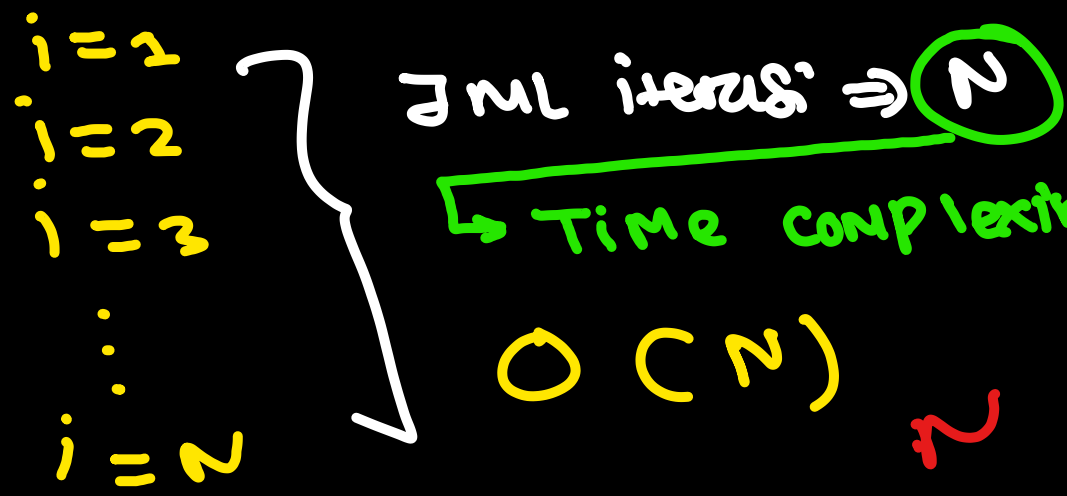
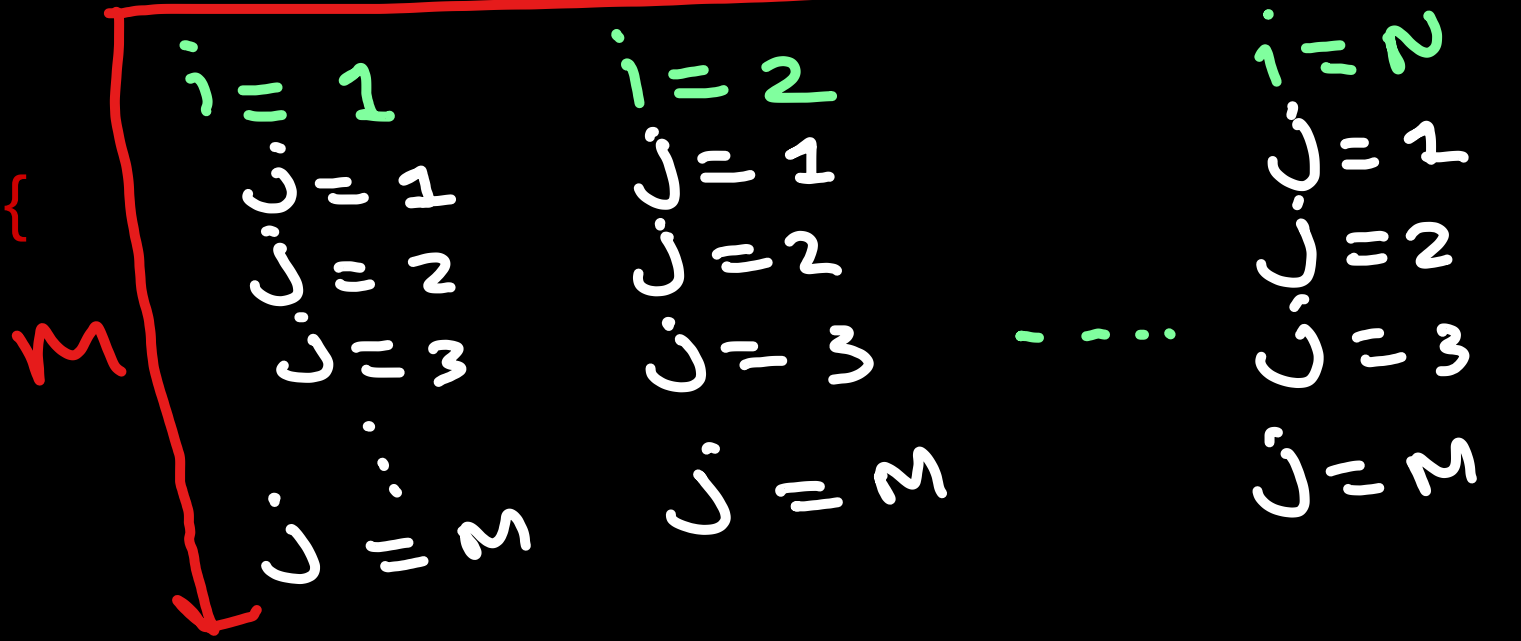
1) Linear

```
for(int i=1; i<=N; i++){
    sum += i;
}
```




```
for(int i = 1; i <= N; i++){
    for(int j = 1; j <= M ; j++){
        cout<<"Hi"<<endl;
    }
}
```

