1

Complexity Analysis & Elementary Number Theory

Correct Solution

Efficient Solution

Problem-Solving

Verdict
Accepted

Performance of an algorithm

- Execution Time
- Memory Usage

- How are they being calculated?

A. Vasya And Password

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Time limit exceeded on test 21

Memory limit exceeded on test 68

01

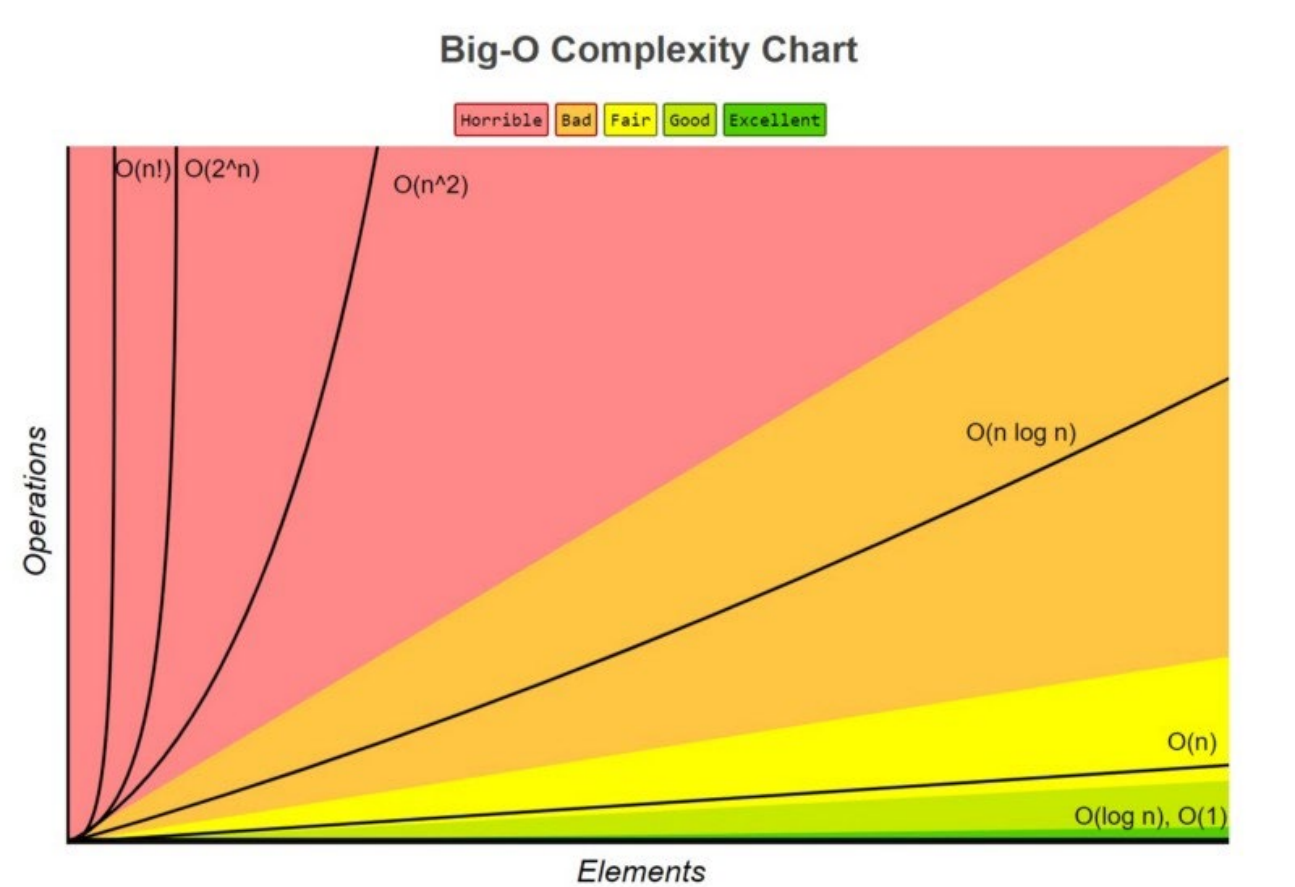
Complexity Analysis

Time Complexity

Time Complexity General Points

- Most online judges run between 10^7 to 10^8 operations/second
- We describe the time execution of our code in Big O notation.
- Big O notation is used for comparing algorithms together.
- In Big O we ignore constants and care about the following order:

