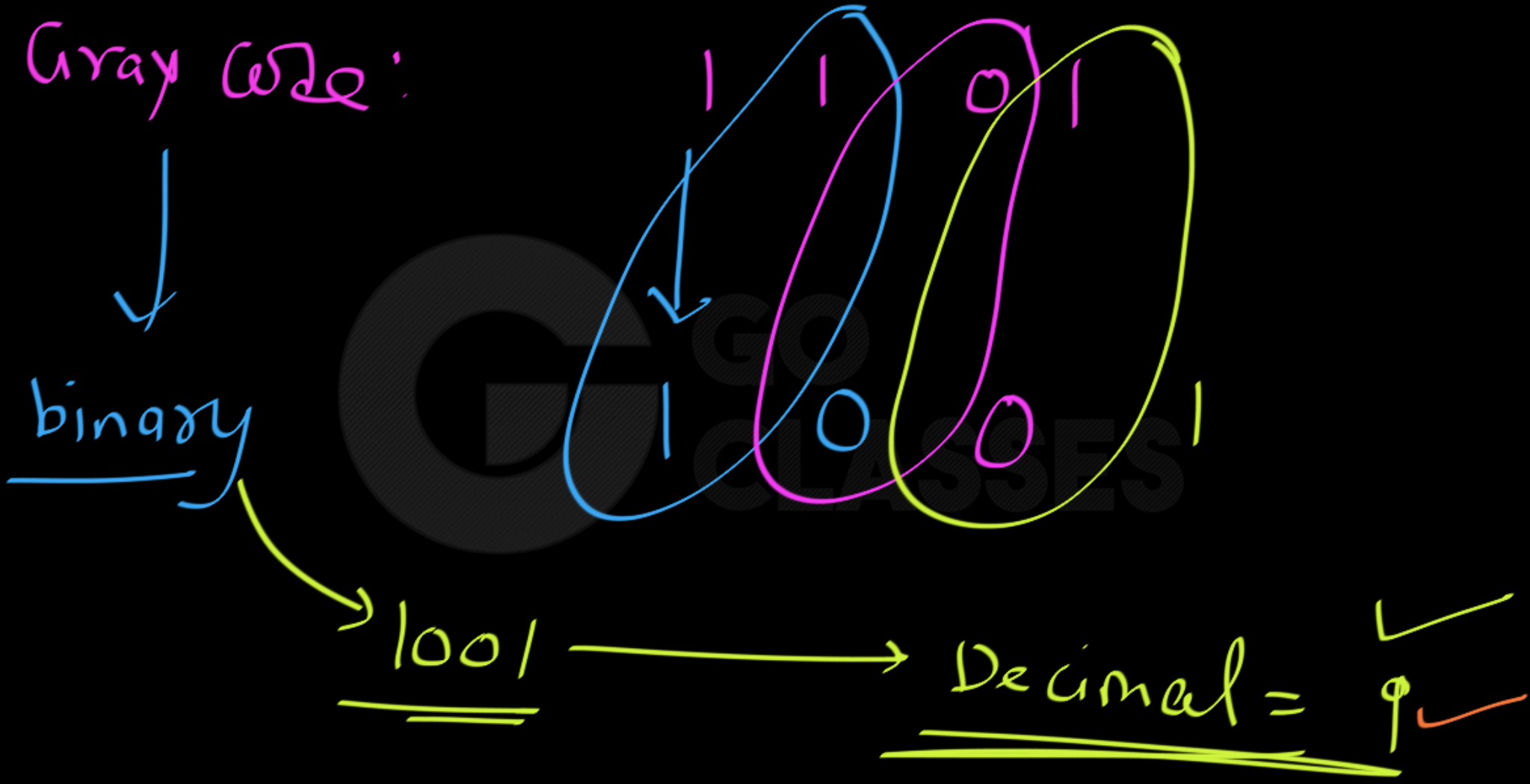
What is m? = 13

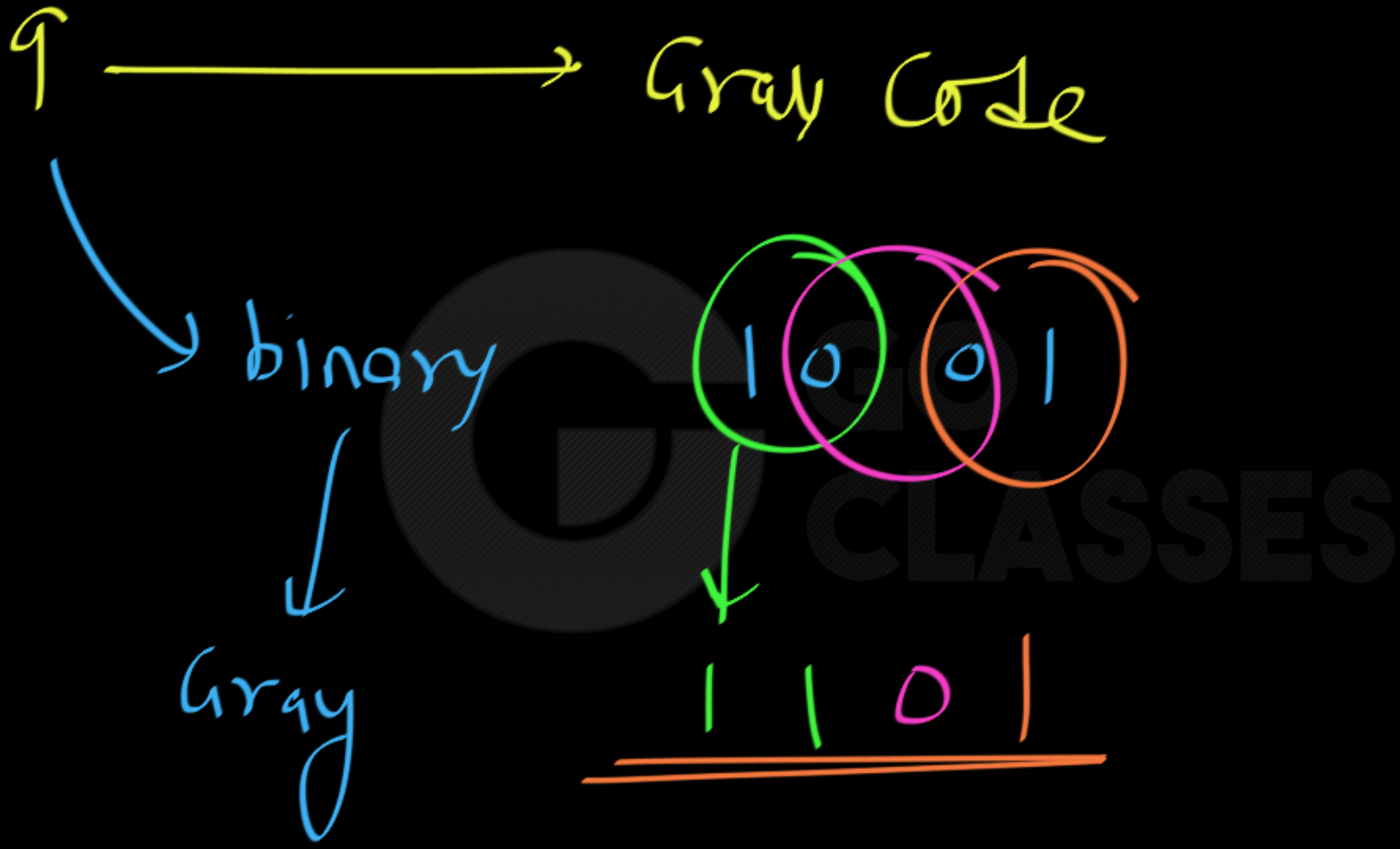
Q: Decimal number m, Who Gray code is 1101.