O(NM)



Setiap i akan for loop sebanyak M
Banyak i nya sebanyak N,

Jumlah iterasi N*M

2) Konstan → O(1) → tdk ada iterasi / Rekursif

```
int N;
cin>>N;
int sum = N * (N + 1) / 2;
cout<<sum<<endl;
```

} O(1)

3) Polinomial

```
for(int i = 1; i<=N; i++){
   for(int j = 1; j<=N; j++){

  }
}
```

} O(N²)

```
for(int i = 1; i<=N; i++){

}
```

O(N)

```
for(int j = 1; j<=M; j++){

}
```

O(M)

O(N+M)

Linear

```
N = dari input
while(N--){

}
```

O(N)

while (...)

Selama = true / 1

N
N-1
N-2
⋮
N-N

1
0

↳ = 0 / False → stop

O(N+1) = O(N)

end
stop

0 = false

4) Logaritmik

```
int N;
cin>>N;
while(N /= 2){
  cout<<"Woy"<<endl;
}
```

sebanyak x

$N * 2 * 2 * 2 * 2 \dots * 2 = N \cdot 2^x$

$N / 2 / 2 / 2 / 2 / \dots / 2 = \frac{N}{2^x}$

$O(C \lg N)$ sebanyak x

Jumlah iterasi = berapa kali $N/2$ sampai $N = 1$

x = jml iterasi

$$\frac{N}{2^x} = 1 \leftrightarrow N = 2^x$$

$$x = {}^2\log N$$

$$x = \frac{\lg N}{\lg 2}$$

$2^3 = 8 \leftrightarrow 2^{\log 8} = 3$

```
int sum(int n){
    if(n == 1){
        return 1;
    }else{
        return sum(n - 1) + n;
    }
}
```

Jml iterasi:
↳ brp kali function dipanggil

Sum(3) = 3 + Sum(2)

Sum(2) = 2 + sum(1) → Base case
↳ end of iteration

Sum(3) → Sum(2) → Sum(1)
1 2 3 → 3 kali iterasi

Sum(N) → Sum(N-1) → Sum(N-2) → ---- → Sum(1)
1 2 3 N

3 kali iterasinya ⇒ N } O(N) sum(N) → base case

Rules of thumb

subsitute constraint = 10^8 → 1 s

O(...)

Constraint : 1<= N <= 10^8

max

O(N) ⇒ O(10^8)

1 s

O(1) → ≤ 1 s

Constraint : 1<= N <= 10^12

max

O(N) ⇒ O(10^{12})

≥ 1 s

TL: 1s →

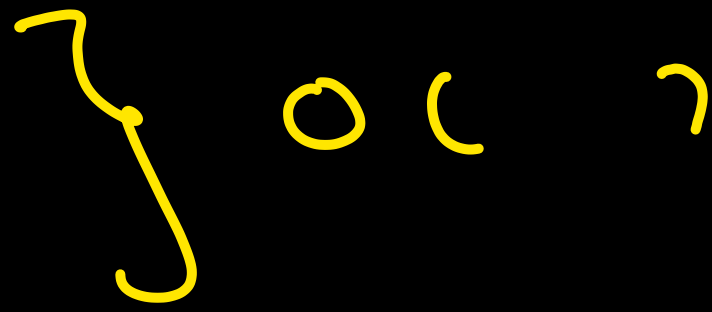
O(lg N)

2 lg 10^{12} →

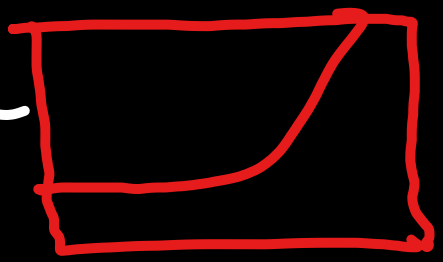
O(lg 10^{12}) = 12 < 10^8 = < 1s

TLE

```
long long f(long long n){
    if(n == 1 || n == 2) return 1;
    return f(n - 1) + f(n - 2);
}
```



5) Eksponential



f(3)

f(4) f(3)

f(3) f(2) f(2) f(1)

f(2) f(1) f(2) f(1)

