Bitwise Operation

errantProgrammer

October 9, 2024

Contents

Chapter 1

Standard Template Library

Es un conjunto de clases de plantillas que nos proporciona la implementación de estructuras de datos y algoritmo. También, proporciona los iteradores y functores que facilitan algoritmos y contenedores.

Chapter 2

Bitwise Operation

2.1 AND &

P	Q	P & Q
0	0	0
0	1	0
1	0	0
1	1	1

Codigo Example

```
6       int a = 7, b = 4;
7       int p = a & b;
8       cout <<"a & b = " << p << endl; // p = 4
```

$$\begin{array}{rr} & 111_2 \\ \& & 100_2 \\ \hline & 100_2 = 4 \end{array}$$

2.2 OR |

P	Q	P |Q
0	0	0
0	1	1
1	0	1
1	1	1

Codigo Example

```
10      int c = 7, d = 4;
11      int q = c | d;
12      cout << "c | d = " << q << endl; // q = 7
13      // XOR
```

$$\begin{array}{rr} & 111_2 \\ | & 100_2 \\ \hline & 111_2 = 7 \end{array}$$

2.3 XOR ^

P	Q	P ^ Q
0	0	0
0	1	1
1	0	1
1	1	0

Codigo Example

```
6       int a = 7, b = 4;
7       int p = a & b;
8       cout <<"a & b = " << p << endl; // p = 4
```

$$\begin{array}{rr} & 111_2 \\ \hat{} & 100_2 \\ \hline & 011_2 = 3 \end{array}$$

2.4 NOT ~

El not lo que hace nos brinda el complemento a 2, del número, no confundir con el negador lógico "!".

P	~ P
0	1
1	0

Codigo Example

```
6       int a = 7, b = 4;
7       int p = a & b;
8       cout <<"a & b = " << p << endl; // p = 4
```

$$\begin{array}{rr} \sim & 1001_2 \\ \hline & 0110_2 = 6 \end{array}$$

P	Q	P ≪ Q
0	0	0
0	1	1
1	0	1
1	1	0

2.5 Left Shift ≪

Codigo Example

```
6     int a = 7, b = 4;
7     int p = a & b;
8     cout <<"a & b = " << p << endl; // p = 4
```

$$\begin{array}{rl} & 111_2 \\ \ll & 2 \\ \hline & 011_2 = 3 \end{array}$$

2.6 Right Shift ≫

P	Q	P ≫ Q
0	0	0
0	1	1
1	0	1
1	1	0

Codigo Example

```
6     int a = 7, b = 4;
7     int p = a & b;
8     cout <<"a & b = " << p << endl; // p = 4
```

$$\begin{array}{rl} & 111_2 \\ \gg & 2 \\ \hline & 011_2 = 3 \end{array}$$

2.7 Complemento a 2

El **complemento a 2** de un número es la forma de como representamos números negativos de un número, para calcular su valor existen multiples formas:

- $C_2(N) = 2^n - N$
- $C_2(N) = C_1(N) + 1$

Pero la forma más sencilla que tenemos de calcularlo, es representando el número a binario, y desde el *bit menos significativo*, avanzamos hacia la izquierda hasta encontrar el primer 1, y a partir de este, invertimos ceros y unos.

2.8 Problemas

Resolución de algunos problemas:

```cpp
/**
 * Date: 05/10/2024
 * URL: https://codeforces.com/problemset/problem/1097/B
 */

#include <bits/stdc++.h>

int main(){
    int n;
    std::cin >> n; //cantidad de angulos
    std::vector<int> angles(n);
    for (int i = 0; i < n; i++) std::cin >> angles[i];

    for (int mask = 0;mask < (1 << n);mask++){ // recorremos la mascara
        int total = 0;// suma total de los angulos
        for (int i = 0; i < n; i++){ // solo vamos a utilizar los n primeros bits
            if(mask & (1 << i)) // verificamos si el i-esimo bit esta prendido
                total += angles[i];
            else
                total -= angles[i];
        }
        if(total % 360 == 0){
            std::cout << "YES";
            return 0;//se acaba el programa
        }
    }

    std::cout << "NO";


    return 0;
}
```