BIT MANIPULATION 2

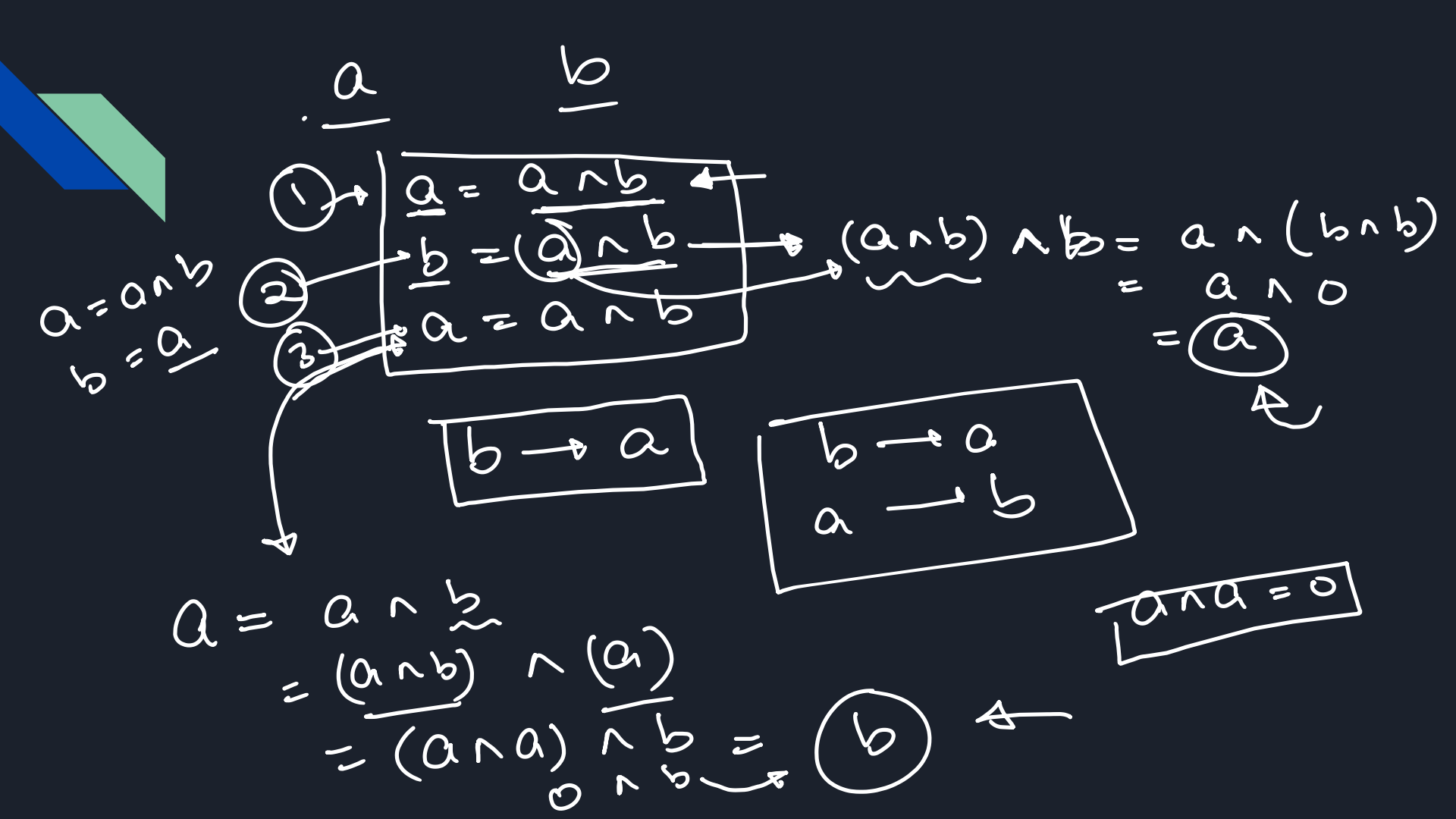
By Harsh Gupta

Goal

- Cool tips for bit manipulation.
- Brute-forcing using bitmasks.
- Understanding bitsets.