2^n

→ g. iterasi

f(n) → 1 → 2^0

f(n-1) f(n-2) 2 2^1

f(n-2) f(n-3) f(n-3) f(n-4) 4 2^2

f(n-3) f(n-4) - - - - - - - 8 2^3

f(2) f(1) → 2^n ⇒ O(2^N)

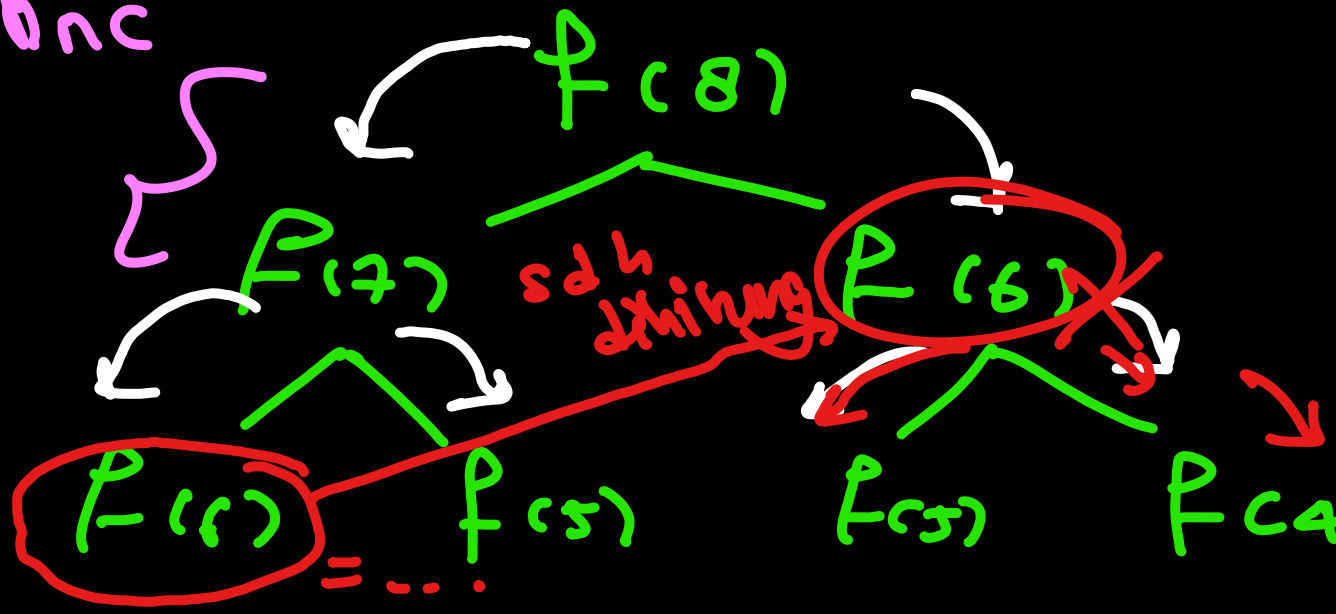
2^1 $1 \le N \le 1000$ → $O(2^N)$

$2^{1000} > 10^8$ TLE

↳ ≥ 1s

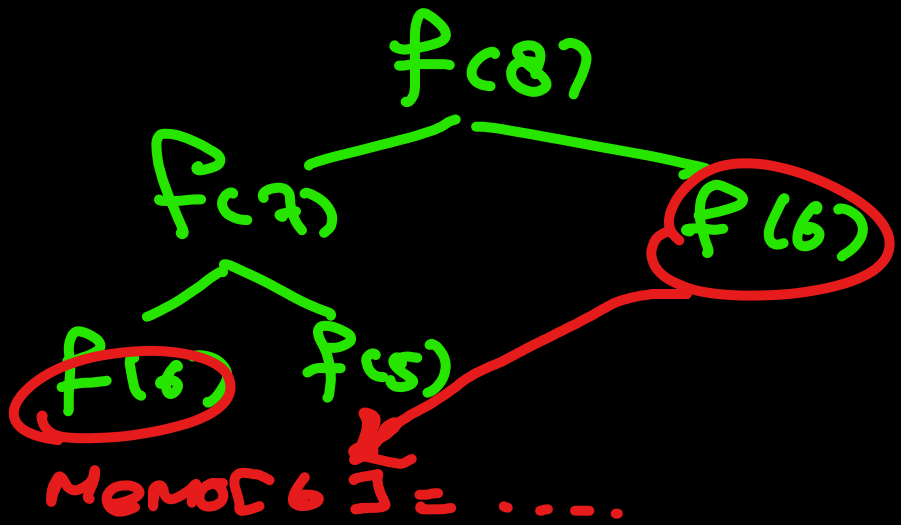
W.O Memoization

Dnc

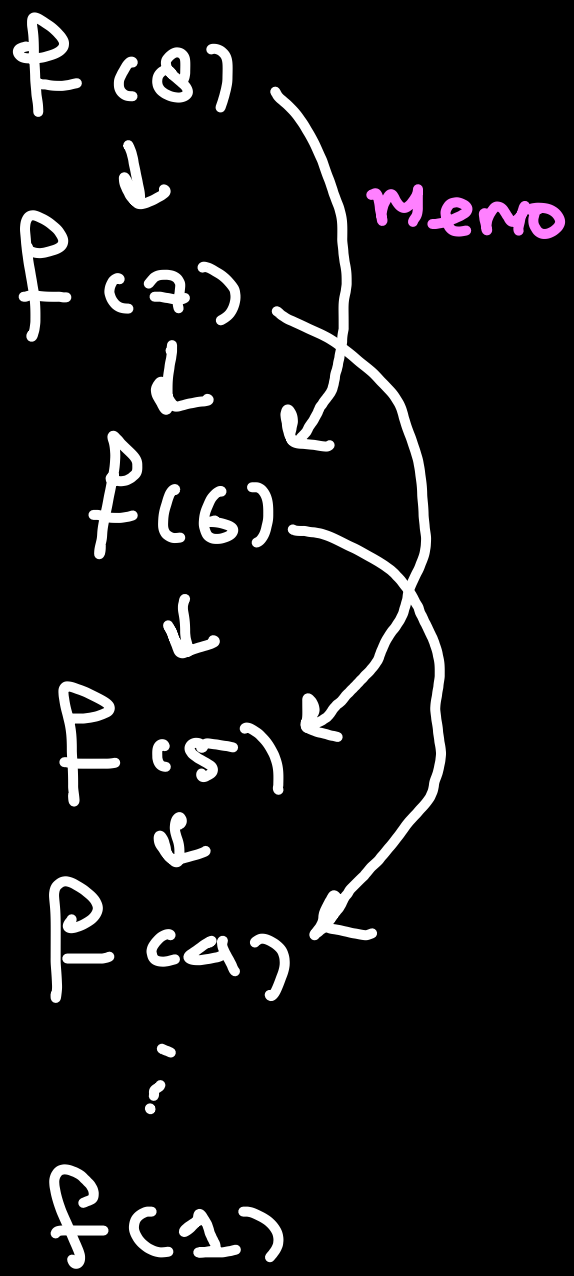


Dynamic Programming

W. Memoization

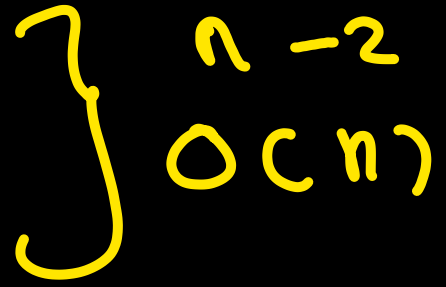


Divide and Conquer



O(N)

```
bool isPrime(int n){
    if(n == 1) return false;
    // Hanya habis dibagi 1 dan dirinya sendiri
    for(int i = 2; i<= n - 1; i++){
        if(n%i == 0) return false;
    }
    return true;
}
```



For Q → O(Q)

isprime(N) → O(N)

} O(Q * N)

$1 \le Q \le 1000$, $1 \le N \le 10^6$

