# Time Complexity

Mohammed Rizin
Unemployed

March 20, 2025

# 1 Time Complexity

## 1.1 Simple For Loop

**Algorithm 1** Simple for loop

**for** $i \leftarrow 0$ **to** $n - 1$ **do**
    ....$STMT$....
**end for**

Time complexity: $O(n)$.

## 1.2 Simple Reverse Loop

**Algorithm 2** Simple reverse loop

**for** $i \leftarrow n$ **down to** 1 **do**
    ....$STMT$....
**end for**

Time complexity: $O(n)$.

## 1.3 For Loop with Step

**Algorithm 3** Simple for loop with step

**for** $i \leftarrow 0$ **to** $n - 1$ **step** 2 **do**
    ....$STMT$....
**end for**

Time complexity: $O(n)$ (as $n/2$ is asymptotically $O(n)$).

## 1.4 Nested Loops

**Algorithm 4** Nested loops

**for** $i \leftarrow 0$ **to** $n - 1$ **do**
    **for** $j \leftarrow 0$ **to** $i - 1$ **do**
        ....$STMT$....
    **end for**
**end for**

Time complexity: $O(n^2)$.

| $i$ | $j$ | STMT | Total STMT | Total Time | Time Complexity |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 2 | 3 | 3 |
| 1 | 1 | 1 | 3 | 6 | 6 |
| 2 | 0 | 1 | 4 | 10 | 10 |
| 2 | 1 | 1 | 5 | 15 | 15 |
| 2 | 2 | 1 | 6 | 21 | 21 |

The total number of executions is $n(n+1)/2$, so the time complexity is $O(n^2)$.

### 1.5 Summation Loop

**Algorithm 5** Summation loop

$p \leftarrow 0$
**for** $i \leftarrow 1$ **while** $p \leq n$ **do**
  $p \leftarrow p + i$
**end for**

| $i$ | $p$ | STMT | Total STMT | Total Time | Time Complexity |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 1 | 2 | 3 | 3 |
| 3 | 6 | 1 | 3 | 6 | 6 |
| 4 | 10 | 1 | 4 | 10 | 10 |
| 5 | 15 | 1 | 5 | 15 | 15 |
| 6 | 21 | 1 | 6 | 21 | 21 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $k$ | $\frac{k(k+1)}{2}$ | 1 | $k$ | $\frac{k(k+1)}{2}$ | $\frac{k(k+1)}{2}$ |

Assuming $p \leq n$:

$$\begin{aligned} p &= \frac{k(k+1)}{2}, \\ \frac{k(k+1)}{2} &> n, \\ k^2 &> n, \\ k &> \sqrt{n}. \end{aligned}$$

Time complexity: $O(\sqrt{n})$.

### 1.6 Multiplication Loop

**Algorithm 6** Multiplication loop

**for** $i \leftarrow 1$ **while** $i \leq n$ **step** $2 \cdot i$ **do**
  *...Statement...*
**end for**

| step | $i$ |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| $\vdots$ | $\vdots$ |
| k | $2^k$ |

$$\begin{aligned} \text{This stops when } i > n : \\ \text{i.e. } 2^k &> n, \\ \log_2 2^k &> \log_2 n \\ k \cdot \log_2 2 &> \log_2 n \\ k &> \log_2 n \end{aligned}$$

Time complexity: $O(\log_2 n)$.
However, If you notice log function, which gives both float and integer values, then should we take floor or ceil of the floating point value.

| n = 8 |
|---|
| 1 |
| 2 |
| 4 |
| $8 < 8$ fails !! |
| So in total we could execute it for 3 iterations. $\log_2 8 = 3$ |

| n = 10 |
|---|
| 1 |
| 2 |
| 4 |
| 8 |
| $16 < 8$ fails !! |
| So in total we could execute it for 4 iterations. $\log_2 10 = 3.32$<br>$ceil(\log_2 10) = 4$ |

## 1.7 Division Loop

**Algorithm 7** For Loop with Division

**for** $i \leftarrow n$ **while** $i \geq 1$ **step** $\frac{i}{2}$ **do**
  Statement
**end for**

| step | $i$ |
|---|---|
| 1 | $n$ |
| 2 | $\frac{n}{2}$ |
| 3 | $\frac{n}{2^2}$ |
| 4 | $\frac{n}{2^3}$ |
| 5 | $\frac{n}{2^4}$ |
| $\vdots$ | $\vdots$ |
| k | $\frac{n}{2^{k-1}}$ |
| k+1 | $\frac{n}{2^k}$ |

The loop stops when:

$$\begin{aligned} \frac{n}{2^{k-1}} &< 1, \\ 2^{k-1} &> n, \\ k-1 &> \log_2 n, \\ k &> \log_2 n + 1. \end{aligned}$$

Thus, the time complexity is $O(\log_2 n)$.

## 1.8 Square Loop

**Algorithm 8** Loop till square of i is less than n

**for** $i \leftarrow 1$ **while** $i^2 < n$ **do**
    Statement
**end for**

| i | $i^2$ |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| $\vdots$ | $\vdots$ |
| k | $k^2$ |

The loop stops when:

$$\begin{aligned} k^2 &\geq n \\ k &\geq \sqrt{n} \end{aligned}$$

Thus, the time complexity is $O(\sqrt{n})$.

## 1.9 Independent loop

**Algorithm 9** Independent for loops

**for** $i \leftarrow 0$ **to** $n-1$ **do**
    $....STMT....$
**end for**
**for** $j \leftarrow 0$ **to** $n-1$ **do**
    $....STMT....$
**end for**

Since this is not nested. They are independent loops. They have $O(n)$ for both the loops adding to $T(n) = 2n$.
But **Time complexity:** $O(n)$.

### 1.10 One loop dependent on another

**Algorithm 10** Independent for loops

```
p ← 0
for i ← 0 while i < n step 2 · i do
    p++                                         ▷ p = log_2(n)
end for
for j ← 0 while j < p step 2 · j do
    ...STMT...                                  ▷ log_2 p
end for
```

Thus this algorithm takes $O(\log \log n)$

## 2 Conclusion

```
for i ← 0 to n − 1 do
    Statement                                   ▷ ► O(n)
end for

for i ← 0 to n − 1 step i + 2 do
    Statement                                   ▷ n/2 ► O(n)
end for

for i ← n to 1 do
    Statement                                   ▷ ► O(n)
end for

for i ← 0 to n − 1 step i · 2 do
    Statement                                   ▷ ► O(log_2 n)
end for

for i ← 0 to n − 1 step i · 3 do
    Statement                                   ▷ ► O(log_3 n)
end for

for i ← n to 1 step i/2 do
    Statement                                   ▷ ► O(log_2 n)
end for
```