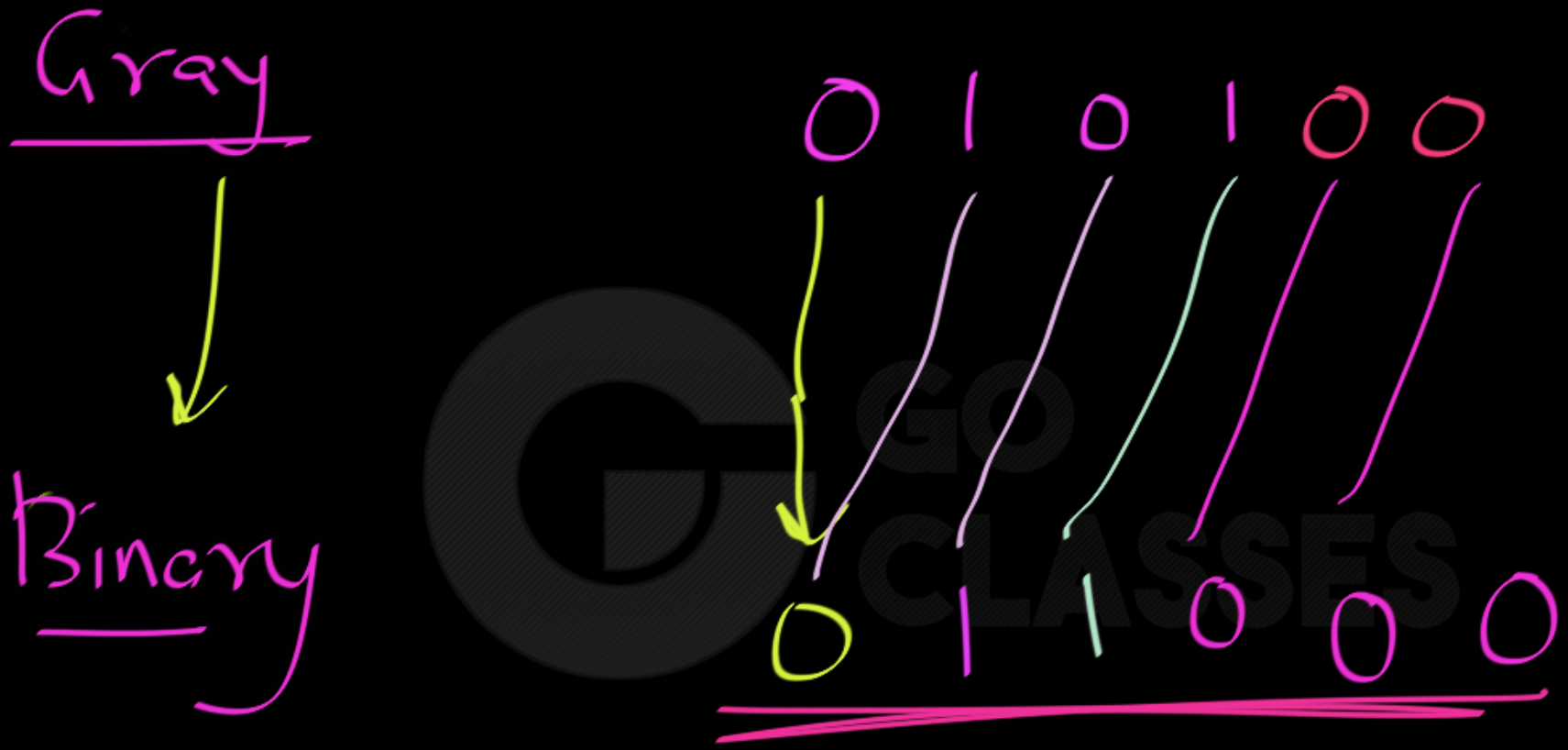
What is m?

