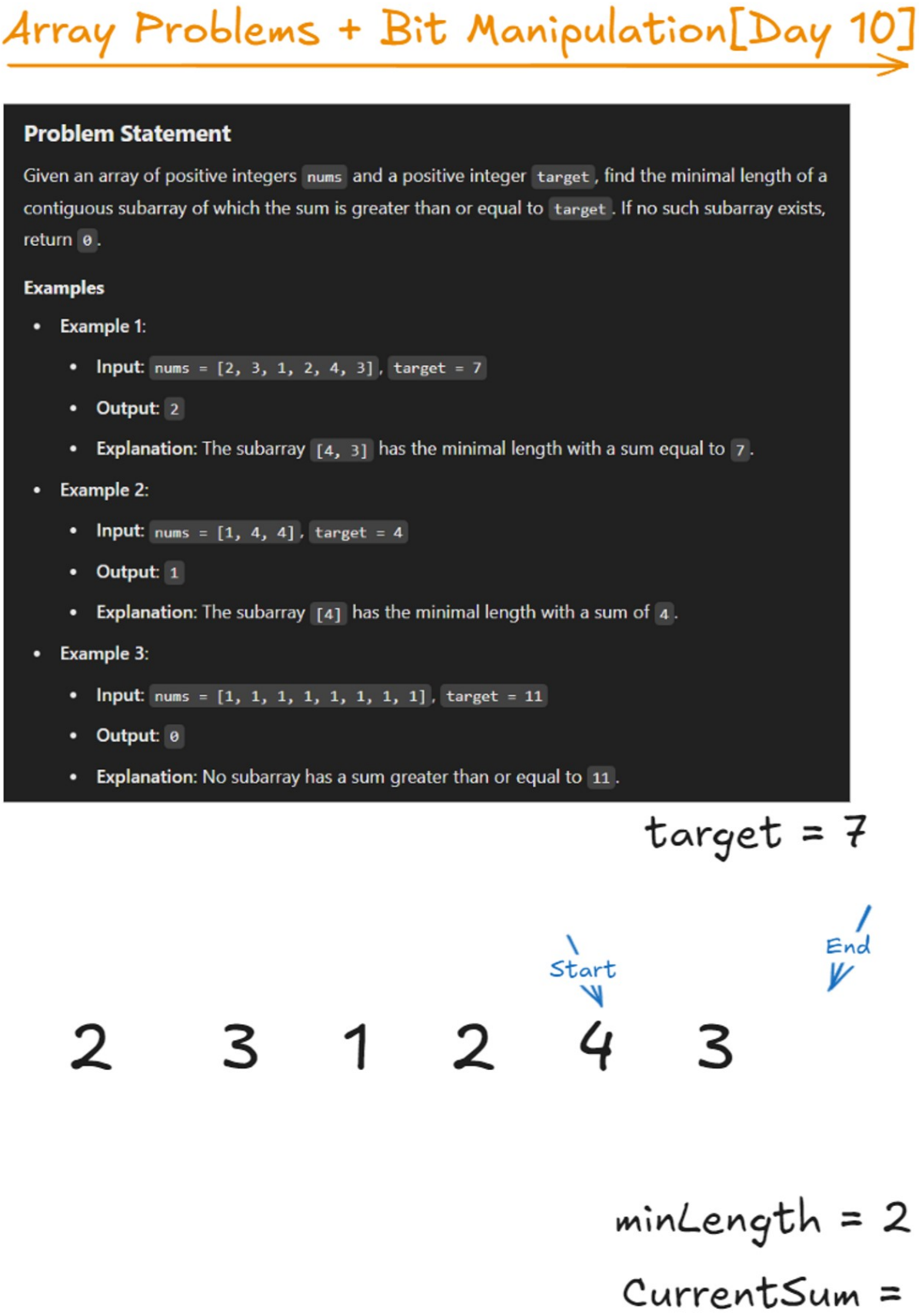
Array Problem + Bit Manipulation [Day 10]