Some Cool Tricks

- Swap 2 numbers without a temporary variable
- (2^k - 1) has last k bits set
- __builtin_popcount and __builtin_popcountll → counting the no. of set bits.
- __builtin_clz / __builtin_clzll → count leading zeros
- __builtin_ctz / __builtin_ctzll → counts # of trailing zeros.


a b
1
2
3
a = a ∧ b
b = a ∧ b
a = a ∧ b
a = a ∧ b
b = a
b → a
b → a
a → b
(a ∧ b) ∧ b = a ∧ (b ∧ b)
= a ∧ 0
= a
a = a ∧ b
= (a ∧ b) ∧ (a)
= (a ∧ a) ∧ b = b
0 ∧ b
a ∧ a = 0

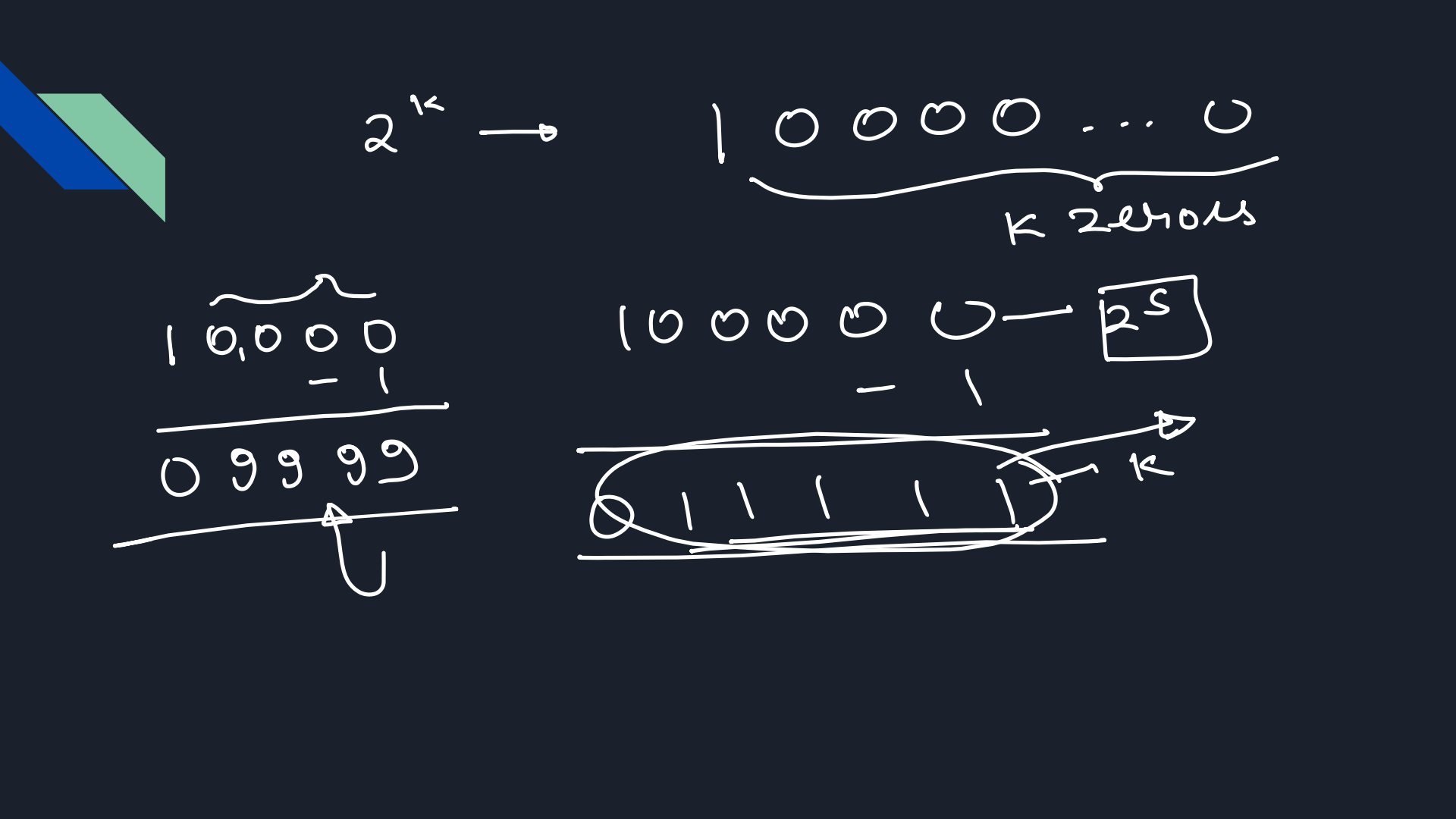
$2^k - 1$

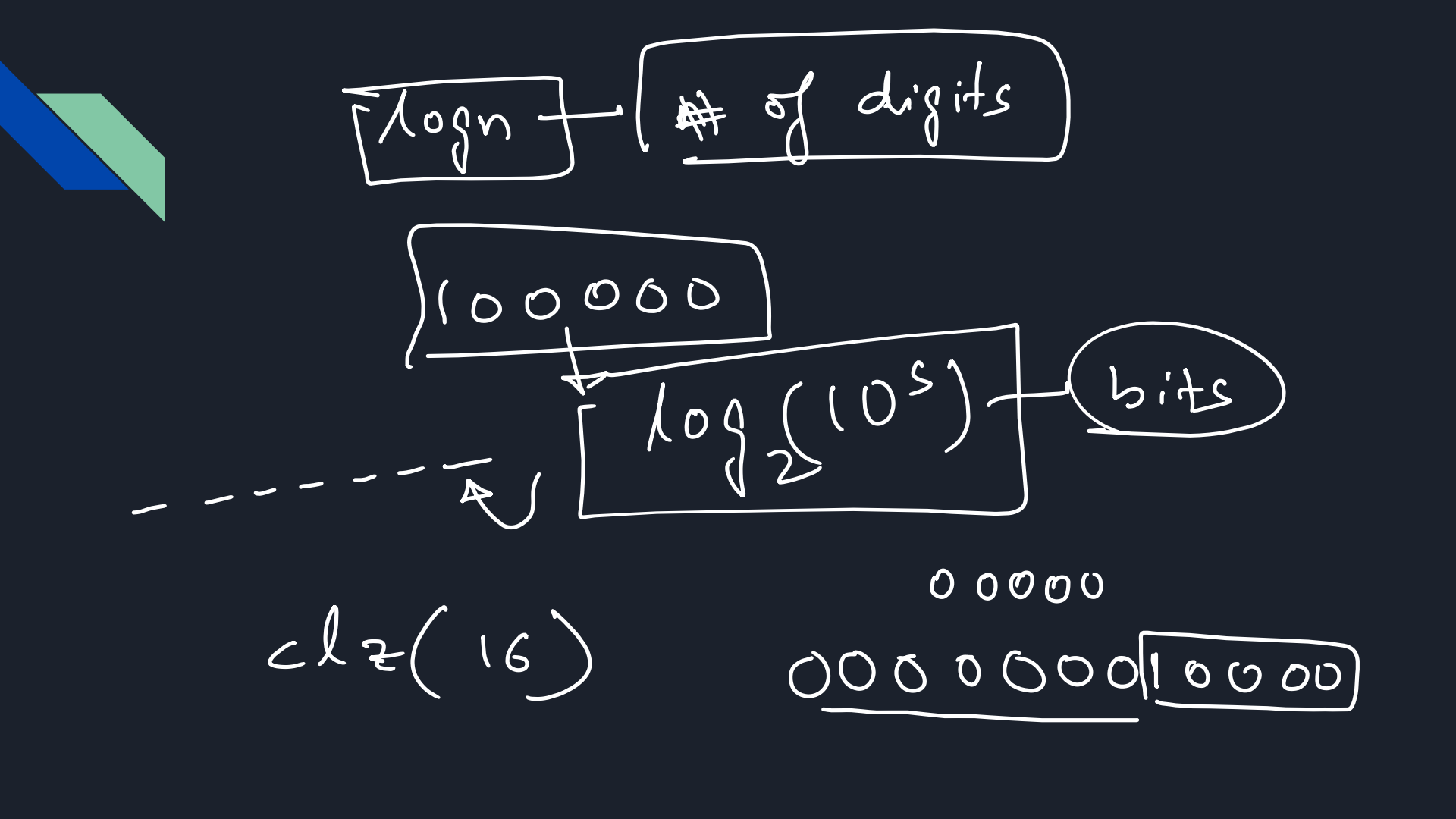
$2^k \rightarrow 100000$ (K zeros)

