Bitwise Operations

What are Bits ?

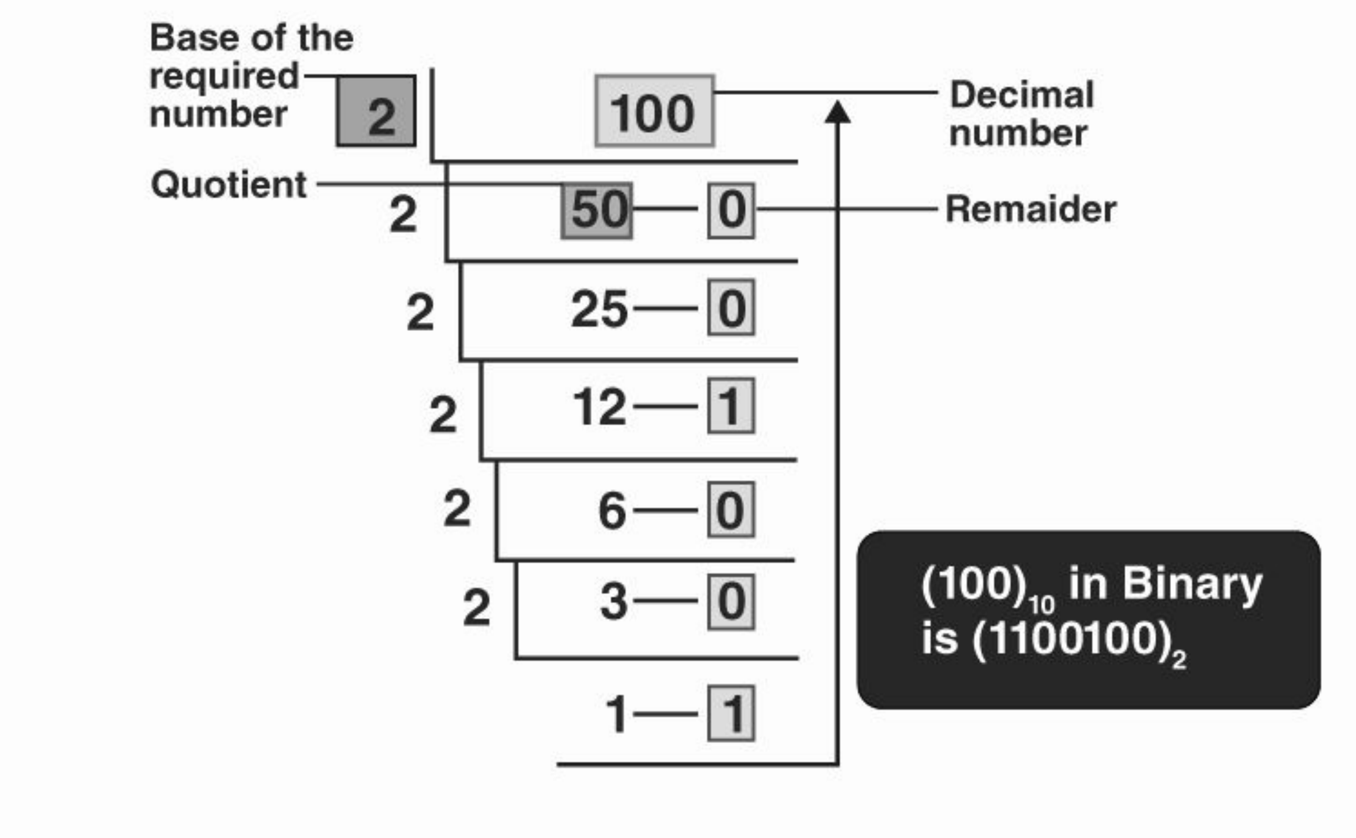
Bit Representation for unsigned?

Why Bitwise Operations?

- Some questions require bit manipulations.

- It can be used to optimize solutions and simplify solutions.