$O(QN) \Rightarrow O(10^3 \cdot 10^6) \rightarrow 10^9 > 10^8$

> 1s

N

$q \rightarrow \underline{\underline{3^2}}$

$N \rightarrow q^2$

$q \rightarrow \sqrt{N}$

$q \rightarrow 3^3$

$q \rightarrow$

42 → 2 · 3 · 7

$7^2 \rightarrow 42$

$1 \rightarrow \text{sqrt}(N)$

$O(Q * \text{sqrt}(N))$

$O(1000 * \sqrt{10^6})$

$O(10^3 * 10^3)$

$= 10^6 < 10^8$

$\leq 1s$

110

5 and 3

5 → 101

3 → 011

```
101
011
---  and
001
```

(binary) 001 → 1 (int)

p	xor	q	=	(p != q)
1		0	=	1
0		1	=	1
1		1	=	0
0		0	=	0

Kehica p != q → p xor q = 0

(1 xor 2) * (1 xor 3) * (...) * (1999 xor 2000)

> 2000

↑ 0

* (1 xor 1)

$$1 + 2 + 3 + \ldots + N = \ldots ?$$

* Algoritma 1 ⇒ for loop
* Algoritma 2 ⇒ Rekursif → gauss summation
* Algoritma 3 ⇒ $\frac{n * (n+1)}{2}$