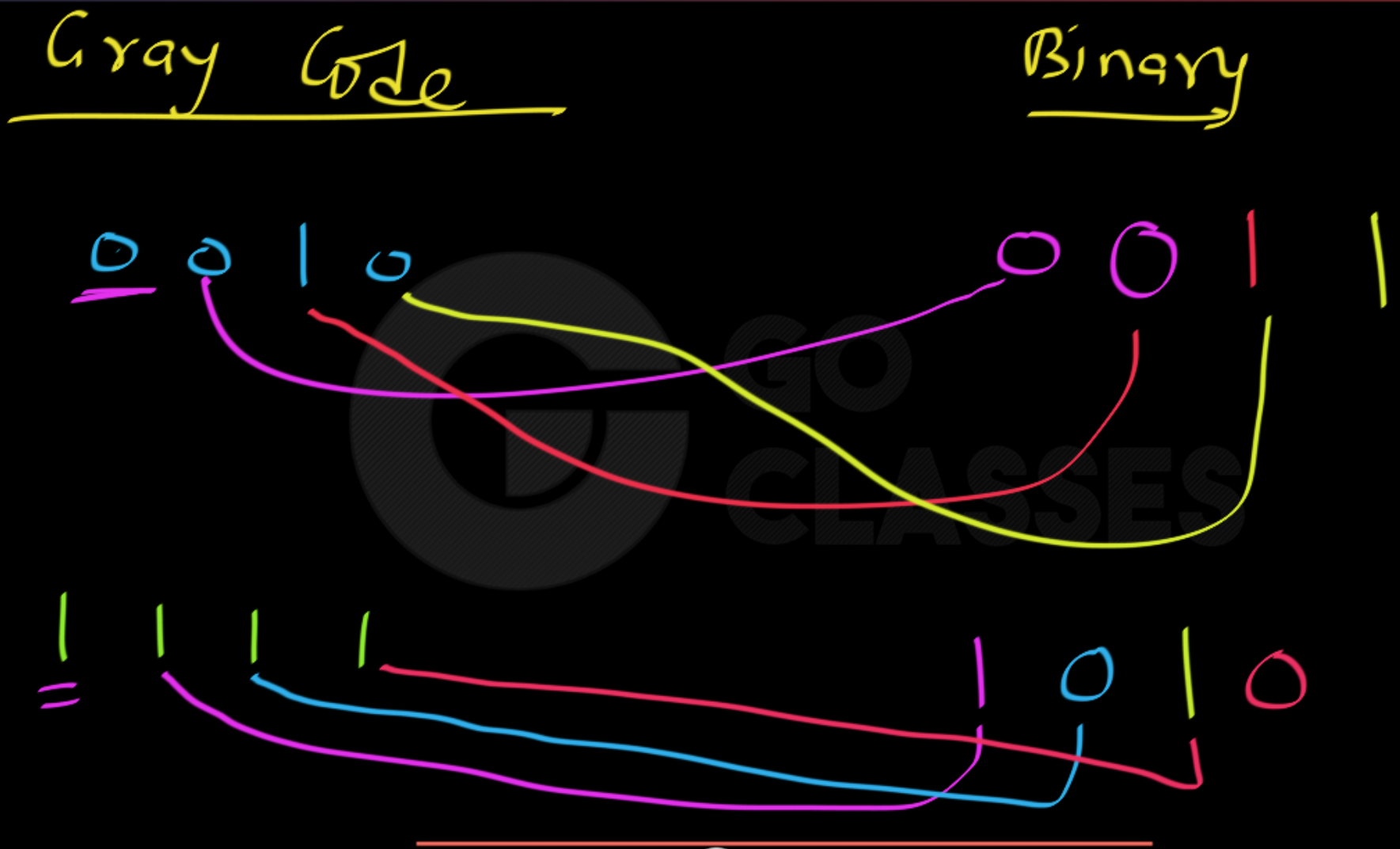
Gray code → binary code → Decimal value

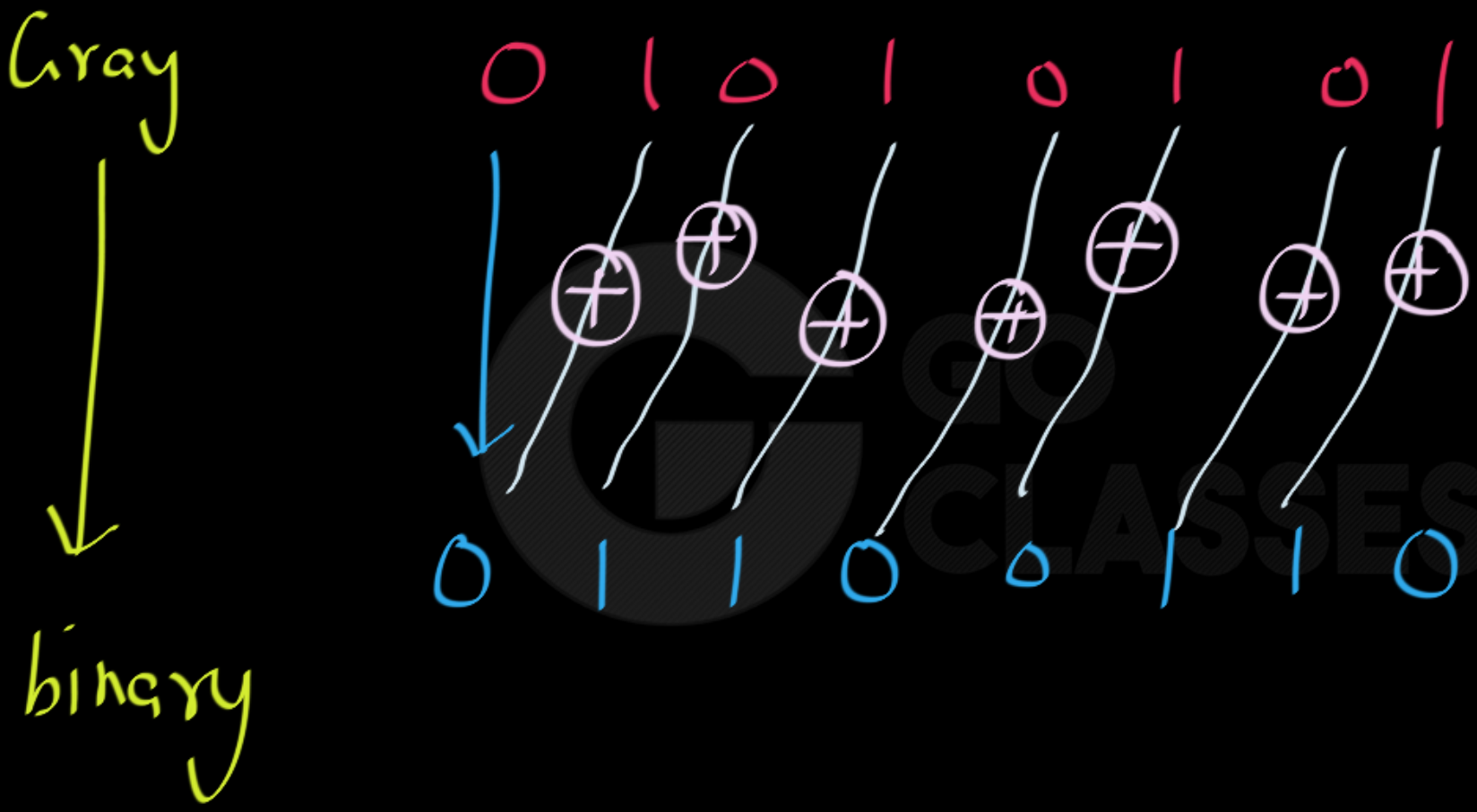

Gray Code
g_3
g_2
g_1
g_0
binary
b_3
$b_3 \oplus g_2$
$b_2 \oplus g_1$
$b_1 \oplus g_0$
b_2
b_1