0 | 0

100000
− 1
99999

100000
− 1
0 1 1 1 1 (K ones)


$2^K \rightarrow$ 1 0 0 0 0 ... 0
K zeros
1 0 0 0 0
- 1
0 9 9 9 9
1 0 0 0 0 0 — 2^5
- 1
K
1 1 1 1 1


log n
# of digits
100000
log2(10^5)
bits
clz(16)
00000
000000000
10000

Bitmasking

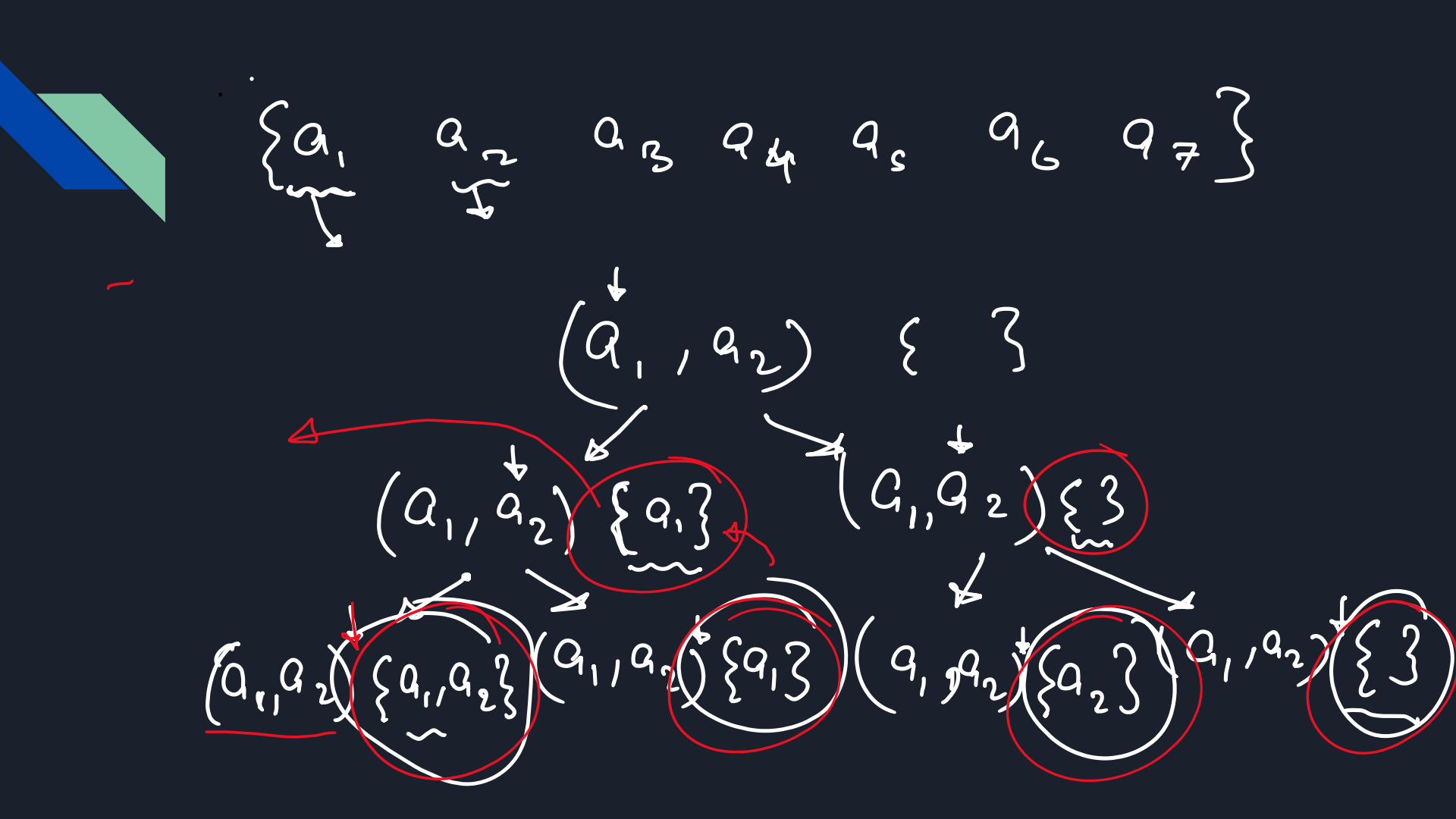
- A bitmask is nothing but a sequence of bits where every bit represents something.

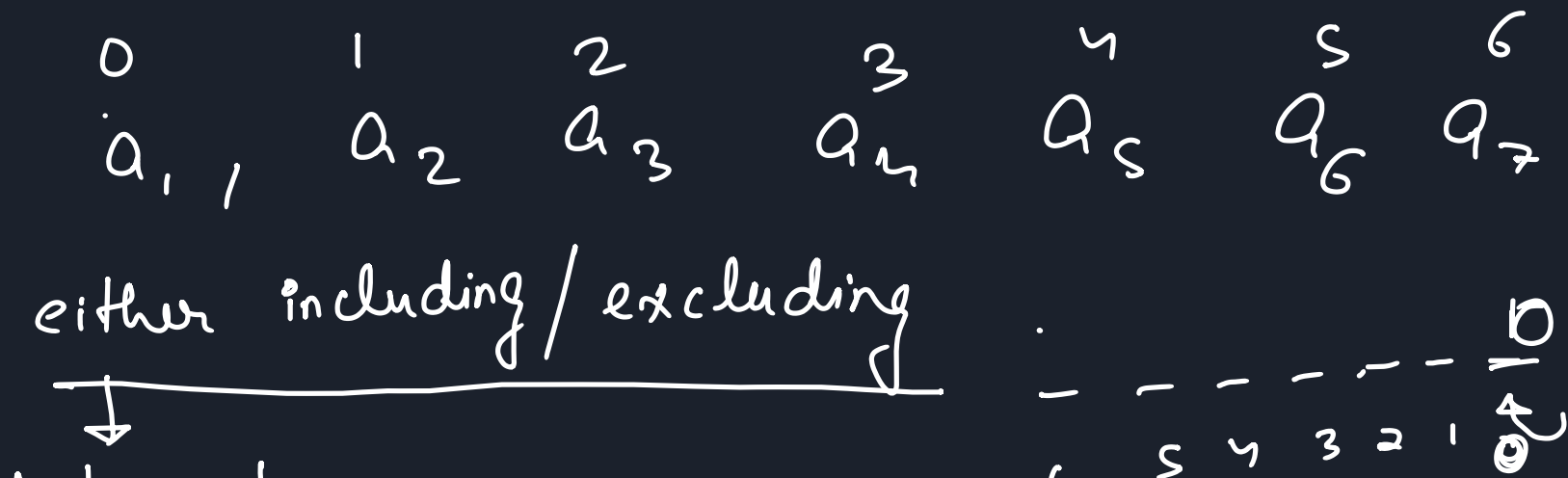
Think of this as a boolean array which takes less memory.

