All about CP

Table of Content

1 Development

This book is publicly developed on GitHub. If you find anything confusing, or you think that there is a better way to express the idea, please make a pull request.

2 FastIO

2.1 The Magic Line

```
1 cin.tie(0)->sync_with_stdio(0);
```

3 Bit Manipulation

3.1 Binary Literal in C++

You can represent binary numbers in C++ by using the `0b` prefix. For example: 13 in binary is `0b1101`.

```
assert(13 == 0b1101);
```

3.2 Signed and Unsigned Integers

Positive integers (both signed and unsigned) are just represented with their binary digits, and negative signed numbers (which can be positive and negative) are usually represented with the Two's complement [1]

```
cout<<bitset<3>(5)<<"\n";
// Output: 101
cout<<bitset<32>(-1)<<"\n";
// Output: 11111111111111111111111111111111
```

3.3 XOR

3.3.1 How to Visualize XOR

Method-1: Non-equivalence Operator

$A \oplus B$ is $true$ if truth value of A and B are different.
The following function uses this algorithm:

```
bool xor(bool a, bool b) {
  if (a!=b)
    return true;
  return false;
}
```

Method-2: Programmable Inverter

Think of XOR as a machine with an on/off button, that takes one bit as input and one bit as output. If the machine is on, the ouput bit will be inverse of input bit, otherwise, the output bit will be the same as input bit.
In $A \oplus B$, one bit decides if the other should be flipped.
The following function uses this algorithm:

```
bool xor(bool a, bool b) {
  if (a)
    return !b;
  return b;
}
```

3.4 Common Bit Operations and Checks

3.4.1 Parity Check

If `n` is an integer(positive or negative) then `n&1` represent parity of `n`. It is similar to `n%2` but better, because unlike `n%2`, `n&1` works for both positive and negative numbers.

```
int n;
n=5;
assert((n&1) == 1);
assert((n%2) == 1);
n=-5;
assert((n&1) == 1);
assert((n%2) == -1);
```

3.4.2 Left Shift as $\times 2$

`x<<y` is equivalent to $x \times 2^y$

```
assert((5<<2) == 20);
```

3.4.3 Right Shift as $\div 2$

`x>>y` is equivalent to $\lfloor \frac{x}{2^y} \rfloor$

```
1 assert((10>>2) == 2);
```

---

1. cp-algorithms - bit manipulation↩