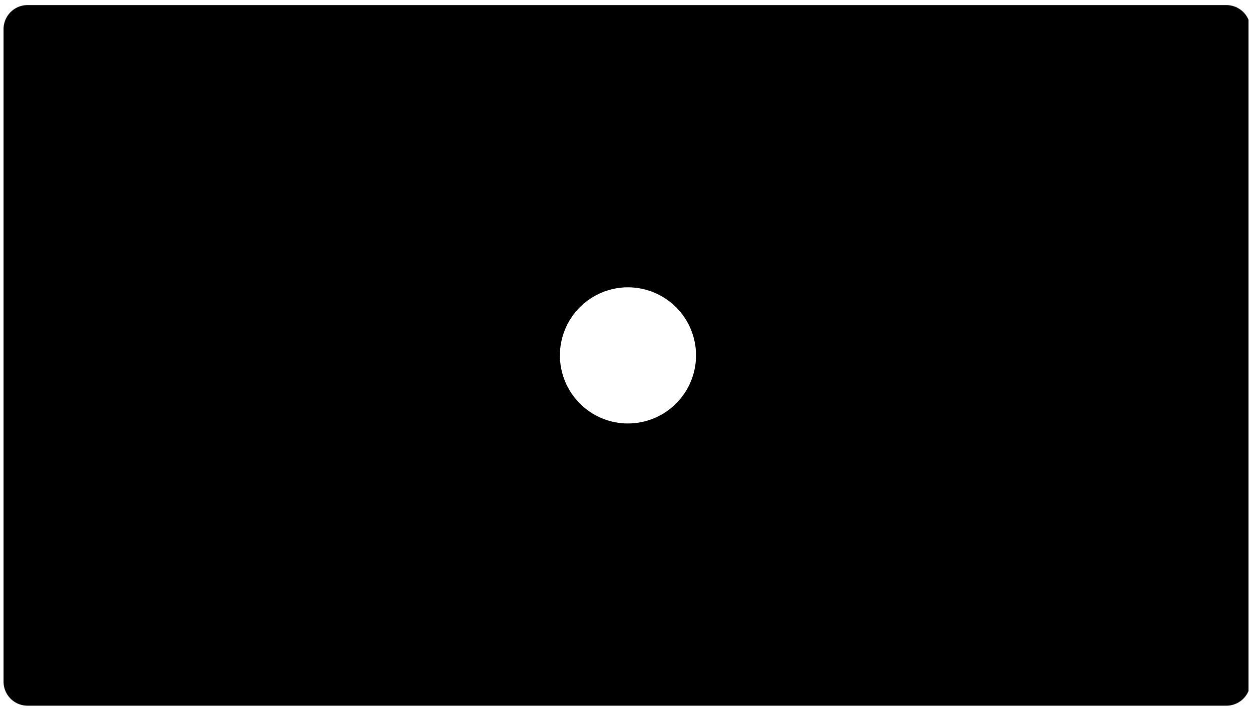
Cyclic Property of Bits



External Practice Problems

[Maximum Number That Sum of the Prices Is Less Than or Equal to K](#)

Mark Status

Problem Statement:

You are given an integer k and an integer x. The price of a number num is calculated by the count of set bits at positions x, 2x, 3x, etc., in its binary representation, starting from the least significant bit. The following table contains examples of how price is calculated.

X	NUM	BINARY REPRESENTATION	PRICE
1	13	000001101	3

X	NUM	BINARY REPRESENTATION	PRICE
2	13	000001101	1
2	233	011101001	3
3	13	000001101	1
3	362	101101010	2

The accumulated price of num is the total price of numbers from 1 to num. num is considered cheap if its accumulated price is less than or equal to k.

Return the greatest cheap number.

Solution

Understanding the Problem

1. **Objective**: The task involves calculating the price of a number based on the count of set bits (1s) at specific positions in its binary representation.

2. **Price Calculation**: The price of a number is determined by counting the set bits at positions x, 2x, 3x, and so on, in its binary representation, starting from the least significant bit.

3. **Example**: For instance, if x=1, then for the number 13 (binary representation: 000001101), the price would be 3 because there are set bits at positions 1, 2, and 4.