Memory consumed by a boolean array of size 30 = 30 bytes = 240 bits

Here we are using 1 byte for storing true/false which could have been represented by a single bit.

Can you think of some better way which consumes less memory?


{q1 q2 q3 q4 q5 q6 q7}
(q1, q2) { }
(q1, q2) {q1}
(q1, q2) { }
(q1, q2) {q1, q2}
(q1, q2) {q1}
(q1, q2) {q2}
(q1, q2) { }


0 1 2 3 4 5 6
a1 a2 a3 a4 a5 a6 a7
either including / excluding
bitmasking
6 5 4 3 2 1 0
6 5 4 3 2 1 0
this particular index is not taken
this particular index is taken

bool vector → false

, 7 bits

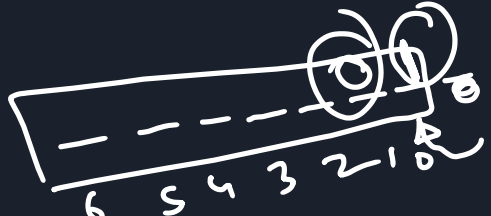
8 bits

three

1 0 0 1 0 0 0

7

subset having element with index 3rd and 6th



1/0

all the subsets

vector<bool>

T/F → 8 bits

30×8 = 240 bits

Instead of using an array of 30 boolean variables

We can use a sequence of 30 bits where

ith bit will be set if a[i] was true

else ith bit will be unset

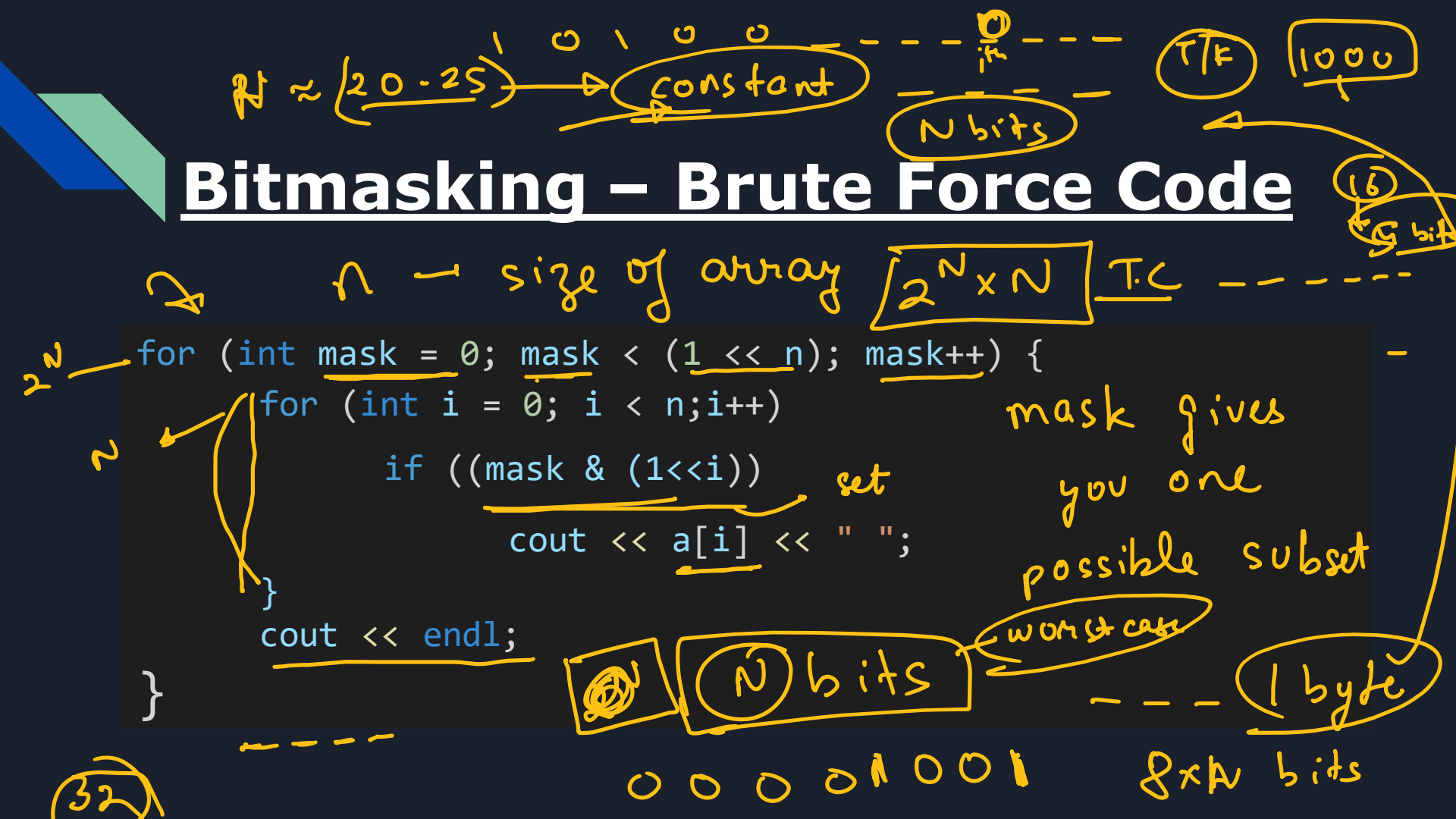
We can use a 32-bit integer for allocating this memory so total memory used now is just 32 bits.