12 September 2024 19:03



CurrentSum = 7

end - start + 1

```java
public class MinSizeSubarraySum {

    public static int minSubArrayLen(int target , int[] nums){
        int minLength = Integer.MAX_VALUE;
        int currentSum = 0;
        int start = 0;

        for(int end = 0; end < nums.length ; end++) {
            currentSum += nums[end];

            while(currentSum >= target){
                minLength = Math.min(minLength , end - start + 1);
                currentSum -= nums[start];
                start++;
            }
        }

        return(minLength == Integer.MAX_VALUE) ? 0 : minLength;
    }
    public static void main(String[] args) {
        int[] nums = {2,3,1,2,4,3};
        int target = 7;
        System.out.println(minSubArrayLen(target , nums));
    }
}
```

```java
public static int minSubArrayLen(int target , int[] nums
    int minLength = Integer.MAX_VALUE;
    int currentSum = 0;
    int start = 0;
    for(int end = 0; end < nums.length ; end++) {
        currentSum += nums[end];
        while(currentSum >= target){
            minLength = Math.min(minLength , end - start
            currentSum -= nums[start];
            start++;
```

Memory

minlength =2
crrSum = 3
start 5
end = 5
target 7

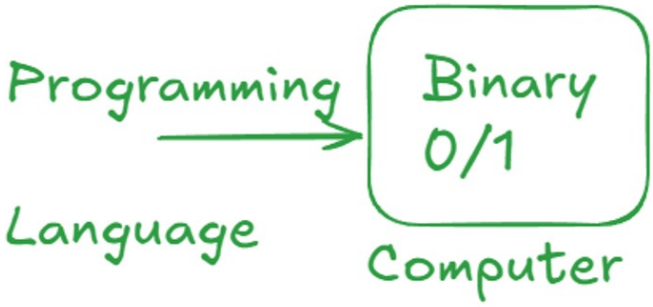
Start End

2 3 1 2 4 3

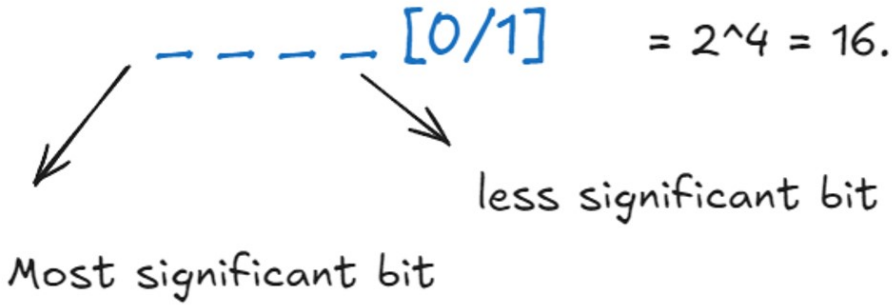
2	3	1	2	4	3
0	1	2	3	4	5

return minlength = 2;

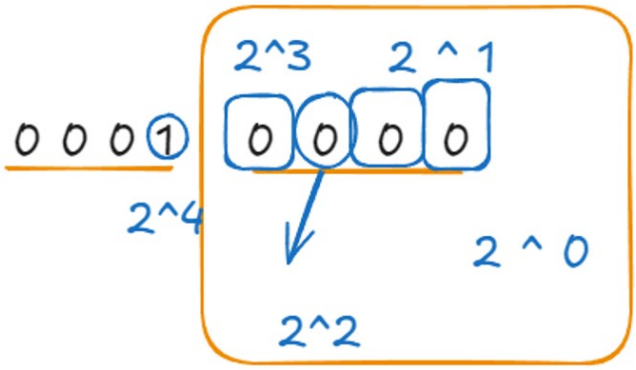
Bit Manipulation.



Convert the value from Decimal To Binary.



2	16
2	8 - 0
2	4 - 0
2	2 - 0



2	2	-	0
2	1	-	0
2	0	-	1

0 - 15

1 0 0 0 0

10000 - decimal. right to left

$0 * 2^0 + 0 * 2^1 + 0 * 2^2 + 0 * 2^3 + 1 * 2^4$
$= 0 + 0 + 0 + 0 + 1 * 16 = 16.$

10101 - decimal

$1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4$
$= 1 + 0 + 4 + 0 + 1 * 16 = 21.$

Decimal To Binary

19
23
31
9
17
27

Binary To Decimal

10010
10111
10100
00100
01101
10110

'&' operator

A & 0 = 0
A & 1* = A

6 & 7
- 0110
- 0111

0110 = 6

```
----------
 0110 = 6
```

1. 4 & 8 - 0100 & 1000 = 0000 = 0

2. 6 & 1 -0110 & 0001 = 0000 = 0

3. 4 & 5 - 0100 & 0101 = 0100 = 4

4. 3 & 9 - 0011 & 1001 = 0001 = 1

5. 7 & 3 - 0111 & 0011 = 0011 = 3

"|" Operator.