Complexity Order



	Cost Time require for line (Units)	Repeatation No. of Times Executed	Total Total Time required in worst case
`int sumOfList( int A[ ], int n)`			
`{`			
`int sum = 0, i;`	1	1	1
`for(i = 0; i < n; i++)`	1 + 1 + 1	1 + (n+1) + n	2n + 2
`sum = sum + A[i];`	2	n	2n
`return sum;`	1	1	1
`}`			
			4n + 4 Total Time required

Examples

```
int main() {
    int n;
    cin>>n;

    for (int i = 0; i < n; i++) {
        // your code
    }

    return 0;
}
```

Examples 1.2

```
int main() {
    int n;
    cin>>n;
    for (int i = 0; i < n; i++)
        // your code
    for (int j = 0; j < n; j++)
        // your code

    return 0;
}
```

Examples 2

```
int n, m;
cin>>n>>m;

for (int i = 0; i < n; i++) {
    // your code
    for (int j = 0; j < m; j++) {
        // your code
    }
}
```

Can You prove the addition rule?

$$\frac{n(n+1)}{2}$$

Examples 3

```
int main() {
    int n, m;
    cin>>n>>m;

    for (int i = 1; i < n; i*=2) {
        // your code
    }

    return 0;
}
```

Can anyone Prove it?

$$i < n$$

Let i=k, each step we multiply by 2: $2^k < n$

Take log to the base 2 for each

$$log\ 2^k < \log(n)$$

$$k.log2 < \log(n)$$

$$k \geq \log(n)$$

$$k = \log(n)$$

Examples 3.2

```
int main() {
    int n;
    cin>>n;
    while(n){
        cout<<n%2;
        n/=2;
    }
    return 0;
}
```

Examples 4

```
int main() {
    int n, m;
    cin>>n>>m;

    for (int i = 0; i < n; i++) {
        // your code
        for (int j = m; j > 0; j/=2) {
            // your code
        }
    }
```

Examples 5

```
string s;
cin>>s;
for (int i = 0; i < s.size(); i++) {
    // your code
    for (int j = i + 1; j < s.size(); j++) {
        // your code
        for (int k = j + 1; k < s.size(); k++) {
            // your code
        }
    }
}
```

Examples 6

```
int n;
cin>>n;
for(int i=0;i<n;i++) {
    // your O(1) code
    for (int j = 0; j < n; j++) {
        // your O(1) code
        for (int k = 0; k < 1000; k++) {
            // your O(1) code
        }
    }
}
```

Examples 7

```
int main() {
    int n;
    cin>>n;
    for (int i = 0; i < sqrt(n); i++) {
        // your code
        for (int j = 0; j*j < n; j++) {
            // your code
        }
    }

    return 0;
}
```

Examples 8 (Teaser)

```
int main() {
    int n;
    cin>>n;
    for (int i = 1; i < n; i*=2) {
        // your O(1) code
        for (int j = 0; j < n; j+=i) {
            // your O(1) code
        }
    }
```

$\boxed{O(n)}$

$O(n^2)$

$O(n.\log(n))$

$O(\log(n))$

Examples 9

$$T(n) = 3n^2 + 5n.\log(n)$$

$$T(n) = 3^n + \sqrt{n} + 40$$

$$T(n) = n! + n^3 + 2n\log(n)^2$$

01

Complexity Analysis

Space Complexity

Examples 10

```
int main() {
    int n;                          //O(1)
    cin>>n;
    int arr[n];                     // O(n)
    int segTree[n][1<<2];           // O(4n)
    int halfArr[n][log2(n)];        // O(nlog(n))

    return 0;
}
```

C++ Programmers Tricks

```
int main() {
        // at the beginning of the main.

        ios_base::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);

        // your code

        return 0;
}
```

Practice with me

https://codeforces.com/group/Qwwauwf6dV/contest/413528

- Time and memory efficient: https://ideone.com/hCtOWN
- Using Prefix Sum: https://ideone.com/jGim 2J
- Brute Forcing the solution: https://ideone.com/H 8xyxY

02

Elementary Number Theory

Contents

Number Theory is

a branch of mathematics that deals with the properties and relationships of numbers, especially positive integers.

