P THINK FAST S

Competitive Programming

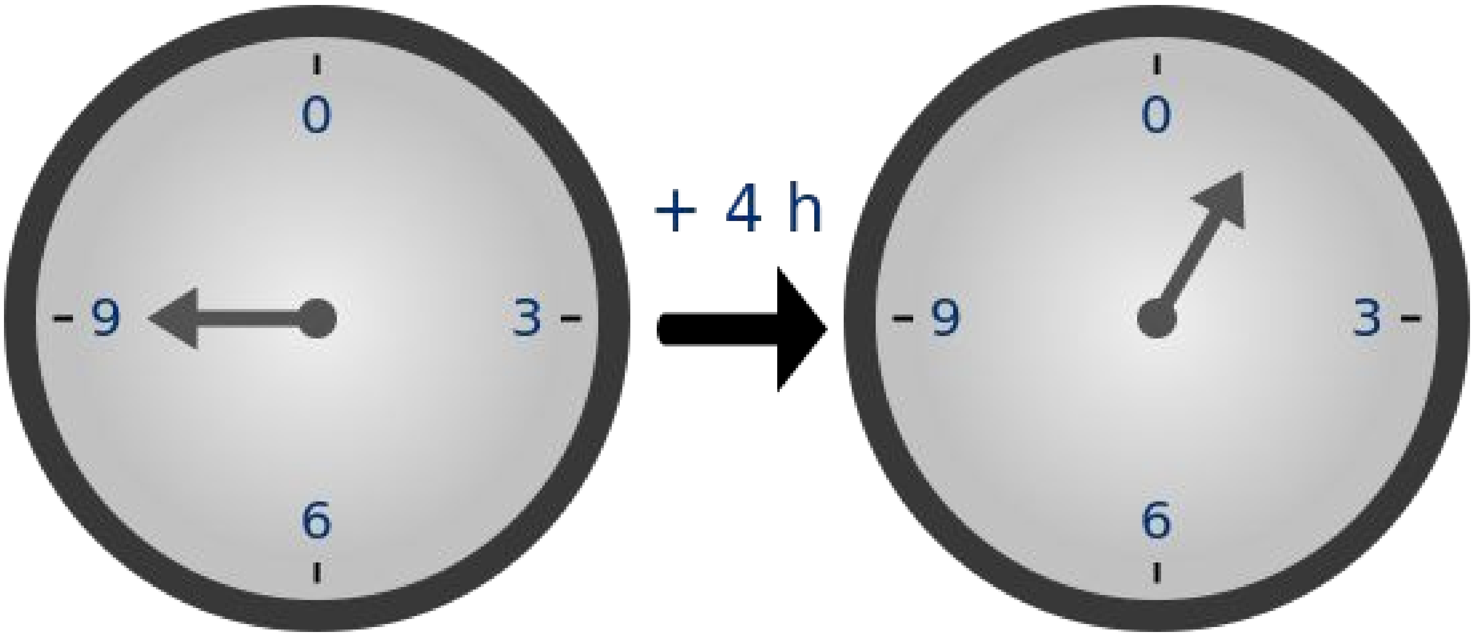
From Problem 2 Solution in O(1)

Number Theory

Modular Arithmetic

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

12-hour Clock **Cycle**



- If it is 9 now, what time:
- after 4 h? 1
- after 16 (4 + 12) h? 1
- after 17 (5 + 12) h? 2
- after 29 (5 + 2*12) h? 2
- before 24 (2*12) h? 9
- before 25 (1+2*12) h? 8

- Facts:
- N = x + m * 12
  - N is number, x < 12, m >= 0
- Every multiple of 12 is useless
- What is less than 12 affects us.
- We can go forward or backward

Modulo (modulus) operation

- a **modulo** n = finds the remainder of after division by n: In C++, operator is **%**
- let a = 27, n = 12, then r = a % n?
- 27 / 12 = **(3+2*12)**/12 = 3/12 + 2 = 2.25
  - q (quotient), the Integer division part is 2
  - **r** (remainder) of division is 3
  - r = 27 % 12 = 3 => Remainder from division
- a = nq + r (q multiple of n + r [< n])
- % operator is finally: $r = a - n\left\lfloor \frac{a}{n} \right\rfloor$
- |r| < n

