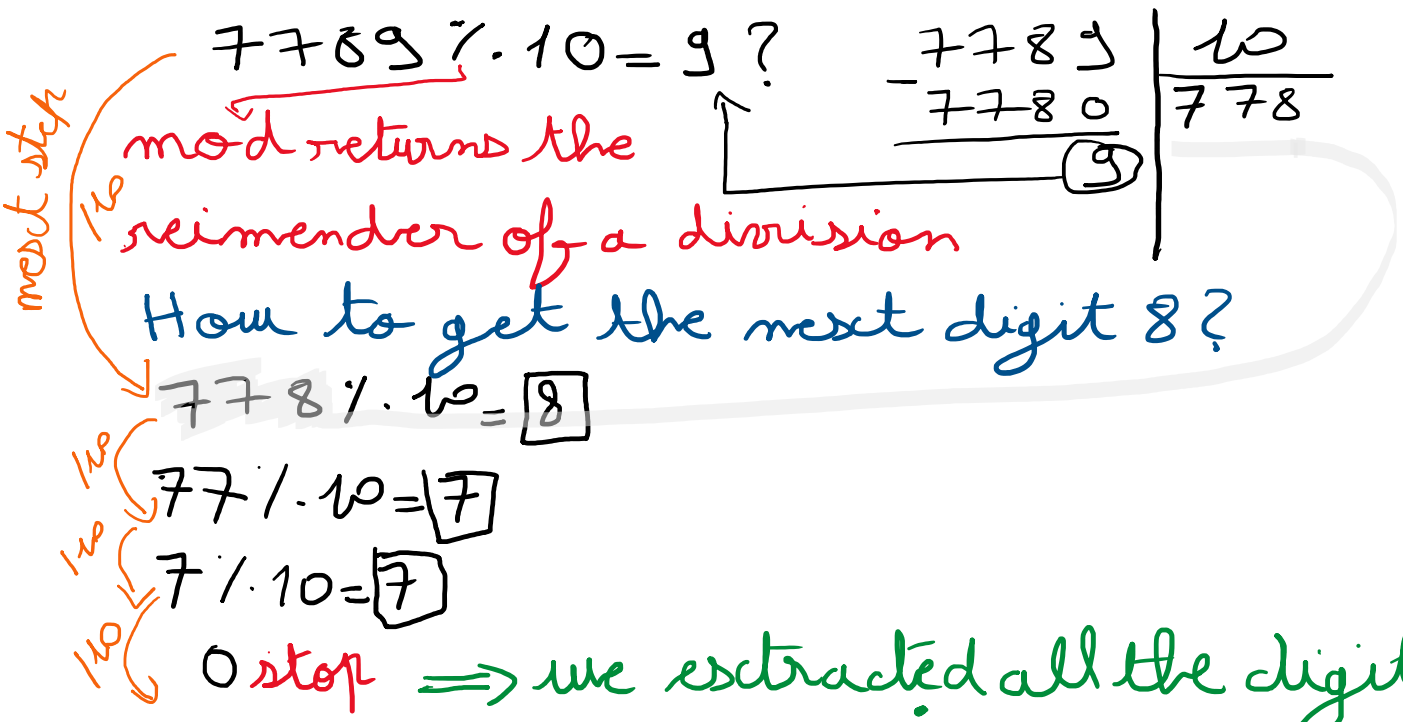
Digits extraction

Saturday, December 21, 2024 12:51 PM



Pseudo Code :

```
N: the number
while( n > 0){
    lastDigit = n % 10
    n = n / 10
}
```

Digits count

Saturday, December 21, 2024 4:02 PM

Count the number of digits

method 1

```
count := 0
while( n > 0){
    count++
    n = n/10
}
```

number of digits = the number of division by 10

$= \log_{10}(n) + 1$

$= \log_{10}(7789) + 1$

$= 3.89 + 1 = 4,89$

$\Rightarrow$ 4 digits TC: $O(\log_{10}(n))$

method 2

```
int Count( int n){
    return (int)( log10(n) + 1 )
}
```

TC: $O(1)$

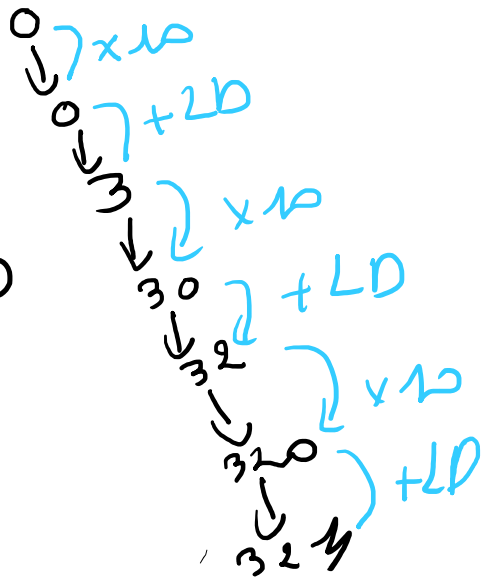
(LeetCode) Reverse & Palindrome Number

Saturday, December 21, 2024 4:32 PM

Examples:

123 → 321 / 120 → 21

```
int getReverse number(int n){
    int result = 0
    while(n>0){
      lastDigit = n % 10
      result = result x10 + LD
      n = n / 10
    }
    return result}
```



Palindrome number

It's a number that reads the same as backward as forward, examples: 121, 1331, 4554

```
bool isPalindrome(int n){
    int copy = n
    pn = getReverseNumber(n)
    if(pn == copy){
        return true
    }
    return false
}
```

(LeetCode) Armstrong Number

Sunday, December 22, 2024 12:01 PM

It's a number that is equal to the sum of its digits each raised to the power of the number of digits. example: $153 = 1^3 + 5^3 + 3^3$

Solution:

Loop over digits and calculate the sum of the digit power l

```
bool isArmstrongNumber(int n){
    int l = to_string(n).length()
    int nCopy = n
    int sum = 0
    while(n > 0){
        sum = pow(n%10, l)
        n = n/10
    }
    if(nCopy == sum){
        return true
    }
    return false
}
```

Get All Divisors

Sunday, December 22, 2024 2:38 PM

Problem statement:

Given an integer N, return all divisors of N.

Example: 12 -> 1, 2, 3, 4, 6, 12

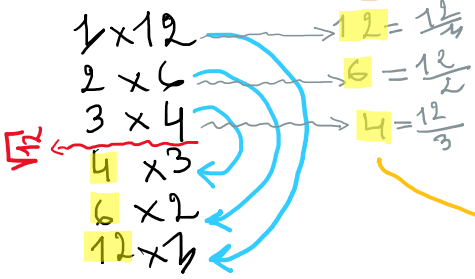
Problem Analysis:

divisors ∈ [1, N]

loop from 1 to N, if i % N == 0 -> divisor

↳ T.C = O(N), (too much) (brute force)

Mathematical observation:



So, we loop to N/2 only, if i is a divisor, $\frac{N}{i}$ is also

↳ T.C = O(√n)

Solution

```
void printAllDivisors(int n){
    for(i=1; i<=√n; i++){
        if(n%i==0){
            print(i)
            if(n/i != i){ print(i)}
        }
    }
}
```

to avoid double counting in case of perfect square, 9, 16, 36 ....

Prime Number

Sunday, December 22, 2024 7:25 PM

Problem statement :

A prime number is a number that is only divisible by 1 and itself. Example: 2

Problem analysis :

The total number of divisors is exactly 2 : so we loop over divisors, if count > 2 return false;

```
bool isPrimeNumber(const int n) {
    if (n <= 1) return false; // 0 and 1 are not prime numbers

    int count = 0;
    for (int i = 1; i <= std::sqrt(n); i++) {
        if (n % i == 0) {
            count++;
            if (i != n / i) count++; // Count only distinct divisors
        }
        if (count > 2) {
            return false;
        }
    }
    return true;
}
```

Digit Concept - GCD

Monday, December 23, 2024 7:08 AM

Problem statement:

the greatest common divisor of any two integers is the largest number that divides both integers. example 9 and 12

factors of 9: 1, 3, 9 } factors of 12: 1, 2, 3, 4, 6, 12

GCD —> 3

Problem analysis:

brute force:

we iterate from 1 to min(N1, N2), if i divides both N1 and N2, we affect it the GCD variable

T.C = O(min(n1, n2))

Better approach:

In the previous approach, in the worst case, i iterates from 1 to min(n1, n2). if min(n1, n2) is a very large number. it will result a large number of iterations.

So, we can iterate from min(n1, n2). if a common divisor is found. we return it directly.

L> T.C: it is the same. but we execute less iterations

```
int findGCD(int n1, int n2) {
    for (int i = std::min(n1, n2); i > 0; i--) { // std::min(a, b)): O(1)
        if (n1 % i == 0 && n2 % i == 0) {
            return i;
        }
    }
    return 1; // In case no other divisor is found
}
```