Even and Odd Numbers

Even

An integer n is even, if and only if, n is twice some integer.

e.g. n=2*k

Odd

An integer n is odd, if and only if, n is twice some integer plus 1

e.g. n = 2k+1

Prime and Divisor Numbers

Prime

An integer n > 1 is prime, if and only if, for all positive integers r and s, if n=r ⎪ s, then r=1 or s=1

e.g. n=1*r

Divisor

An integer n is divisible by an integer d ($d \neq 0$), if and only if, n equals d times of some integer k

e.g. n = d*k

getDivisors

```
void getDivisors(int n){
    for(int i=1;i<=n;i++)
        if(n%i==0)
            cout<<i<<" ";            // 1 2 3 4 6 12
    cout<<endl;
}
```

getDivisorsSqrt

```
void getDivisorsSqrt(int n){
    for(int i=1;i*i<=n;i++)
        if(n%i==0)
            cout<<i<<" "<<n/i<<" "; // 1 12 2 6 3 4
    cout<<endl;
}
```

Did you notice something wrong?

If n = 9, 3 will be printed twice.

Practice

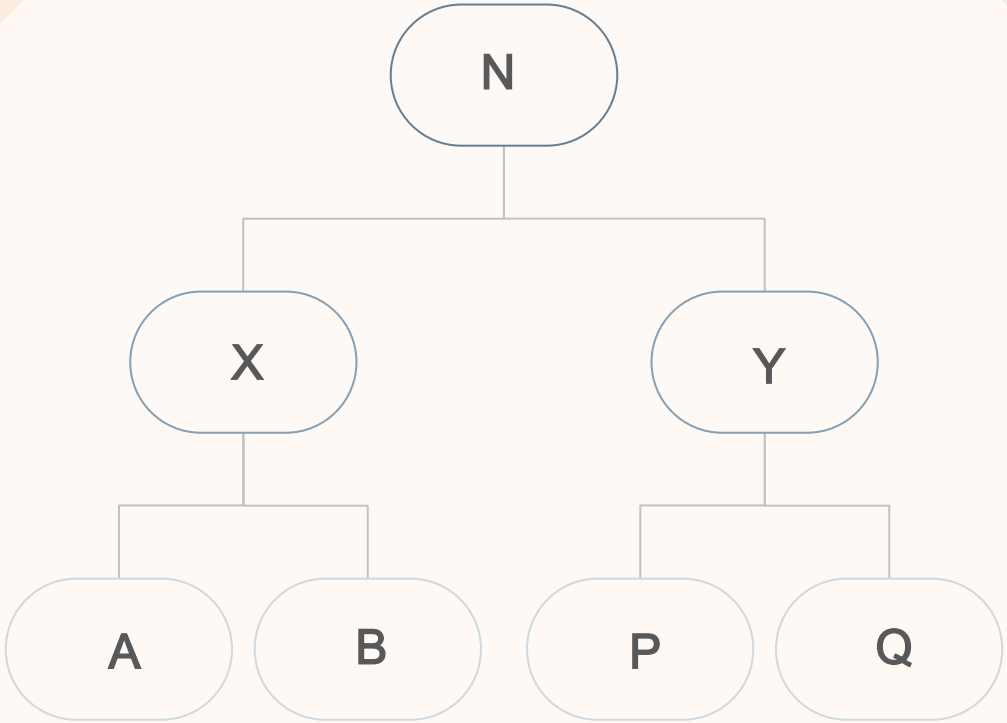
https://codeforces.com/contest/762/problem/A