Back to the clock

- If it is 9 now, what time:
- after 4 h? => 9 + 4 = 13 h => 13 % 12 = 1
- after 16 (4 + 12) h? => 9 + 16 = 25 h => 25 % 12 = 1
- after 17 (5 + 12) h? => 9 + 17 = 26 h => 26 % 12 = 2
- after 29 (5 + 2*12) h? => 9 + 29 = 38 h => 38 % 12 = 2
- before 24 (2*12) h? 9 => 9 - 24 = -15h => hmm
  - 15 % 12 = 3….hmm, we are sure results should be 9 too
  - **+ve is not as same -ve**
  - Fact: r = a % n = (a+qn)%n => I.e. adding multiplier on doesn't affect results
  - -15 + 12 = -3, still negative, **add another 12**
  - -3 + 12 = 9 … Good! Done
  - In C++: -15h % 12 = -3, so you need to add 12 **only once**
- What time before 25 (1+2*12) h? 9 - 25 = -16 => -16 % 12 = -4 [in C++]
  - Add 1 cycle to make it positive: -4 + 12 = 8 hours
- In C++: for any r => (a % n + a) % n is always positive

modulus is expensive

- % and / are time expensive operations
- If you can avoid them, avoid them
- One scenario, when you are sure results can be fixed with little +/- of mod value
  - we can directly do: a = (a%n + n)%n
  - 1 addition and 2 mod operations
  - maybe we can fix results with e.g. 2 comparison/add
  - while(a >= n)   a -= n;
  - while(a < 0)   a += n;

Facts

- To get modulus => add/remove cycles of n till
  - 0 <= r <= n-1
  - 27 % 12 => 15 % 12 => 3 % 12 = 3
  - -15 % 12 => -3 % 12 => 9 % 12
- |a%n| has n-1: 0, 1, ...n-1
- In C++:
  - a%3 => -2, -1, 0, 1, 2 [for a -ve or +ve]
  - a % n (for +ve) or (a % n + n) % n (generally)
- a%n = 0 => a divisible by n
- If a%n == b%n => (a-b)%n = 0
- largest n such that a%n = b%n is n = b-a

Facts

- (a % n) %n = a % n
- (n ^ x) % n = 0 for any x >= 0
- -a%n != a%n => (3 %12 = 3 vs -3 %12 = 9)
- ((-a%n)+(a%n))%n = 0
- (a+b) % n = (a%n + b%n) % n
- (a+b+c+d)%n?
  - You can take mod of every one and sum
  - or ((((a%n+(b%n))%n+c%n)%n+d%n)%n
- x % (a+b) != x % a + x %b
- x%10 [the last digit]. x/10 [remove last digit]

Facts

- (a*b) % n = (a%n * b%n) % n
- (a^b) % n = ((a%n)^b) % n
- (a^b) % n => assume b even and x = b/2
  - ((a^x) % n * (a^x) % n)%n
- (1/a) % n ? modular **multiplicative inverse**
- ((a*b) % n * (1/a)%n) %n = b % n
- a % (2^n) = a & (n-1) => E.g. a%4 = a&3
- a % 0 is undefined
- When -ve result => result = (result + n)%n

Facts

- What is wrong here?

```
bool is_odd(int n) {
    return n % 2 == 1;
}
```

```
bool is_odd(int n) {
    return n % 2 == 1 || n % 2 == -1;
}
```

```
bool is_odd(int n) {
    return n % 2 != 0;
}
```

Cycling examples

- A machine keeps generating the sequence 5 2 7 1 for infinity..what is its value after 10^12 steps? 5 2 7 1 **5 2 7 1** 5 2 7 1 ….
  - After 0 steps => 5 After 3 steps => 5
  - After 4 steps => 5 After 5 steps => 2
  - It keep cycling. Remove all cycling at once: 10^12 % 4
  - Rings, Cycles, ...should trigger the **mod**
- Given position X in array, iterate back M steps? We may cycle and back to array end

Why modulus?

- Either cycle (ring) is nature of the problem
  - 12-hour clock, week is 7 days, year is 356/366 days
- Encryption Algorithms, Pseudo-random Generators
- For fun, e.g. what is the last digit of 2^100?
- In competitions, final result is too big, but we want to avoid using big integers. Using mode, truncate results
- You are sure final results <= n, but intermediate results overflow. Take intermediate % x (x > n)
  - 1001 - 1111 + 153 = **43** ...let x = 44
  - ((1001%44 + ((-1111%44)+44)%44 + 153%44)%44
  - (33+33+21)%44 = 87%44 = 43

UVA 408, 10006, CF447-A, CF284-A, 332A, 11155, 132A, 374, 128,

SRM 144-D2-1

CF476-D2-C

https://www.hackerrank.com/domains/mathematics/fundamentals

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً