ICP - 2024

MINOR PROJECT - 1

NAME - UDBHAV ASHISH ARYA
REGD NO. - 2341001108
SECTION - 23413C2

PROBLEM1_ROTATERIGHT

```java
public static void rotateRightBy2Bits(int[] s) {
        int n = s.length;
        int d = 2;
        int[] last2bits = new int[n];

        for (int i = 0; i < n-1; i++) {
            last2bits[i+1] = s[i] << (Integer.SIZE - d);
        }
        last2bits[0] = s[n-1] << (Integer.SIZE - d);

        for (int i = 0; i < n; i++) {
            s[i] = s[i] >> 2 | last2bits[i];
        }
    }
```

Udbhav227

1. Initialization:

- int n = s.length; (n is the length of the array s)
- int d = 2; (d is the number of bits to rotate each element to the right)
- int[] last2bits = new int[n]; (An array to store the last 2 bits of each element, which will be used later.)

2. Calculating last 2 bits:

```java
for (int i = 0; i < n-1; i++) {
    last2bits[i+1] = s[i] << (Integer.SIZE - d);
}
last2bits[0] = s[n-1] << (Integer.SIZE - d);
```

Udbhav227

- The first loop (from i = 0 to i < n−1) shifts each element in s[] to the left by (Integer.SIZE − d that is 32 − 2 = 30) bits and stores the result in the last2bits[] array. This effectively extracts the last 2 bits of each element and shifts them to the leftmost positions in the new array.

- The last element is handled separately in the loop, ensuring that the last 2 bits of the last element are stored in last2bits[0].

3. Rotating each element:

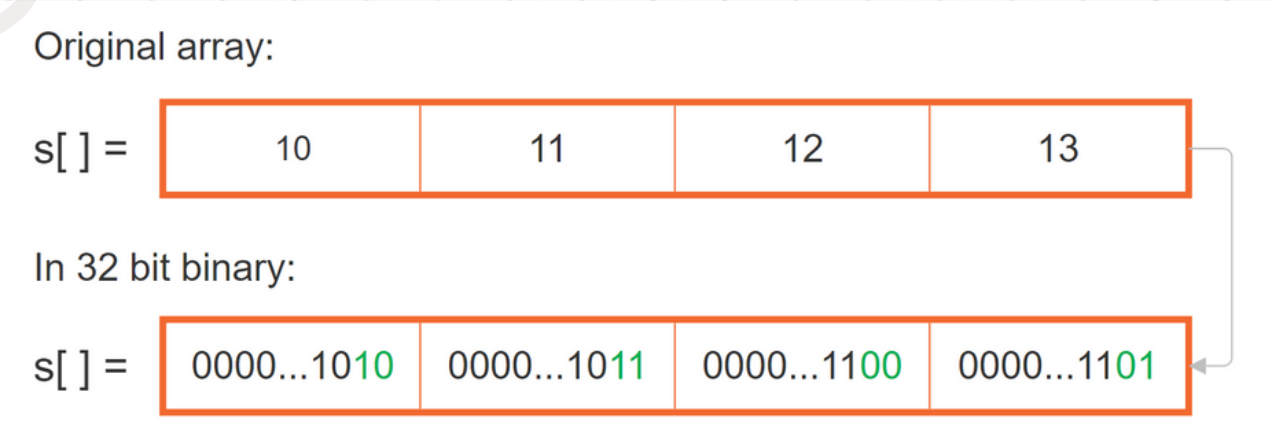
```
for (int i = 0; i < n; i++) {
    s[i] = s[i] >> 2 | last2bits[i];
}
```

Udbhav227

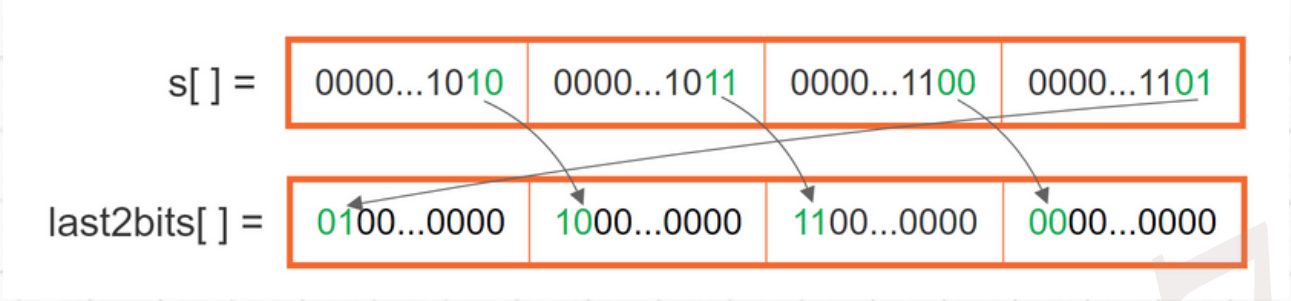
- The second loop (from i = 0 to i < n) rotates each element in the original array s[] to the right by 2 bits and sets the last 2 bits using the values stored in the last2bits array.
- This is achieved by shifting the original bits to the right by 2 (s[i] >> 2) and then combining it with the last 2 bits of the corresponding element from last2bits using the bitwise OR (|) operation.

Example

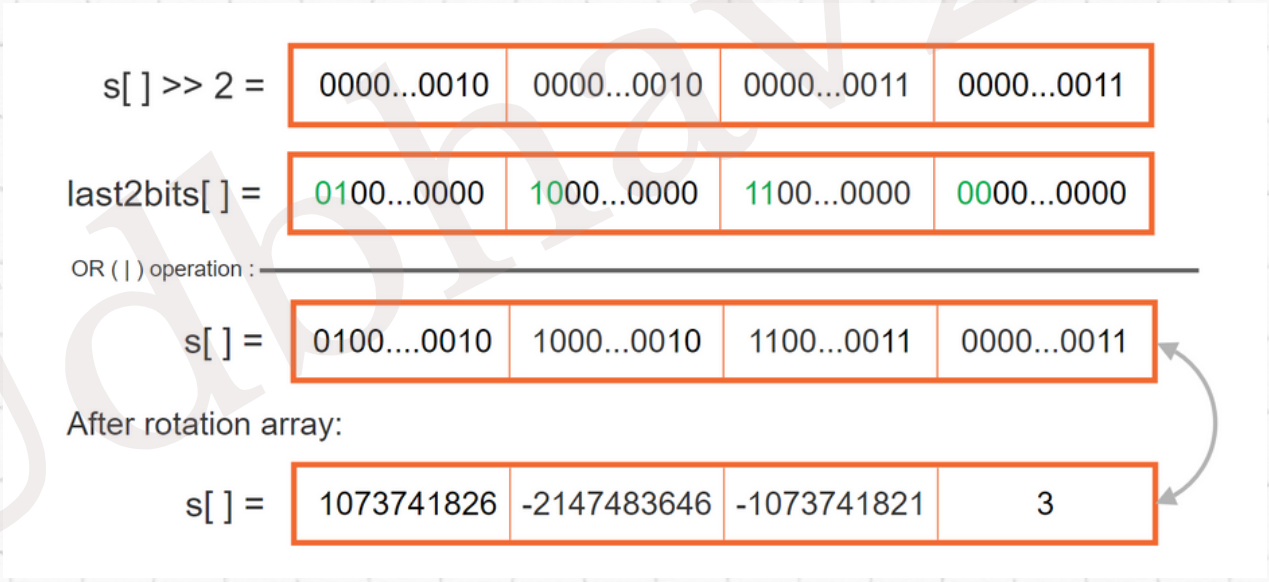
Let's see what happens with the input array {10, 11, 12, 13}



From (i = 0 to i < n) left shift 30 bits (Integer.SIZE - 2) of s[i] and store the last 2 bit of each element in last2bit[i + 1] . The last element is handled separately in the loop, ensuring that the last 2 bits of the last element are stored in last2bits[0]. (where Integer.SIZE = 32 bits)



Now shift the original bits to the right by 2 (s[i] >> 2) and then combining it with the last 2 bits of the corresponding element from last2bits[ ] using the bitwise OR (|) operation.



So, the original array {10, 11, 12, 13} is rotated to the right by 2 bits, resulting in the new array :
{ 1073741826, -2147483646, -1073741821, 3}.

-END OF PROGRAM 1-

PROBLEM2_CONVERTTOANYBASE

```java
public static String convertToAnyBase(int n, int b) {
        char[] code = new char[26];
        for (int i = 10; i <= 35; i++) {
            code[i - 10] = (char) ('A' + i - 10);
        }

        String result = "";
        while (n != 0) {
            int remainder = n % b;
            n = n / b;
            if (b > 10 && remainder > 9) {
                result = code[remainder - 10] + result;
            } else {
                result = remainder + result;
            }
        }

        return result;
    }
```


This method takes two arguments, n and b. n is the decimal number to convert and b is the base to convert to.
The function first creates an array of characters called code. This array will contain the characters that will be used to represent the digits in the converted number. For bases 2 to 9, the characters 0 to 9 are used. For bases 10 to 35, the characters A to Z are used.

The function then enters a loop that continues until n is equal to 0. In each iteration of the loop, the following steps are performed:

- Calculate the remainder of n when divided by b. This is the digit that will be in the rightmost position of the converted number.
- Divide n by b. This removes the digit that was just calculated from n.
- If b is greater than 10 and the remainder is greater than 9, then use the character at remainder - 10 in the code array to represent the digit. Otherwise, use the remainder itself to represent the digit.
- Append the digit character to the result string.

After the loop exits, the result string will contain the converted number in base b.
Here are some examples of how the function works:

- convertToAnyBase(10, 2) will return "1010" because 10 in binary is 1010.
- convertToAnyBase(15, 16) will return "F" because 15 in hexadecimal is F.
- convertToAnyBase(100, 8) will return "144" because 100 in octal is 144.