perplexity

why we subtract 1

Great question! The reason we often **subtract 1** in expressions like $2^n - 1$ comes from how binary counting and bit representation work in computers.

Why Do We Subtract 1? ($2^n - 1$)

1. Range Starts at Zero

When counting with n bits in binary:

- You start at **0** (all bits zero).
- You end at $2^n - 1$ (all bits one).

For example, with 3 bits:

- Possible values: 000 (0), 001 (1), 010 (2), 011 (3), 100 (4), 101 (5), 110 (6), 111 (7)
- That's 8 values, from 0 up to 7.
- The maximum value:
  $2^3 - 1 = 8 - 1 =$ **7** (binary 111)

2. Bit Masks and Boundaries

- If you want to represent or mask n bits, setting all n bits to 1 gives you $2^n - 1$.
  - 1 bit: 1 (1)
  - 2 bits: 3 (11)
  - 8 bits: 255 (1111 1111)
- Why not just 2^n? Because 2^n would require an extra bit (it's 1 followed by n zeros).

3. Counting the Total Combinations

- With n bits, you can make 2^n different patterns, counting from 0. So the highest pattern you can reach is $2^n - 1$.
- Example: 8 bits (a byte) gives numbers from 0 to 255 (not 256).

4. Common in Programming

- **Arrays**: The last index is length – 1 (`arr[arr.length - 1]`)
- **Maximum unsigned value**: $2^n - 1$

Analogy

Think of how we count from 0 to 9 (for 10 numbers)—the highest value is 9, which is 10 – 1.

In Summary

- Computers count from **0 up to (2^n – 1)**, so we subtract 1 to get the maximum value representable with n bits.
- **Subtracting 1** makes sure we end at the largest value possible for the chosen number of bits.

If you see "minus one" in these formulas, it's usually to indicate the highest value or last item, given zero-based counting.