Solution:

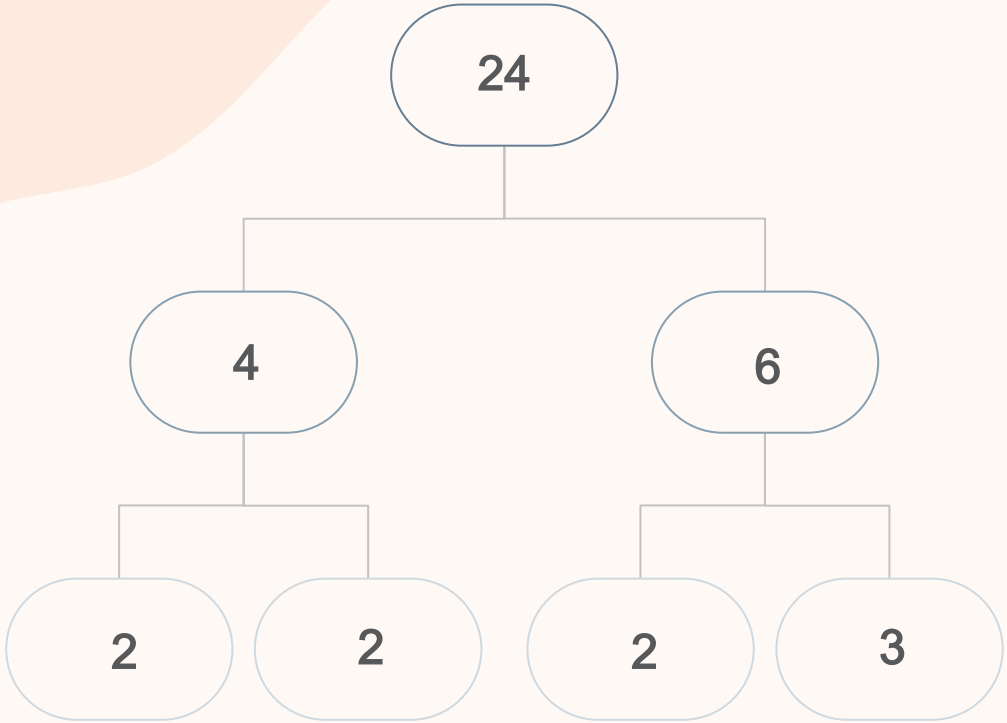
https://ideone.com/mlHxlx

me Factorization

Prime Factorization



Prime Factorization



$N = 2^3 \lceil 3^1 = 24$

Division Algorithm

$$N = P_1^a \times P_2^b \times P_3^c \times \ldots$$

N is the product of all of its prime numbers raised to a certain power

```
void prime_factors_sqrt(int n){                    // n = 12
    for (int i=2;i*i<=n;i++)
        if(n%i==0)
            while(n%i==0)
                cout<<i<<" ", n/= i;               // 2 2 3
    if(n>1) cout<<n<<endl;
}
```

Greatest Common Divisor (GCD)

We can get all divisors and easily take all common divisors and multiply them together. Or…

```
int gcd (int a, int b) {
    while (b) {
        a %= b;
        swap(a, b);
    }
    return a;
}
```

Use __gcd(a, b); built-in function.

Depending on your compiler it could be gcd(a, b); too

Least/Lowest Common Multiple (LCM)

A method to find the smallest possible multiple of two or more numbers.

```
int lmc (int a, int b) {
    return a*b/__gcd(a, b);
}
```

Better return a/__gcd(a, b)*b; to avoid overflow.

But its still preferable to use long long and focus on constrains.

Practice

https://codeforces.com/contest/1764/problem/B
Solution: **https://ideone.com/PQhnLD**

https://www.youtube.com/watch?v=gN-nIXpl2rQ&t=2073s
(first 2 hours, third hour is off-topic)

Factorization:
https://www.youtube.com/watch?v=PTxi1Uh6tks
(1:04:40 -> 1:57:30)

Resources

Thanks

Do you have any questions?