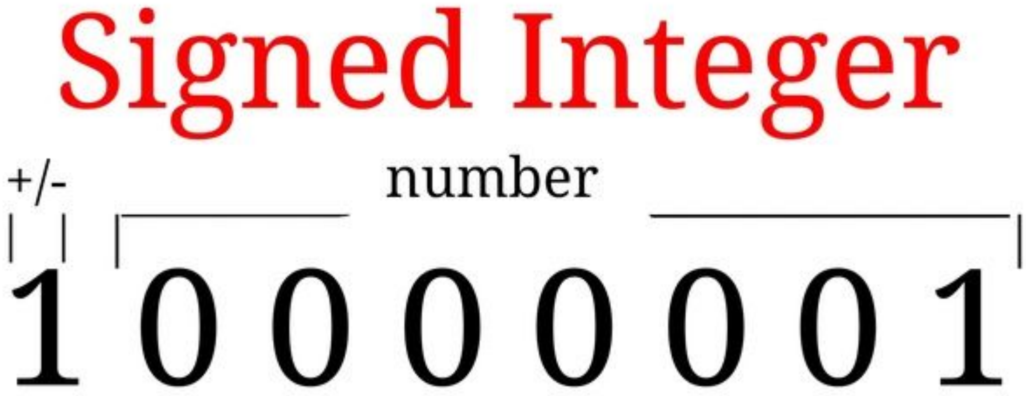
Bit Representation



A2SV
Africa To Silicon Valley

Bit Representation for signed?