4. **Cyclic Property of Bits**:

   - In binary representation, bits exhibit a cyclic pattern as numbers increase.
   - Each column (bit position) toggles between 0 and 1 at regular intervals.
   - The interval at which a bit toggles doubles for each subsequent bit position.

Observations and Approach:

1. **Bit Patterns**: Observing the binary representation of numbers from 0 to num helps identify patterns in each column (bit position).

2. **Column Behavior**: Each column exhibits a specific pattern due to the cyclic property of bits:

   - The 1st column toggles after every 1 row.
   - The 2nd column toggles after every 2 rows.
   - The 3rd column toggles after every 4 rows.

- The pattern continues with each column toggling at intervals of powers of 2.

3. **Grouping Bits**: Columns can be grouped into blocks where each block contains a certain number of 0s followed by the same number of 1s. This grouping helps in counting set bits efficiently.

4. **Formula Derivation**: Using the observations, a formula is derived to calculate the number of set bits in each column up to a given number num.

Using Binary Search

1. **Objective**: The goal is to find the greatest cheap number within a specified range.

2. **Binary Search Logic**: Binary search is employed due to the property that greater numbers have higher prices, and lesser numbers have lower prices.

3. **Algorithm Steps**:

   - Start binary search within the range 1 to 1e15.
   - Check if the current number's price is within the given budget (k).
   - If the price is acceptable, search on the right side of the current number.
   - If the price exceeds the budget, search on the left side of the current number.

By incorporating the cyclic property of bits, the solution efficiently calculates the price of numbers and finds the greatest cheap number within the specified range.

Code:

Copy

```
class Solution {
public:
    // utility function to calculate the number of bits in a number
    int countBits(long long n) {
        int count = 0;
        while(n) {
            count++;
            n = n >> 1;
        }
        return count;
    }
    // utility function to quickly get power of 2 using bit manipulation
    long long powerOf2(int i) {
        return 1LL << i;
    }
    long long calc(long long n, int x) {
        // get the leftmost bit
        int i = countBits(n);
        long long price = 0;

        // increment n to account 0th row in the count of groups
        n++;
        while(i) {
            // if current column is valid, calculate the number of 1s in the c
            if(i % x == 0)
                price += (n / powerOf2(i)) * (powerOf2(i - 1)) + max(0LL, (n %

            // move to the next column
            i--;
        }

        return price;
    }
    long long findMaximumNumber(long long k, int x) {
        long long low = 1;
        long long high = 1e15;

        // binary search
        while(low <= high) {
            long long mid = (high - low) / 2 + low;
            long long res = calc(mid, x);
            if(res <= k)
                low = mid + 1;
            else
                high = mid - 1;
        }

        return high;
    }
};
```

Time Complexity:

The provided solution consists of three main functions:

1. `countBits`: Counts the number of bits in a given number using bit manipulation, with a time complexity of O(log n).
2. `powerOf2`: Returns 2 raised to the power of a given input using bit manipulation, with a time complexity of O(1).
3. `calc`: Calculates the price of a number based on the count of set bits at specific positions in its binary representation, with a time complexity of O(log n).

The `findMaximumNumber` function utilizes binary search to find the greatest cheap number within a specified range. It iteratively calls the `calc` function to determine the price of each number, with a binary search time complexity of O(log range * log n), where range refers to the range of numbers being searched and n represents the input number passed to the `calc` function.

Overall, the solution efficiently computes the desired result by leveraging bit manipulation techniques and binary search algorithm, with a time complexity of O(log range * log n).

Completed

Write a comment

B I U

• Before you post...

I Understand

- **No Cheating or Violation of Honor Code** : Users are usually prohibited from discussing or sharing solutions for active contests. This is to prevent cheating and to maintain the integrity of the platform.
- **Avoid Posting Full Solutions** : Users are encouraged not to post entire solutions to problems, especially for active contests. Instead, it's common to share hints, tips, or discuss general approaches without giving away the entire solution.
- **Be Respectful** : Users are expected to be respectful and considerate of others. Avoid using offensive language, making personal attacks, or engaging in any form of harassment.
- **Stay On Topic** : Discussions should stay relevant to the problem at hand. Off-topic discussions or spamming are generally discouraged.
- **No Advertising or Self-Promotion** : Users should not use the platform for advertising or self-promotion. This includes promoting external services, products, or personal projects.