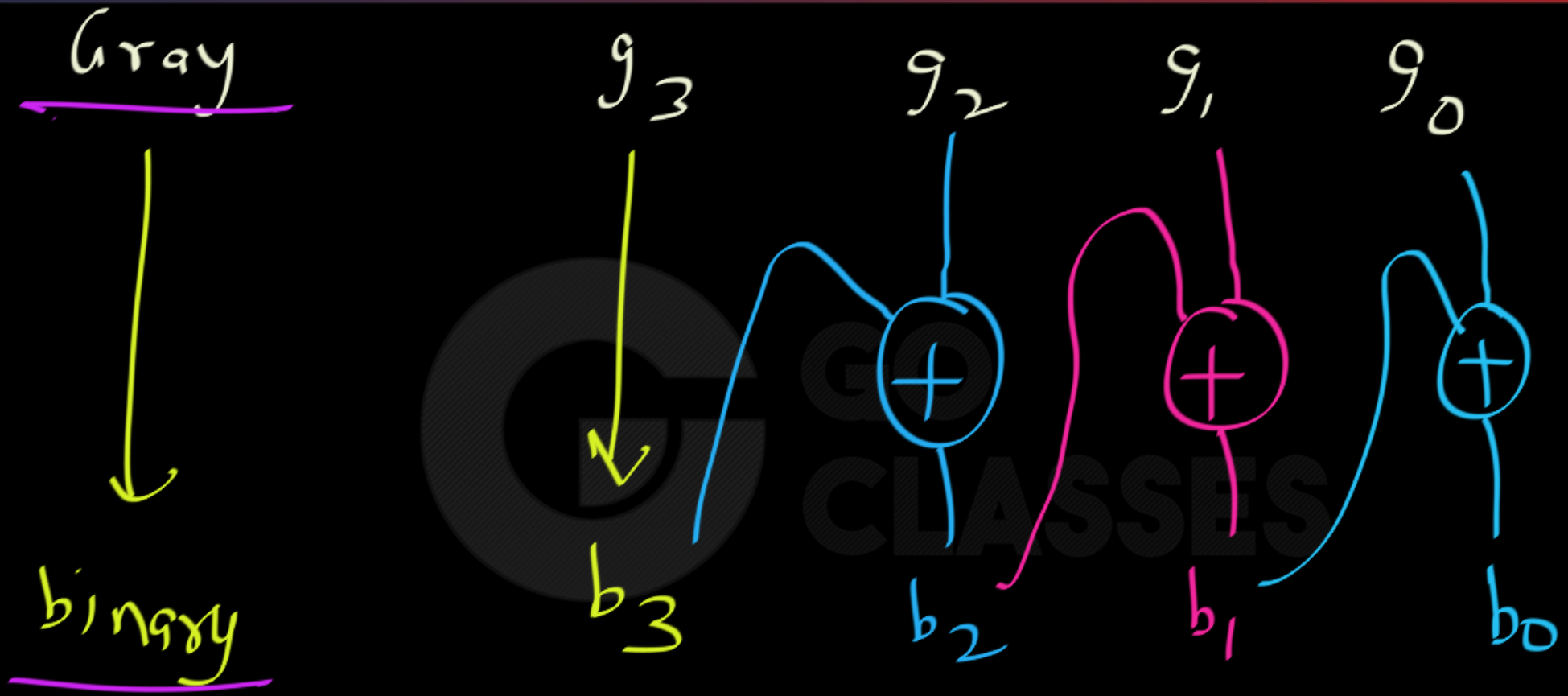

Gray code :
1 1 0 1
binary
1 0 0 1
1001
Decimal = 9


9
Gray Code
binary
1 0 0 1
Gray
1 1 0 1


Gray
Binary
0 1 0 1 0 0
0 1 1 0 0 0


Gray Code
Binary
0 0 1 0
0 0 1 1
1 1 1 1
1 0 1 0