Bit Representation



2's Complement

A method to represent negative numbers

Most popular method

Allows adding negative numbers with the same logic gates as positive numbers

Main Idea: $x + (-x) = 0$

Africa To Silicon Valley

2’s Complement

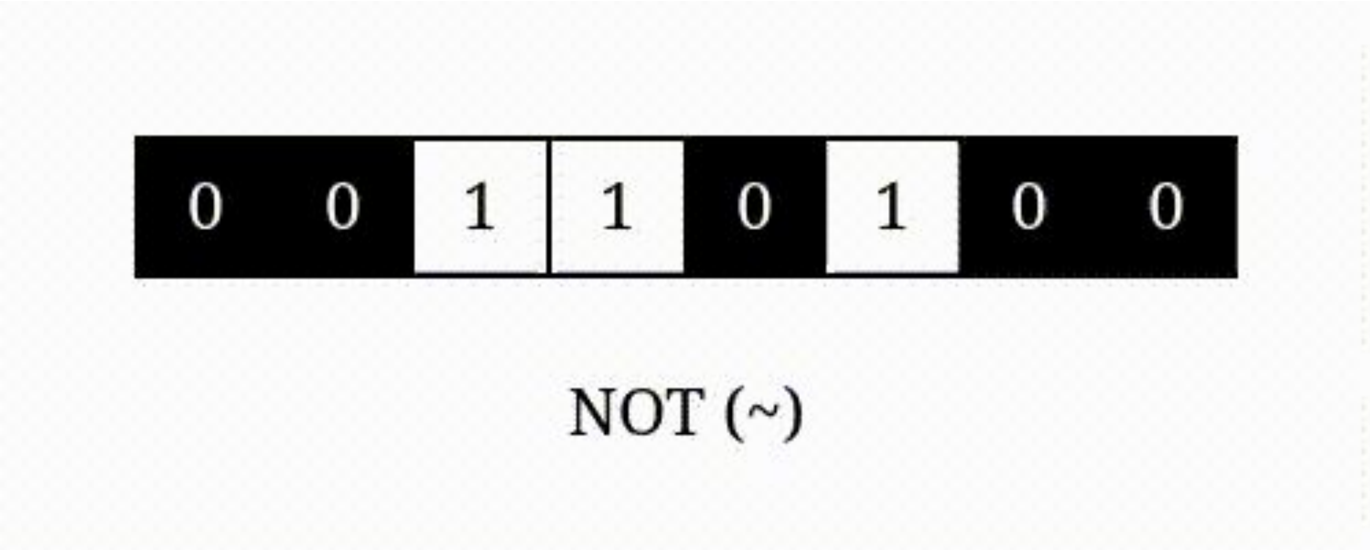
How to convert number to 2's complement

1. Convert the positive number to binary
2. Flip the bits (1 to 0 and 0 to 1)
3. Add 1

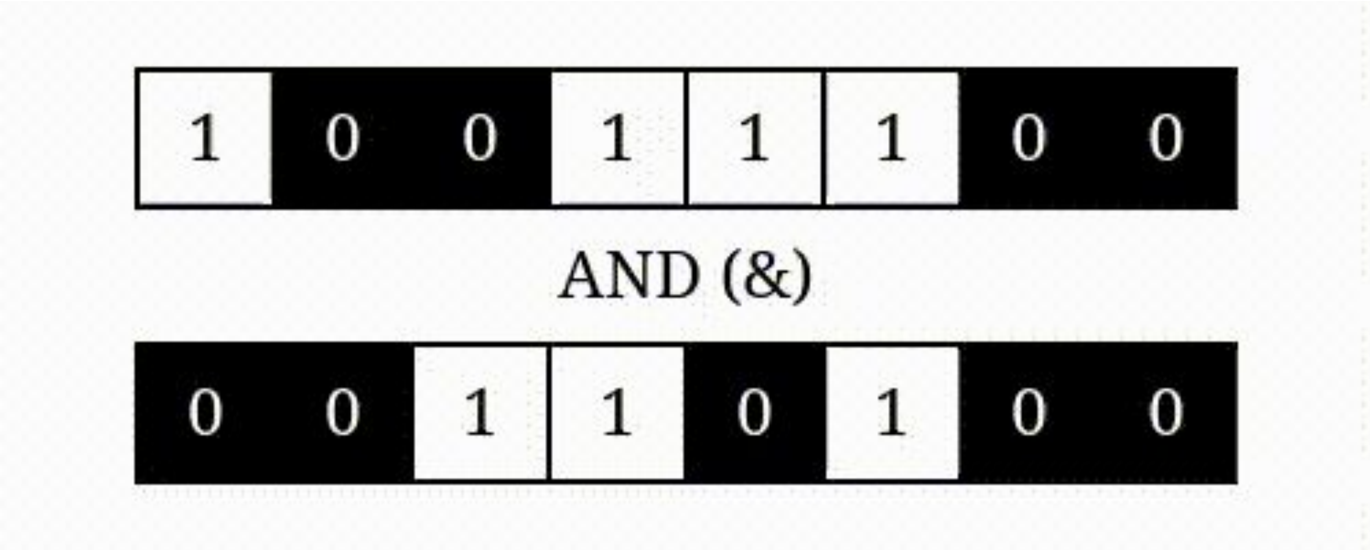
Bit Operators

- NOT (~)
- AND (&)
- OR (|)
- XOR (^)
- Bit Shifts (<<,>>)

NOT



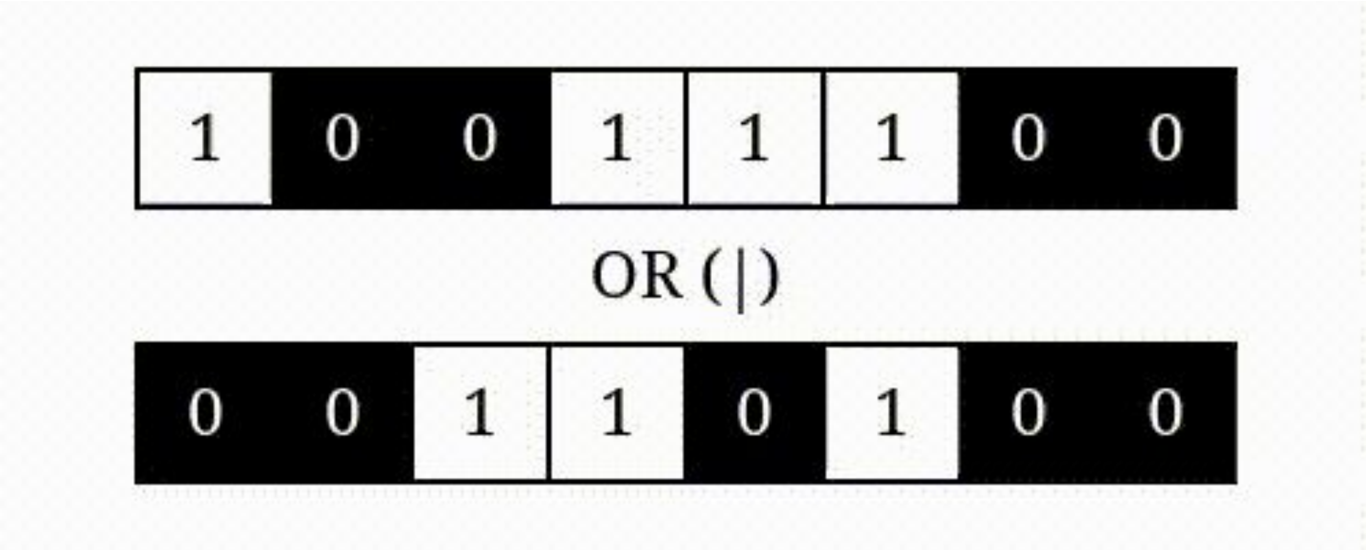
AND



Check yourself

$3 \,\&\, 2 = ?$

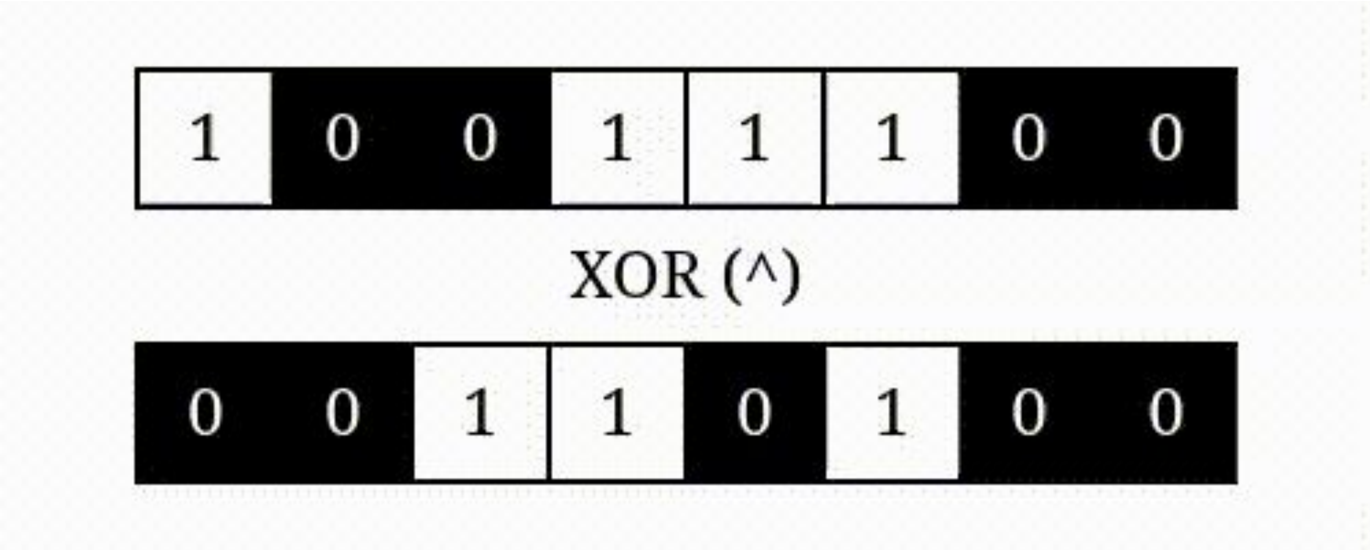
OR



Check yourself

$$7 \mid 8 = ?$$

XOR

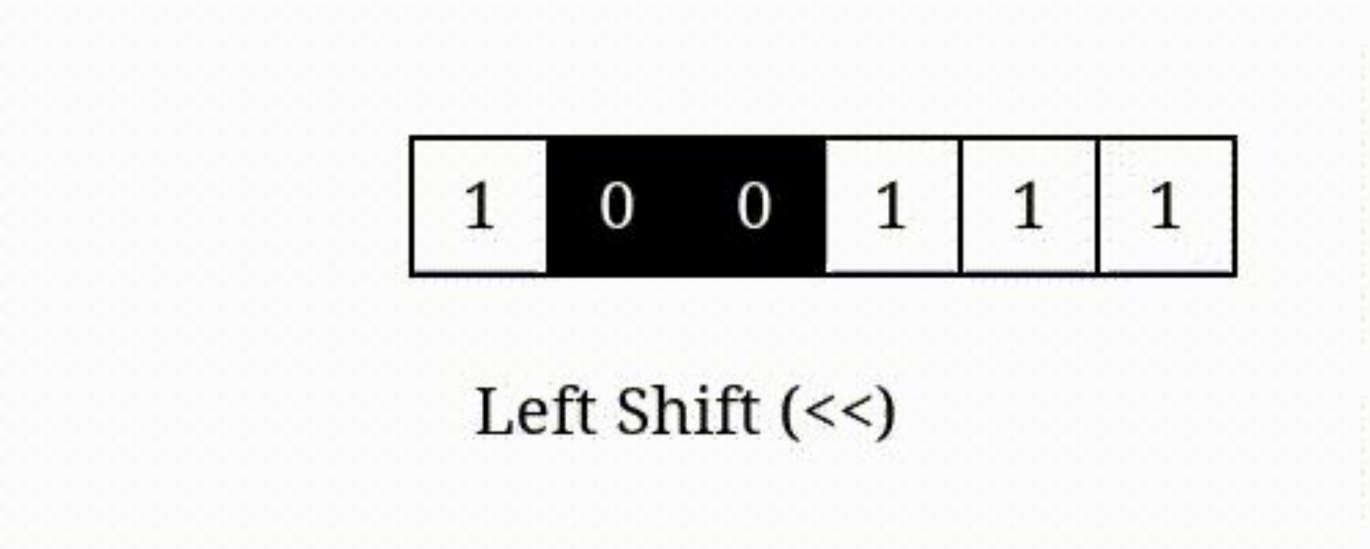


Check yourself

$4 \wedge 9 = ?$

A2SV
Africa To Silicon Valley

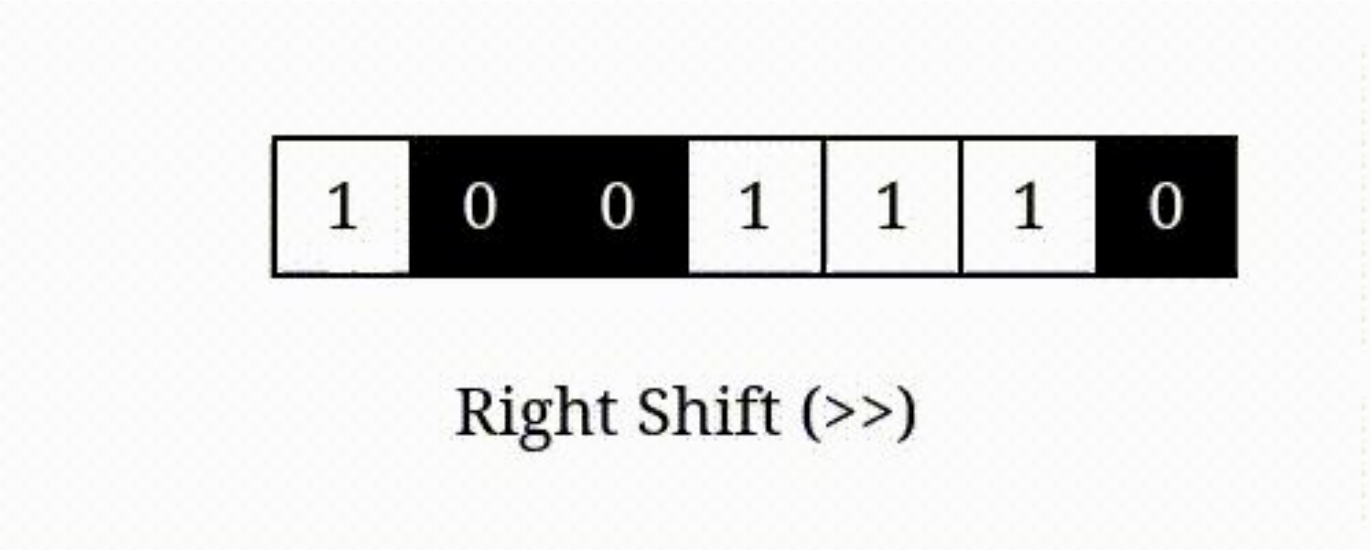
Left Shift



Check yourself

$$1 << 3 = ?$$

Right Shift



Check yourself

$$16 >> 3 = ?$$

- Each right shift operation reduces the number to its half.

A2SV
Africa To Silicon Valley

Bit Operators

A	B	A | B	A & B	A^ B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

A2SV
Africa To Silicon Valley

Question #1

First let's solve it using normal approach.

Then let's solve it using bits (shifting operation)

338. Counting Bits

Hint

Easy · 8.6K · 408

Companies

Given an integer `n`, return *an array* `ans` *of length* `n + 1` *such that for each* `i` (`0 <= i <= n`), `ans[i]` *is the* ***number of*** `1`***'s*** *in the binary representation of* `i`.

Example 1:

```
Input: n = 2
Output: [0,1,1]
Explanation:
0 --> 0
1 --> 1
2 --> 10
```

