

# How to Write and Analyze an Algorithm

Mohammed Rizin  
Unemployed

March 20, 2025

## 1 How to Write an algorithm

---

**Algorithm 1** Swapping two numbers

---

```
procedure SWAP( $a, b$ )  
     $a \rightarrow temp$   
     $b := a$   
     $b \leftarrow temp$   
end procedure
```

---

We can write algorithm in any format. The intent behind writing an algorithm is to make it easy for the programmer to quickly understand it and make it into program.

## 2 How to Analyze an algorithm?

### 2.1 Criteria for Analyzing an algorithm

1. **Time Complexity :**

This is the most important criteria for analyzing an algorithm. It is the measure of the amount of time taken by an algorithm to run as a function of the length of the input. It is denoted by  $T(n)$ , where  $n$  is the size of the input. This is not the actual time taken by the algorithm to run, but a Time Function.

2. **Space Complexity :**

Since the program going to be implemented and executed on a computer, We need to know how much space does the algorithm consumes.

3. **Network data transfer:**

But thats not all, maybe we run the algorithm on a cloud server, then we How much data is being transferred over the network. Is there any unnecessary data or large data being transferred.

4. **Power Consumption:**

If we are running the algorithm on a mobile devices(such as laptops, smartphones, etc...), then we need to know how much power does the algorithm consumes.

5. **CPU Register Usage:**

How much CPU registers are being used by the algorithm.

The criteria depends on the project you're working. Based on the project requirements, you can choose the criteria for analyzing the algorithm.

### 2.2 Example of Analyzing an algorithm

---

**Algorithm 2** Swapping two numbers

---

```
procedure SWAP( $a, b$ )  
     $temp \leftarrow a$  ▷ 1 unit of time  
     $a \leftarrow b$  ▷ 1 unit of time  
     $b \leftarrow temp$  ▷ 1 unit of time  
end procedure
```

---

### 2.2.1 Time Complexity

Every single statement in the algorithm takes one unit of time.  $f(n) = 3$   
 $O(n) = 1$

For eg,

Although  $x = 5 * a + 3 * b$  statement does have 4 operations, It only takes 1 unit of time. If you want to go in detail, you can consider the number of operations in the statement. You can also consider the machine code generated by the compiler.

Suppose you are going to your friends house and you have a car, you can just take it and go. However, if you wanna go to Mars, you need to analyze every single details on how to contact NASA, how to get into the rocket, how to survive in space, etc... There is many aspects to consider. So in this case you need to analyze the algorithm in detail. It depends on the project requirements.

### 2.2.2 Space Complexity

The total number of variables used in the algorithm.

$S(n) = 3words$  i.e.  $a, b, temp$

In this case, the space complexity is 3.

$O(n) = 1$

Even if it 3000 variables, the space complexity is 1 (i.e.  $O(n) = 1$ ).

## 3 Frequency Count Method

### 3.1 Example

---

**Algorithm 3** Sum of all elements in an array

---

```
procedure SUM( $A, n$ )  
     $S \leftarrow 0$  ▷ 1 unit of time  
    for (  $\underbrace{i = 0;}_1 \underbrace{i < n;}_{n+1} \underbrace{i++}_n$  ) do ▷  $n+1$  units of time  
         $S \leftarrow S + A[i]$   
    end for  
end procedure
```

---

#### 3.1.1 Time Complexity

If there is any statement that is executed for some number of times, then we can use the Frequency Count Method to analyze the algorithm.

Suppose we have array of size 5 i.e.  $[1, 2, 3, 4, 5]$ .

These are the values of  $i$  in the loop.

Initially  $i = 0$

After 1st iteration  $i = 1$  Since  $i$  is post incremented, the value of  $i$  is updated after the loop.

Check if  $i < n$

After 2nd iteration  $i = 2$

Check if  $i < n$

...

After 5th iteration  $i = 5$

Check if  $i \leq n$

Since the condition is checked  $n + 1$  times, the time taken by that statement is  $n + 1$  units of time.

Whatever statement inside the loop is executed  $n$  times. So the total time taken by the algorithm is  $T(n) = 2n + 3$ . So the time complexity is  $O(n) = n$ .

### 3.1.2 Space Complexity

$A \rightarrow$  array of size  $n$

$S \rightarrow 1word$

$n \rightarrow 1word$

$i \rightarrow 1word$

So the space complexity is  $S(n) = n + 3$  i.e.  $O(n) = n$ .

## 3.2 Addition of two matrices

---

**Algorithm 4** Addition of Matrices

---

```
procedure MATADD( $A, B, n$ )  
  for (  $i \leftarrow 0; i < n; i++$  ) do  
    for (  $j \leftarrow 0; j < n; j++$  ) do  
       $C[i, j] \leftarrow A[i, j] + B[i, j]$   
    end for  
  end for  
end procedure
```

---