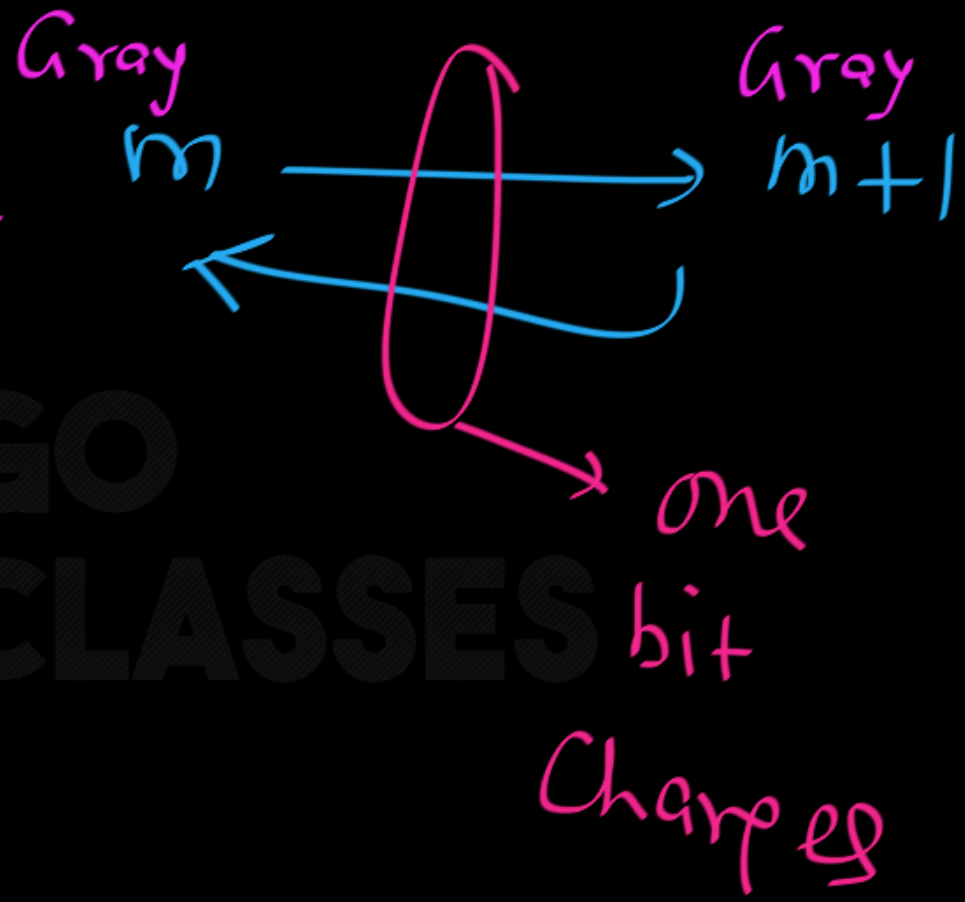

Gray
0 1 0 1 0 1 0 1
binary
0 1 1 0 0 1 1 0


Gray
binary
g_3
g_2
g_1
g_0
b_3
b_2
b_1
b_0

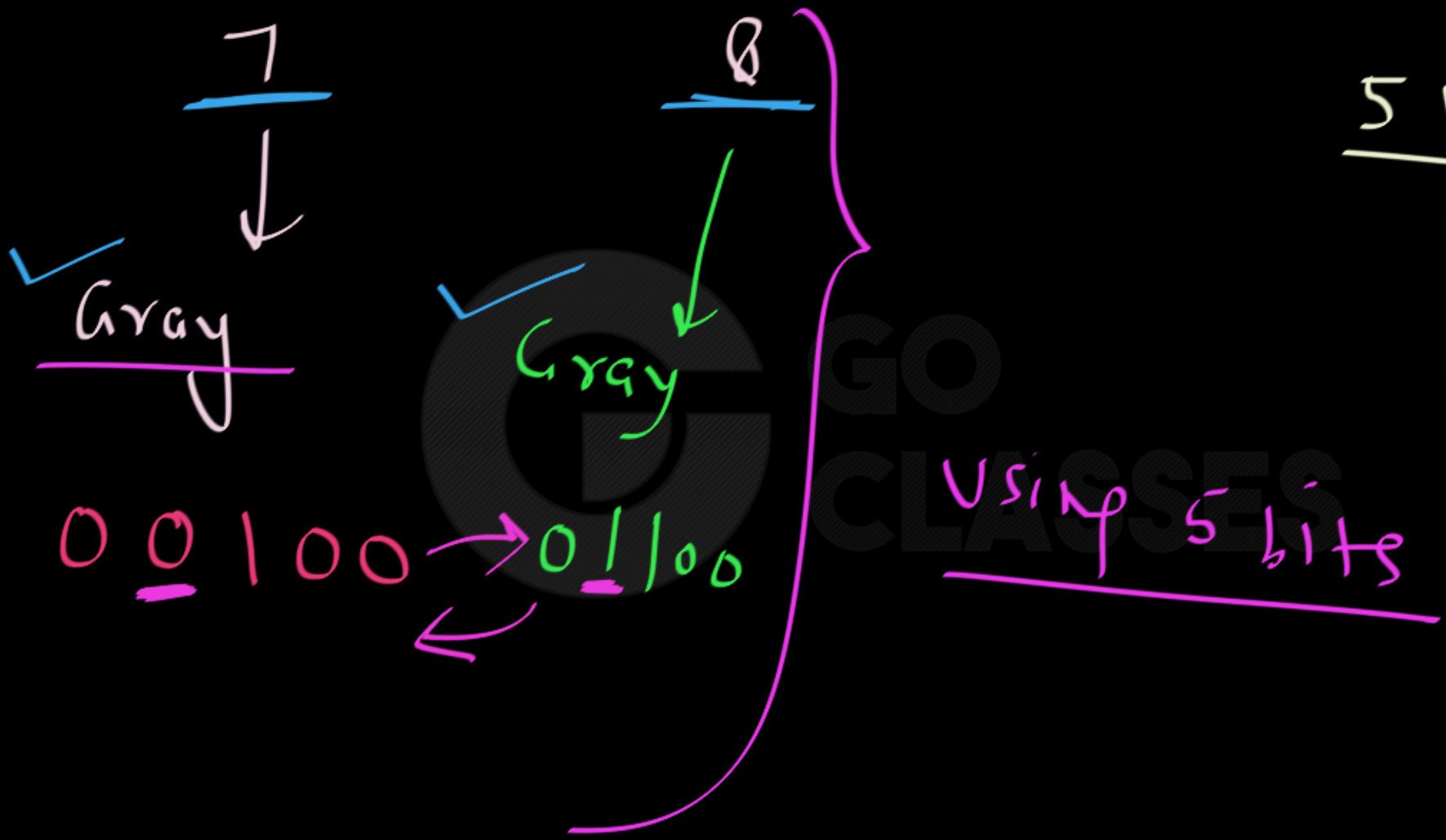

binary
0 1 0 1 0 1
Gray
0 1 1 1 1 1

Note:

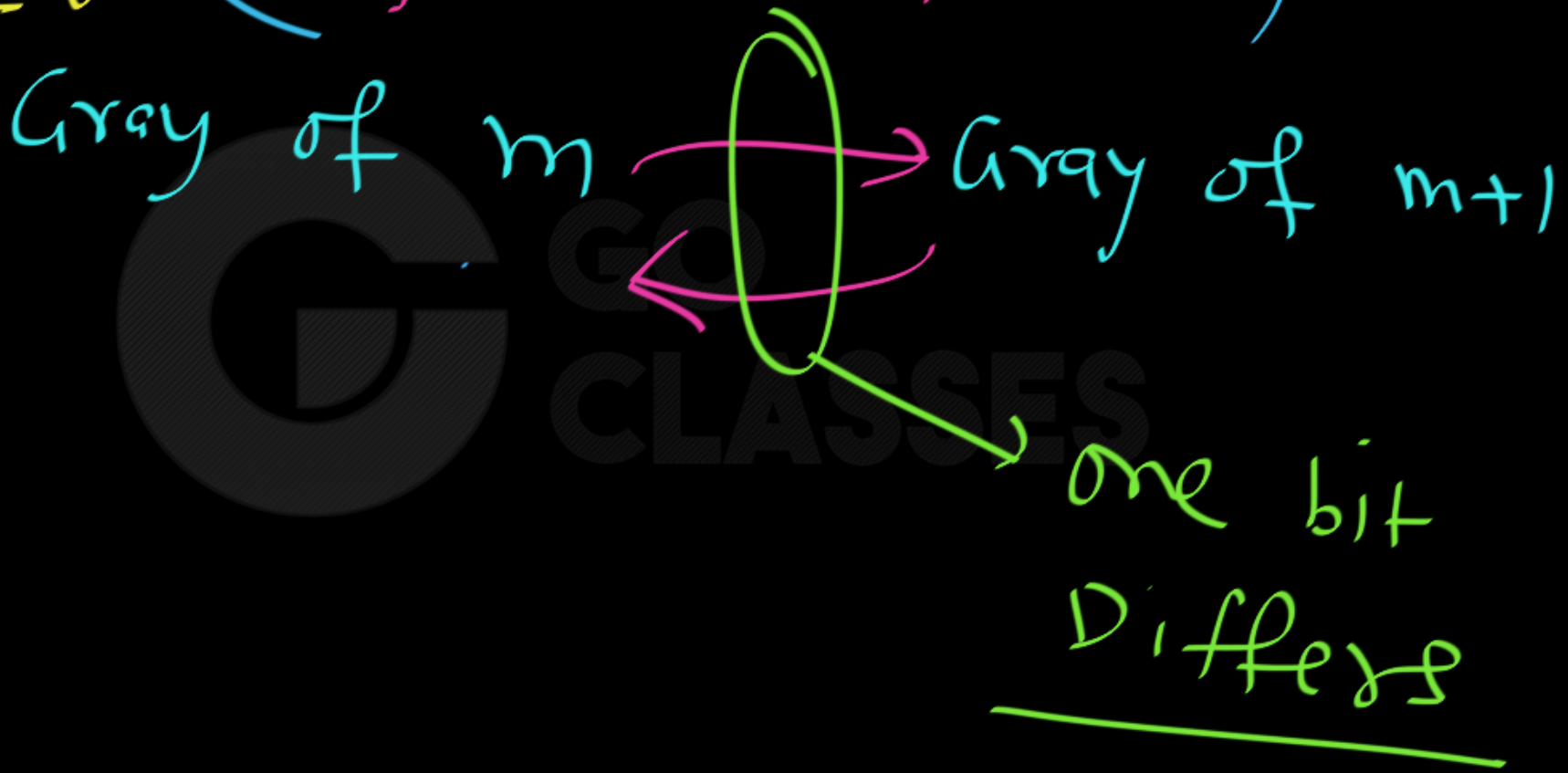
Decimal




7
Gray
0100
1100
8
Gray
15
Gray
01000
16
Gray
11000


7
8
5 bits
Gray
Gray
00100 → 01100
Using 5 bits

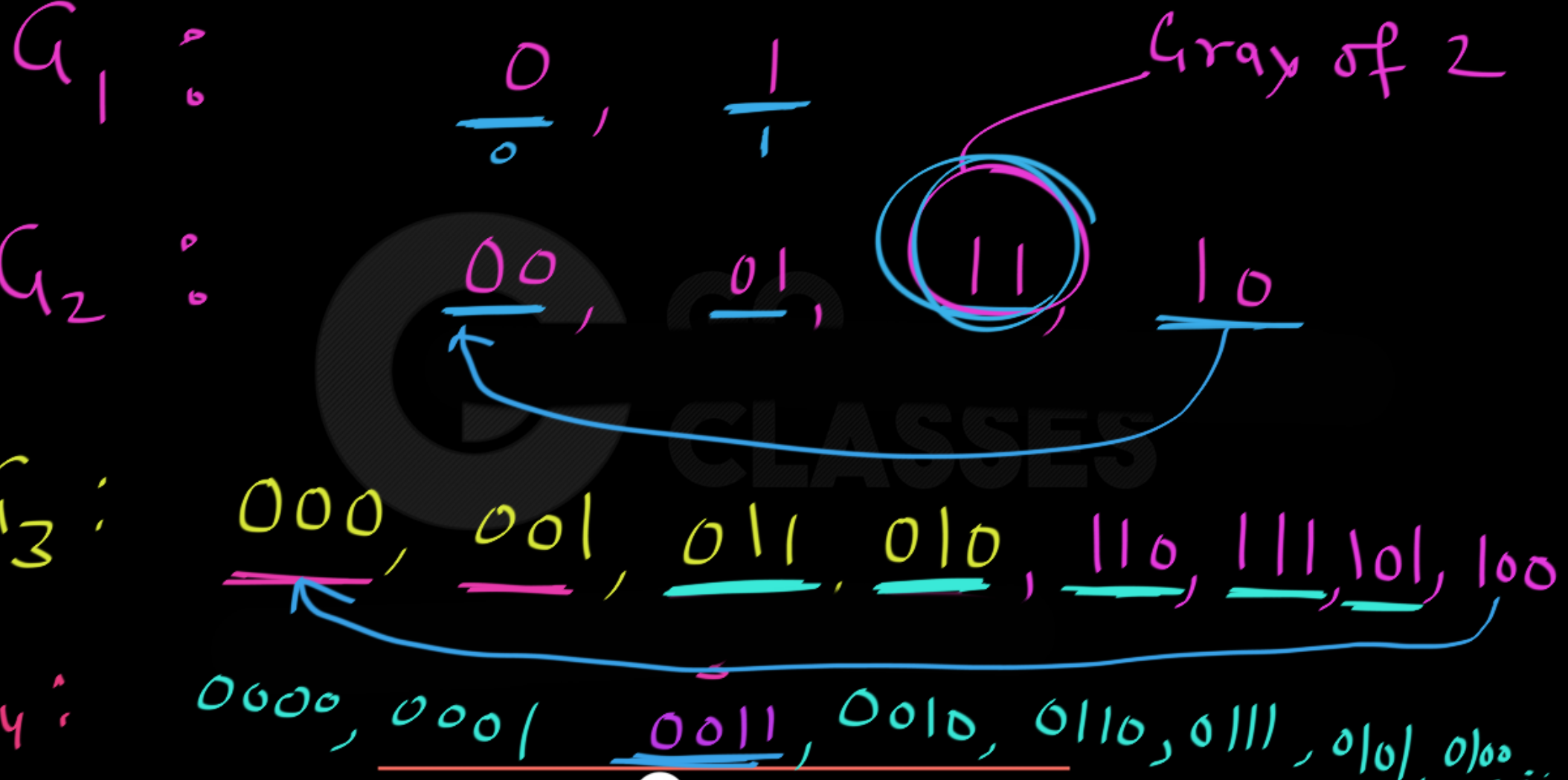
Property: (Any number of bits)