A2SV Africa To Silicon Valley

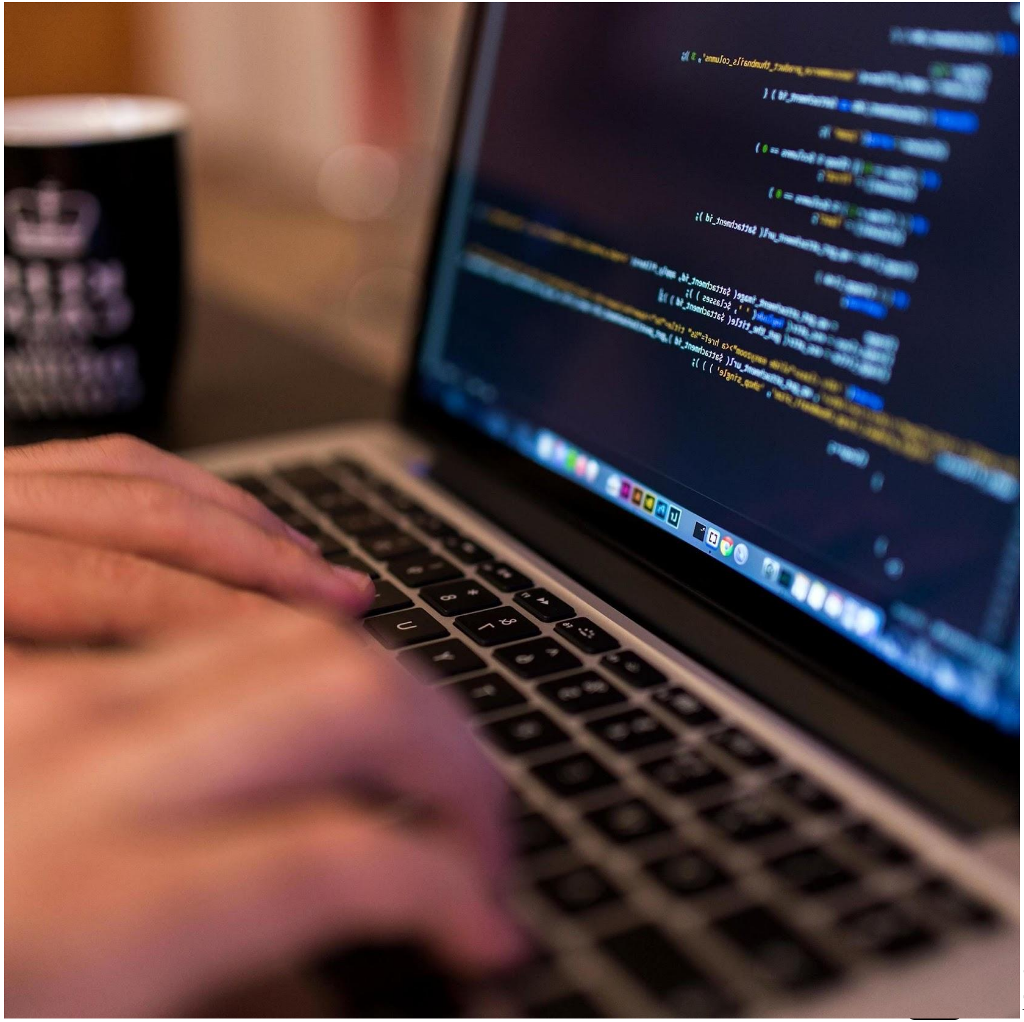
Bitwise operators properties

- Commutative
  - x ^ y = y ^ x
- Associative
  - x & (y & z) = (x & y) & z

- Does this property hold for all bit operations?

Africa To Silicon Valley

Practice Time



Question #2

First let's solve it using normal approach.

Then let's solve it using bits.

[0,2] ⇒ 2

268. Missing Number

Easy 7834 3002 Add to List Share

Given an array `nums` containing `n` distinct numbers in the range `[0, n]`, return *the only number in the range that is missing from the array.*

Example 1:

```
Input: nums = [3,0,1]
Output: 2
Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.
```

Example 2:

```
Input: nums = [0,1]
Output: 2
Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.
```

Example 3:

```
Input: nums = [9,6,4,2,3,5,7,0,1]
Output: 8
Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.
```

Bit masking

- Way of optimizing storage
- Store information in a single bit

Test 5th Bit

```
num = 10001000100111

 1  = 00000000000001
```

Africa To Silicon Valley

Test 5th Bit

```
num  = 10001000100111

 1   = 00000000000001

1<<5  = 00000000100000


   10001000100111
 & 00000000100000
   00000000100000 (!= 0)
```

Africa To Silicon Valley

Implement

```
import sys
# kth bit from the right, 1 indexed
def kthBitTest(num : int, k : int) -> int:
    # TODO
    Mask = 1 << k
    Result = Num & num
    If result != 0:
        Return True
    Retur False


def test():
```

Implement

```
import sys
# Set the Kth bit from the right, 1 indexed
def kthBitSet(num : int, k : int) -> int:
    # make a musk, the musk goint 1<<k
    # we or it
    Mask = 1<<k
    New_num = num | mask


def test():
    assert kthBitSet(6, 1) == 7,'Ooops'
    assert kthBitSet(3, 4) == 11, 'Ooops'
    print('Niceee')
```

Implement

```python
import sys
# turn off the kth bit from the right, 1 indexed
def turnOffKthBit(num : int, k : int) -> int:


def test():
    assert kthBitTest(6, 1) == 6,'Ooops'
    assert kthBitTest(6, 2) == 4,'Ooops'
    assert kthBitTest(3, 4) == 3, 'Ooops'
    print('Niceee')

test()
```

Africa To Silicon Valley

Bit masking

Test k^{th} bit is set: `num & (1 << k) != 0`.

Set k^{th} bit: `num |= (1 << k)`.

Turn off k^{th} bit: `num &= ~(1 << k)`.

Toggle the k^{th} bit: `num ^= (1 << k)`.

46. Permutations

Medium 15.1K 256

Companies

Given an array `nums` of distinct integers, return *all the possible permutations*. You can return the answer in **any order**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Example 2:

```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```

Example 3:

```
Input: nums = [1]
Output: [[1]]
```

Question #3

- Use bit mask to keep track of used numbers

Python Built-in Functions

- `bin()`: This built-in function can be used to convert an integer to a binary string.

- `int()`: This built-in function can be used to convert a binary string to an integer.

Africa To Silicon Valley

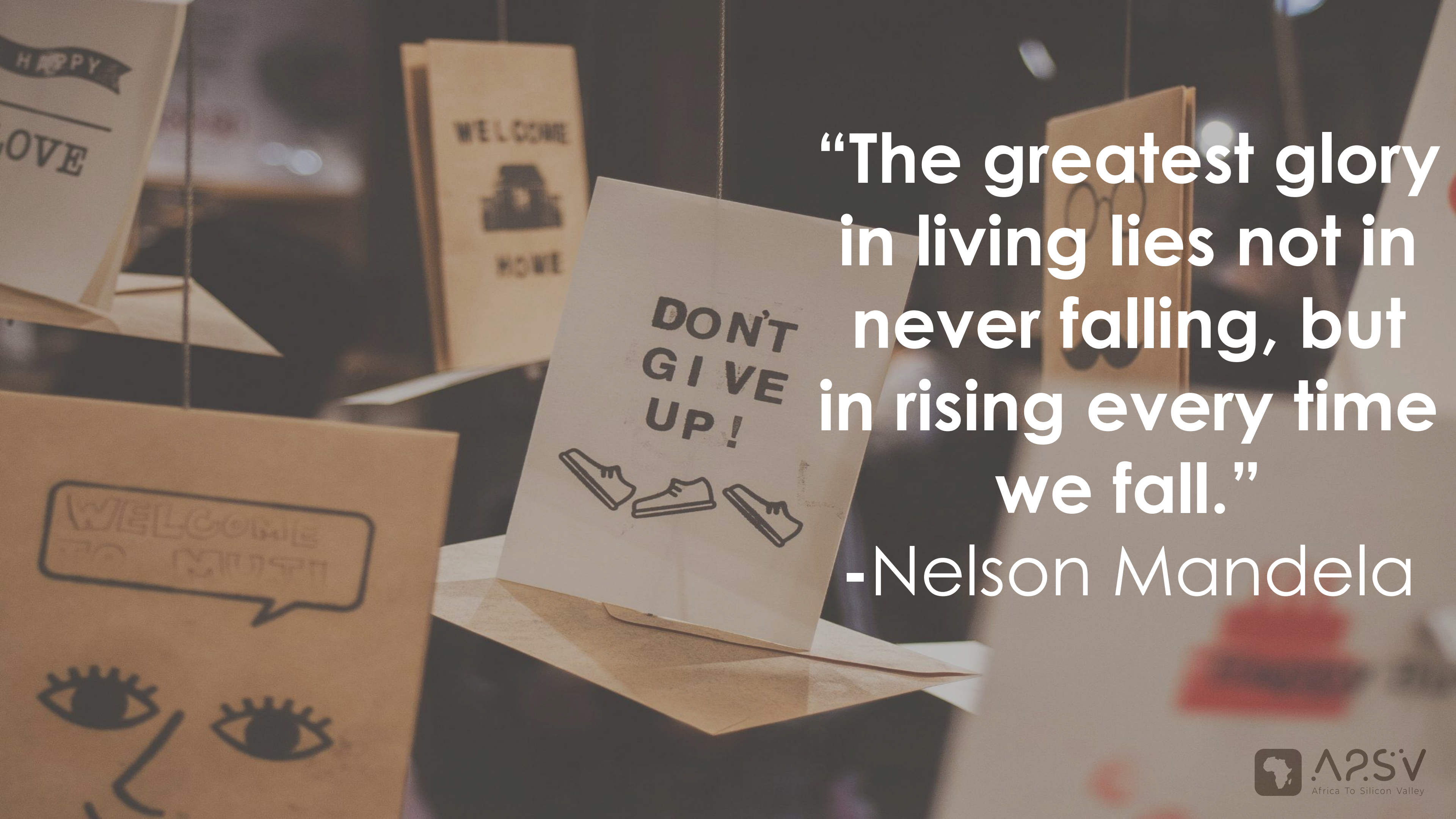
Python Built-in Functions

- `bit_length()`: This method can be called on an integer and returns the number of bits required to represent the integer in binary, excluding the sign bit.

- `bit_count()` : this method can be called on an integer and returns the number of set bits.

Practice Problems

- Single Number II
- Single Number III
- Number Complement
- Sum of Two Integers
- Add Binary
- Hamming Distance
- Counting Bits
- Subsets
- Count Words Obtained After Adding a Letter



"The greatest glory in living lies not in never falling, but in rising every time we fall."

-Nelson Mandela

Africa To Silicon Valley