Below are the comprehensive revision notes for the live class on bit manipulation concepts. This class covers the foundational aspects of binary systems, data types, and important bitwise operations in detail.

Bit Manipulation Revision Notes

Introduction to Number Systems

Decimal and Binary Systems

- **Decimal System**: Commonly used in daily life. Consists of 10 digits (0-9).
- **Binary System**: Used by computers. Comprises 2 digits (0 and 1). The binary equivalent of numbers in the decimal system can be calculated by converting each decimal digit to its binary form【7:8†transcript】.

Binary Addition

- Binary addition is akin to decimal addition, but it is handled in terms of 0s and 1s.
  - Example of binary addition:
    - $10101 + 1101 \longrightarrow 100010$.
    - Here, one must carry over whenever two 1s are added, leading to an overflow to the next bit【7:13†transcript】.

Negative Numbers in Binary

Two's Complement Representation

- **Two's complement** is the standard way of representing negative numbers in binary.
- To find the two's complement of a number:
  1. Flip all bits (change 0s to 1s and vice versa).
  2. Add 1 to the least significant bit (LSB).
  - Example for -5 using 8 bits:
    - Binary of 5: `00000101`

- This result now represents -5 in 8-bit binary【7:2†transcript】.

Bitwise Operators

Types of Bitwise Operators

- **AND (`&`)**: Results in 1 if both bits are 1.
- **OR (`|`)**: Results in 1 if at least one of the bits is 1.
- **XOR (`^`)**: Results in 1 if only one of the bits is 1.
- **NOT (`~`)**: Inverts all bits (0 becomes 1, and 1 becomes 0) 【7:10†transcript】.

Practical Example with XOR

- XOR can be used in algorithms to find the single non-duplicate element in an array where every other element appears twice:
  - For an array `[1, 3, 5, 1, 3]`, XORing all elements will cancel out duplicates, leaving the unique element【7:11†transcript】.

Data Types and Memory Considerations

Working with Different Data Types

- The choice of data type (e.g., `int`, `long`) impacts how much data can be stored.
- For large values, using a 64-bit `long` instead of a 32-bit `int` might be essential to prevent overflow, particularly in calculations involving large sums or data arrays【7:9†transcript】.

Importance of Constraints in Determining Data Types

- When dealing with arrays or large numbers, assess constraints to determine appropriate data types to prevent overflow and ensure accuracy in results【7:12†transcript】.

Conclusion

computer systems and programming operations. Understanding binary arithmetic, bitwise operations, and memory constraints is essential for efficient algorithm design and implementation.

These notes should serve as a thorough guide to revisiting and reinforcing the concepts discussed in the class. If there are further details required or particular examples needed, let me know!