Gray Codes

Properties of G_n

(Gray Codes of length n)


G_1 : 0, 1
Gray of 2
G_2 : 00, 01, 11, 10
G_3 : 000, 001, 011, 010, 110, 111, 101, 100
G_4 : 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100 ...

(1) Gray Codes are Reflexive Codes

(2) In G_n j Gray Codes are Cyclic Codes.

G_n :

G_2 : 00, 01, 11, 10

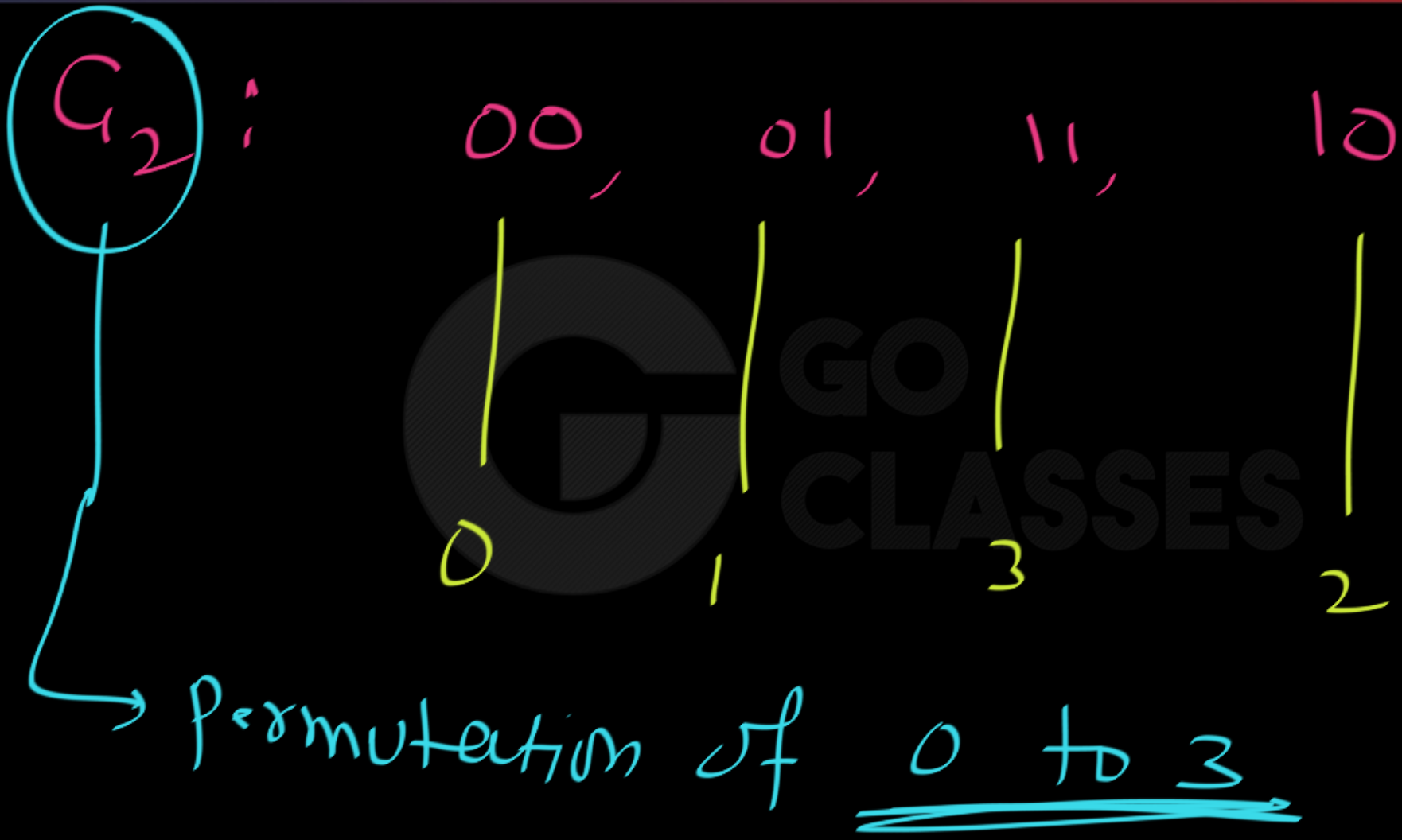
k-map :

G_2 : 00, 01, 11, 10

Permutation of all bit-strings of 2 bits.

G_3: 000, 001, 011, 010, 110, 111, 101, 100

Permutation of All 3-bit bit strings.


G_2 :
00, 01, 11, 10
0 1 3 2
Permutation of 0 to 3

G_1 is embedded as first half of G_2.