dp with bitmasking

32 bit numbers


n numbers in the array
2^N - 1
0
0 0 0 0 0 0 0 0
mask
0 → empty set
(minimum)
0 (maximum)
2^N - 1
when we include all the elements.
last N bits as 1
(1<<n)-1
1 0 0 0 0 0 0 0
2^n - 1
10^2
100
- 1
99

Bitmasking – Brute Force Code



n = 3

{ 1, 2, 3 }
0th 1st 2nd

for (mask 0 → 7)

2nd bit

10100
00100
00100

10000
00100
00000

mask → 0
mask → 1
mask → 2
mask → 3
mask → 4
mask → 5
mask → 6
mask → 7

000 → { }
001 → { 1 }
010 → { 2 }
011 → { 1, 2 }
100 → { 3 }
101 → { 1, 3 }
110 → { 2, 3 }
111 → { 1, 2, 3 }

2^3

THINK

4 numbers of 32 bits

45 → 2nd number

320

45 - 32 = 13 → 13th bit

But what if the array size was 100?

We cannot create a single variable with that many bits.

But we can create, 4 32-bit integers or 2 64-bit integers.

Looks like it is becoming complicated.

If we required 320 bits then it might become complicated for us.

Bitsets

In C++ STL, we have a data structure for this - **Bitsets**

Syntax:-

Defining bitset:

```
bitset<320> bit;
```

Accessing and setting ith bit:

bit[i] can be used to access and set ith bit just like we use arr[i] to access or set the ith element of an array

Bitsets (CONTD...)

- Binary Operations can be performed on Bitsets
- We can initialize bitsets with some binary string or integer also.

Some Useful Functions:-

- **bit.count()** -> returns the number of set bits
- **bit.flip()** -> flips all the bits
- **any(), none()** and **all()**
- **to_string()** -> returns binary string

Bit Independent Problems

In these problems, whether the current bit is set or not doesn't affect the other bits, so we can just solve for every bit individually.

Problem:

- XORwice

Bit Dependent Problems

Here the bits are interdependent, whether we set a bit or unset it can affect other bits.

Example - Find the largest number less than 2^8 where at most 3 bits are set.

Here the whether we can set a bit or not depends upon the other bits as we cannot set more than 3 bits.

Problem:

- Gardener and the Array

largest $< 2^3$ → atmost 3 bits set