A | 0* = A
A | 1 = 1

```
6 | 7
= 0 1 1 0
  0 1 1 1
  ---------
  0 1 1 1 = 7
```

1. 4 | 8 - 0100 | 1000 = 1100 = 12

2. 6 | 1 -0110 | 0001 = 0111 = 7

3. 4 | 5 - 0100 | 0101 = 0101 = 5

4. 3 | 9 - 0011 | 1001 = 1011 = 11

5. 7 | 3 - 0111 | 0011 = 0111 = 7

"^" [XOR] Operator.

[if 2 same value = 0]
[if 2 different value = 1]

0 0 = 0
0 1 = 1
1 0 = 1
1 1 = 0

6 ^ 7
= 0 1 1 0
  0 1 1 1
  ----
  0 0 0 1 = 1

1. 4 ^ 8 - 0100 ^ 1000 = 1100 = 12

2. 6 ^ 1 -0110 ^ 0001 = 0111 = 7

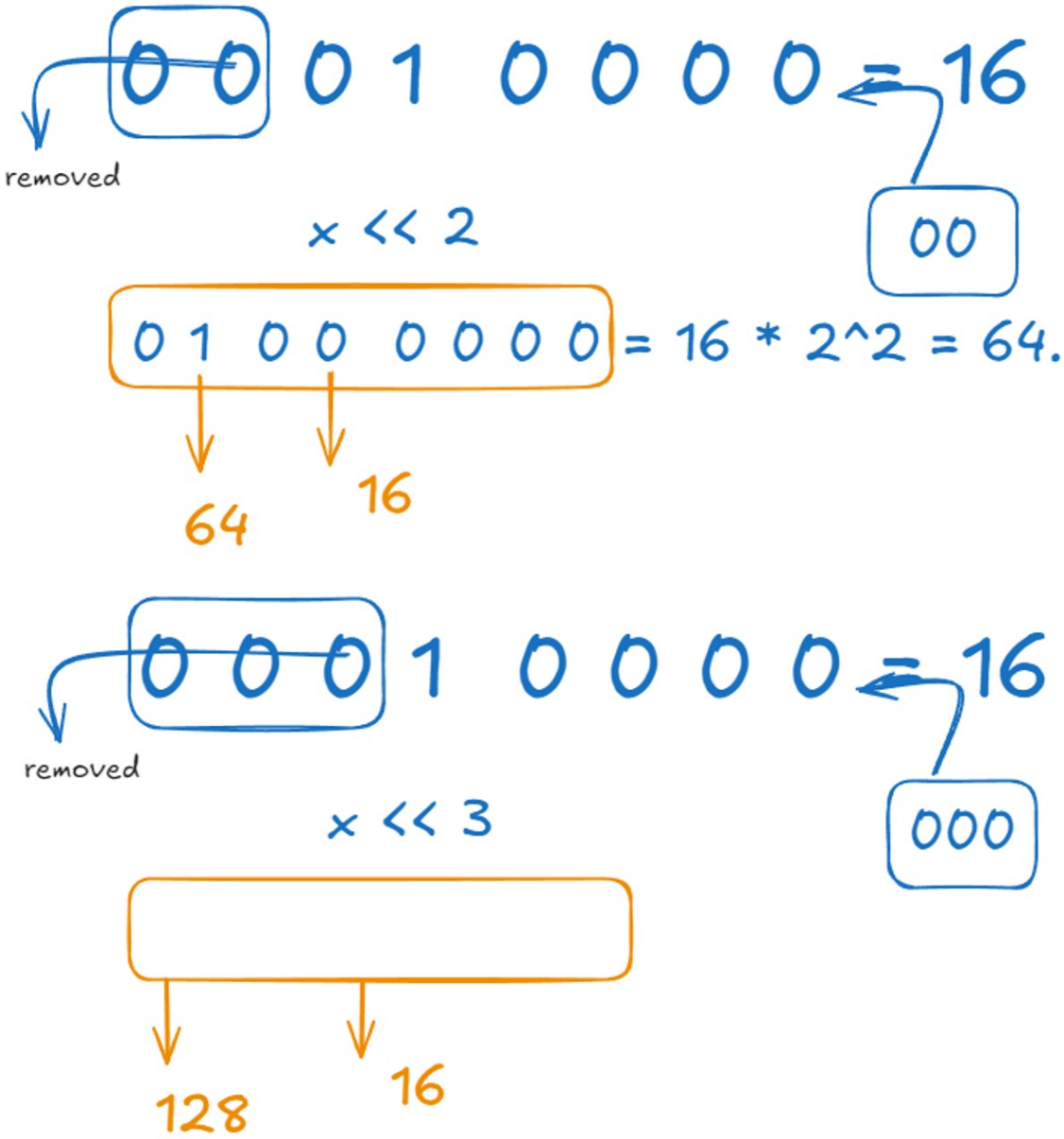
3. 4 ^ 5 - 0100 ^ 0101 = 0001 = 1

4. 3 ^ 9 - 0011 ^ 1001 = 1010 = 10

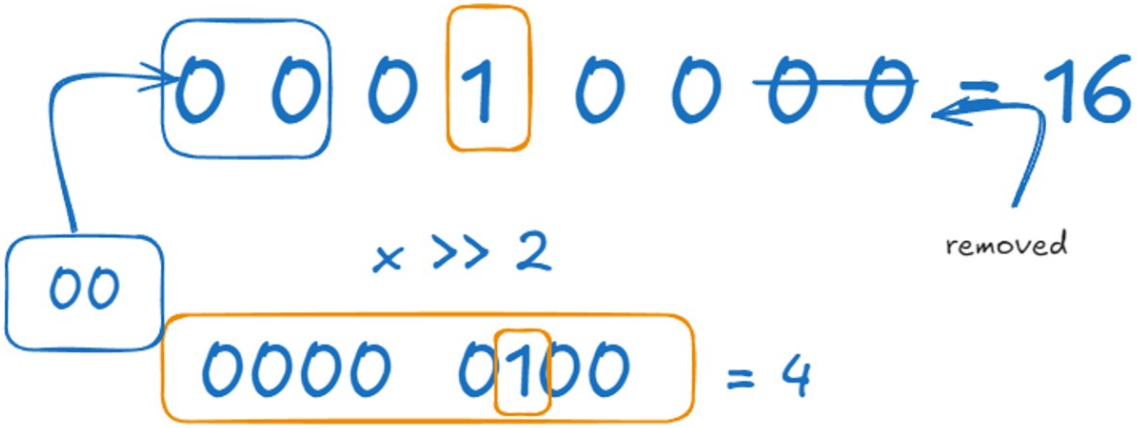
5. 7 ^ 3 - 0111 ^ 0011 = 0100 = 4

"<<" Left Shift

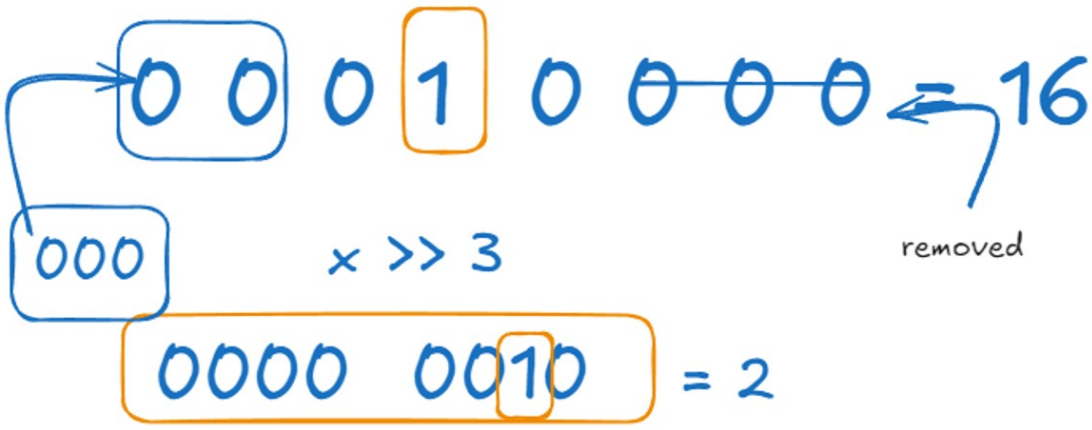
0 0 0 1 0 0 0 0 = 16


0 0 0 1 0 0 0 0 = 16
removed
00
x << 2
0 1 0 0 0 0 0 0 = 16 * 2^2 = 64.
64
16
0 0 0 1 0 0 0 0 = 16
removed
000
x << 3
128
16

"<<" Right Shift


0 0 0 1 0 0 0 0 = 16
removed
00
x >> 2
0000 0100 = 4

16 / 2^2 = 16 / 4 = 4.



16 / 2^3 = 16 / 8 = 2.

Question Time

1. 24 Make right/left shift 2,3

2. 36 Make right/left shift 2,1

Problem Statement

Sum of XOR of All Pairs

Given an array of integers, you need to find the sum of XOR of all possible pairs in the array.

Definition: For an array `arr` with `n` elements, you need to calculate the sum of `arr[i] ^ arr[j]` for all pairs `(i, j)` where `i < j`.

Example

Input:

Explanation:

- XOR of (1, 2) = `1 ^ 2 = 3`

Example

Input:

```
arr = [1, 2, 3]
```

Output:

```
6
```

Explanation:

- XOR of (1, 2) = `1 ^ 2 = 3`
- XOR of (1, 3) = `1 ^ 3 = 2`
- XOR of (2, 3) = `2 ^ 3 = 1`

Sum = 3 + 2 + 1 = 6

```
public class SumOfXor {

    public static int sumofXOR(int[] arr){
        int sum = 0;
        int n = arr.length;
        for(int i =0 ; i < n; i++){
            for(int j = i+1 ; j<n; j++){
                sum += arr[i] ^ arr[j];
            }
        }
        return sum;
    }
    public static void main(String[] args) {
        int[] arr = {1,2,3};
        System.out.println(sumofXOR(arr));
    }
}
```

arr = [4,5,6] --> [4,5] [4,6] [5,6]

[4,5] = 0100 ^ 0101 = 0001 - 1
[4,6] = 0100 ^ 0110 = 0010 - 2
[5,6] = 0101 ^ 0110 = 0011 - 3
1+2+3 = 6

Problem Statement

Problem Statement

You are given an integer `N`, and your task is to calculate the XOR of all numbers from 1 to `N`.

XOR Properties:

- **XOR is commutative and associative**, which means the order of operations doesn't matter:
  - `A ^ B ^ C = A ^ (B ^ C) = (A ^ B) ^ C`
- **XOR of a number with itself is 0**:
  - `A ^ A = 0`
- **XOR of a number with 0 is the number itself**:
  - `A ^ 0 = A`

Example:

If `N = 5`, the XOR of all numbers from 1 to 5 is:

XOR(1) = 1

XOR(1,2) = 1 ^ 2 = 3

XOR(1,2,3) = (1^2)^3 = 3 ^ 3 = 0

XOR(1,2,3,4) = (1^2^3)^4 = 4

XOR(1,2,3,4,5) = 4 ^ 5 = 1

XOR(1,2,3,4,5,6) = 1 ^ 6 = 7

XOR(1,2,3,4,5,6,7) = 7 ^7 = 0

XOR(1,2,3,4,5,6,7,8) = 0 ^ 8 = 8

N % 4 == 1 return 1.

N % 4 == 2 return N + 1;

N % 4 == 0 Return N.

else - return 0.

```
import java.util.Scanner;
```

```java
import java.util.Scanner;
public class XORFrom1ToN {
    public static int xorFrom1toN(int n){
        if(n % 4 ==0){
            return n;
        }else if(n% 4 == 1){
            return 1;
        }else if( n % 4 == 2){
            return n + 1;
        }else{
            return 0;
        }
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the value of N");
        int n = scn.nextInt();
        int result = xorFrom1toN(n);
        System.out.println("XOR From 1 to " + n + " is : " + result);
    }
}
```

Completed

Attendance Code: 98A54F0E