G_1 G_2

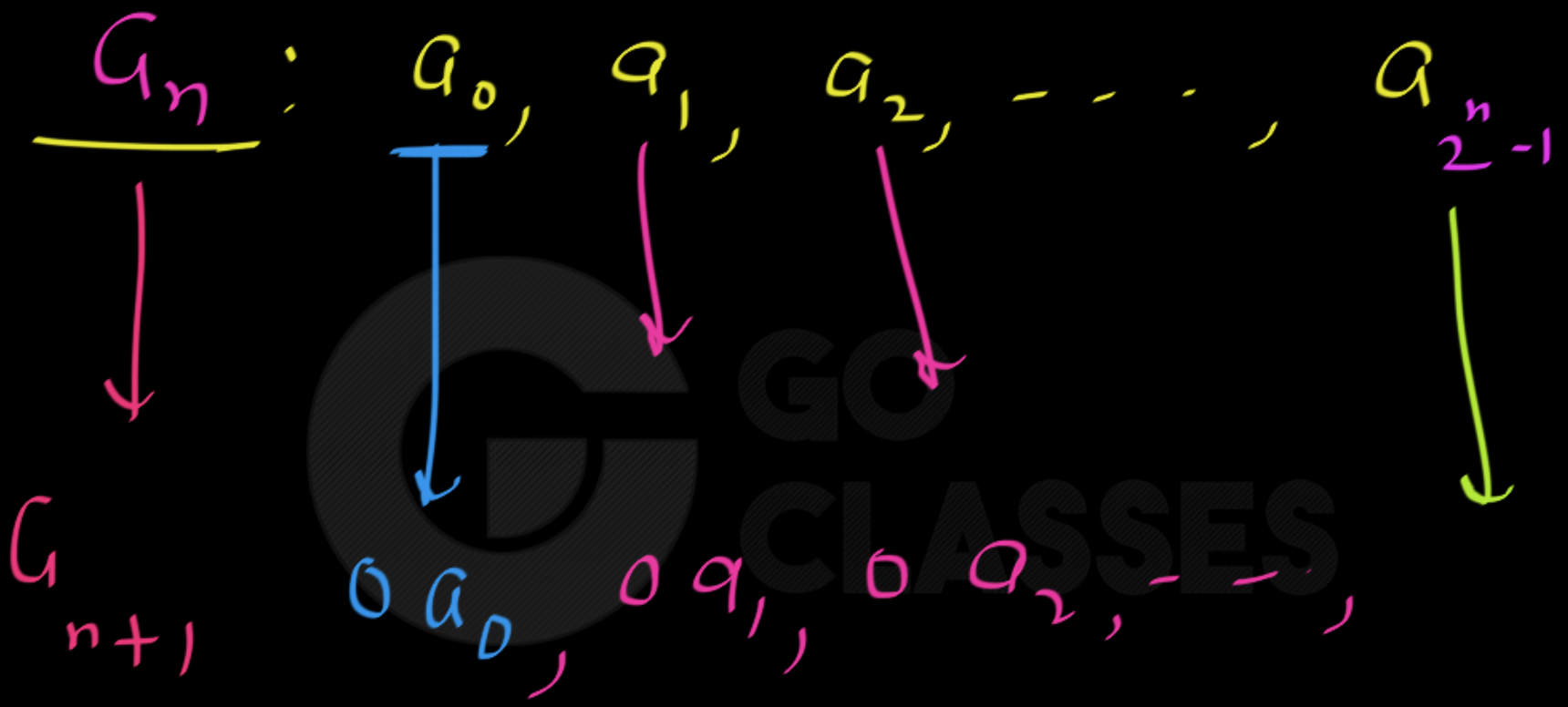
0, 1 00, 01, 1, 0

G_2 is embedded as first half of G_3.

G_2 G_3

00, 01, 11, 10

000, 001, 011, 010, ...


$G_n : a_0, a_1, a_2, \ldots, a_{2^n-1}$
G_{n+1}
$0a_0, 0a_1, 0a_2, \ldots,$

- G_n is a permutation of the numbers 0, …, $2^n - 1$. (Each number appears exactly once in the list.)
- G_n is embedded as the first half of G_{n+1}.
- Therefore, the coding is *stable*, in the sense that once a binary number appears in G_n it appears in the same position in all longer lists; so it makes sense to talk about *the* reflective Gray code value of a number: $G(m)$ = the mth reflecting Gray code, counting from 0.
- Each entry in G_n differs by only one bit from the previous entry. (The Hamming distance is 1.)
- The last entry in G_n differs by only one bit from the first entry. (The code is cyclic.)

Gray Codes

Gray Code Finding Another Way

the nth Gray code is obtained by computing $n \oplus \left\lfloor \frac{n}{2} \right\rfloor$.

Deminal number = n

↓

Gray Code := $n \oplus \left\lfloor \frac{n}{2} \right\rfloor$

bitwise EXOR

Decimal = 29 ⟶ Gray Code of 29?

↓

binary: 11101

↓

Gray 10011

Decimal = 29 ⟶ Gray Code of 29?

$m = 29 = 11101$

$\left\lfloor \frac{m}{2} \right\rfloor = 14 = \oplus\ 01110$

10011

Gray Code of 29

C-Program:

```
main {
int m;
int a = m/2;
r = a ^ m;
```

m → binary

m/2 → $\left\lfloor \frac{m}{2} \right\rfloor$

What is the Content of r ?

C-Program:

```
main {
int m;
int a = m/2;
r = a ^ m;
}
```

m → binary

$\lfloor \frac{m}{2} \rfloor$

What is the Content of r ? ⇒ Gray code of m.

$m \oplus (m/2)$ $\equiv$ Gray code of m.

bitwise Exor

$m = 14 =$ 1 1 1 0 → Gray code of 14 → 1 0 0 1

$\left\lfloor \frac{m}{2} \right\rfloor = 7 =$ 0 1 1 1

⊕

1 0 0 1

→ Gray code of 14

Example

Grey code for 23:

$(23)_{10} = (0001\ 0111)_2$

Function: Result = Num $\oplus$ (Num / 2)

$//(\text{div } 2)_{10} == (>> 1)_2$

	0	0	0	1	0	1	1	1	
$\oplus$	0	0	0	0	1	0	1	1	
	0	0	0	1	1	1	0	0	$= (28)_{10}$

Gray Code:

1. Motivation
2. Recursive Construction / Reflexive property
3. binary $\longleftrightarrow$ Gray
4. Properties of G_n

5 Decimal m

Gray code $= m \oplus \left\lfloor \frac{m}{2} \right\rfloor$

6 Application: Error Correction/Detection

CN → hamming code

(7) Application :

- Hypercube Graph