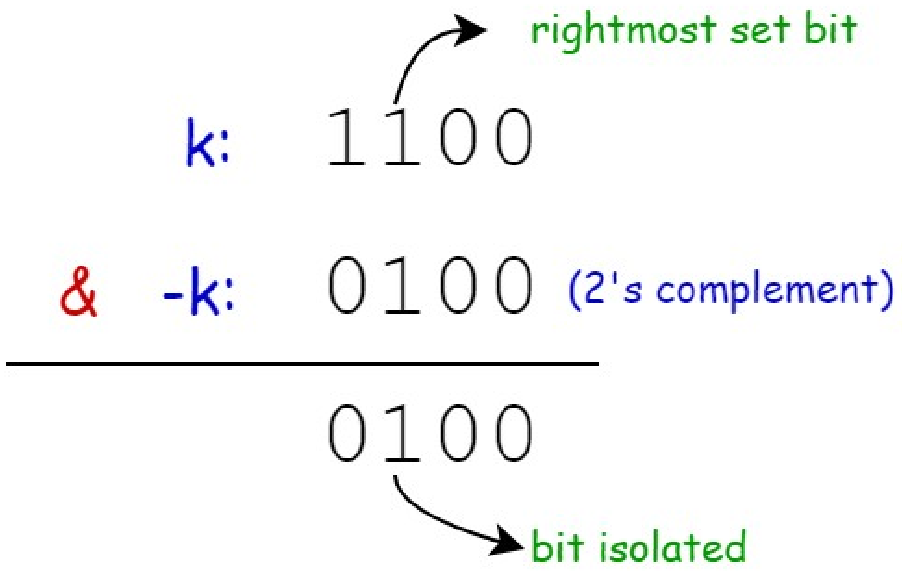
1545. Find Kth Bit in Nth Binary String

leetcode editorial

We begin by using the expression `k&−k` to find the rightmost set bit in `k`. This operation isolates the smallest power of 2 in `k`. Why is this important? The rightmost set bit indicates how deep we are in the sequence's structure, guiding us to the appropriate section of the sequence, especially in relation to any inversions.

The following diagram illustrates how we isolate the rightmost set bit using `k&−k`:

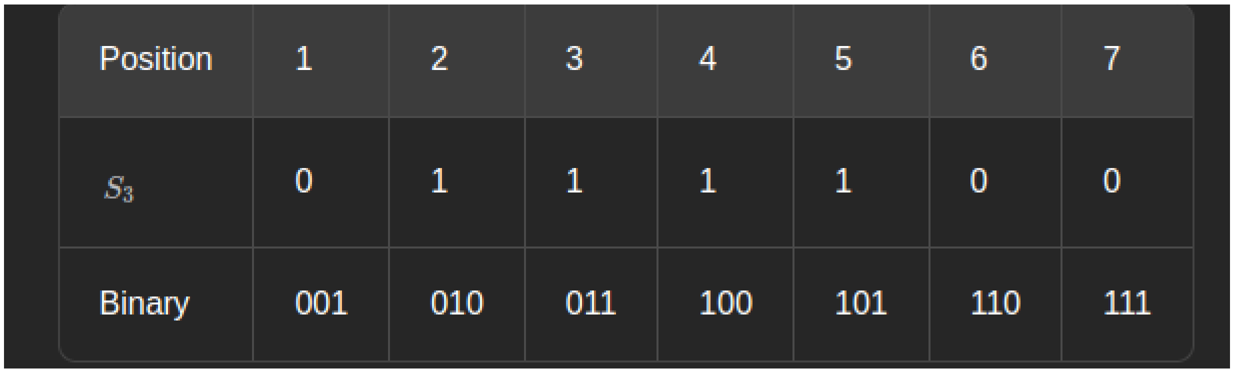


To clarify the above concept, let’s break down the binary representation of positions step by step.

Consider the following sequences:

- S1="0"
- S2="0"+"1"+"1"="011"
- S3="011"+"1"+"100"="0111100"

Now, let’s analyze S3="0111100" in detail:



Position	1	2	3	4	5	6	7
S_3	0	1	1	1	1	0	0
Binary	001	010	011	100	101	110	111

1. First Bit (Leftmost) of the Binary Representation:
    - If it’s 0, we are in the left half of the string (positions 1-3).
    - If it’s 1, we are either in the right half (positions 5-7) or the middle (position 4).
2. Second Bit of the Binary Representation:
    - In the left half (positions 1-3 represented as 001-011):
        - If it’s 0, we are in the left quarter (position 1).
        - If it’s 1, we are in the right quarter of the left half (positions 2-3).
    - In the right half (positions 5-7 represented as 101-111):
        - If it’s 0, we are in the left quarter of the right half (position 5).
        - If it’s 1, we are in the right quarter (positions 6-7).
3. Third Bit (Rightmost) of the Binary Representation:
    - This indicates whether we are at an odd or even position.

This pattern continues for larger strings. Each bit in the binary representation narrows down which section of the string we are examining.

For instance, consider position 6 (binary 110):

- The first bit is 1, indicating we are in the right half of the string.
- The second bit is 1, showing we are in the right quarter of the right half.
- The third bit is 0, indicating we are at an even position.

From this information, we can determine:

1. Whether we are in an inverted section (right half).
2. How many times the bit has been inverted, which depends on our depth in the sections.
3. What the original bit was based on the odd/even position.

This is how the binary representation of the position correlates with the string's structure. We know this approach might seem unconventional and is tougher than what you might have originally thought of. We recommend dry-running this approach a couple of times to digest it completely, simply reading this explanation is not enough.

So to determine if the bit at position `k` has been inverted, we check the bits to the left of the rightmost set bit. We calculate `k` divided by `positionInSection` (the result of `k&−k`) and then shift the result right by one bit.

If the resulting bit is 1, it indicates we are in a section of the sequence that has been inverted. We then need to ascertain the original state of the bit, regardless of any inversions. If `k` is even, the original bit is 1; if `k` is odd, the original bit is 0. This check tells us the bit's state before any transformations occur.

Finally, we decide what to return based on whether `k` is in an inverted section:

- If `k` is in an inverted part, we flip the original bit (changing 0 to 1 or 1 to 0).
- If `k` is not in an inverted part, we return the original bit as it is.

Algorithm

- Calculate the position within the current section by performing a bitwise AND operation between `k` and its two's complement (`-k`).
- Determine if the bit is in an inverted part of the sequence:
  - Divide `k` by the position in section.
  - Right shift the result by 1 bit.
  - Perform a bitwise AND with 1.
  - Check if the result equals 1.
- Determine if the original bit (before any inversions) is a 1:
  - Perform a bitwise AND between `k` and 1.
  - Check if the result equals 0.
- If the bit is in an inverted part of the sequence:
  - Return '0' if the original bit was 1, otherwise return '1'.
- If the bit is not in an inverted part:
  - Return '1' if the original bit was 1, otherwise return '0'.