

Algorithms & Data Structures I

Algorithm Analysis: Complexity & Correctness (Courses 1–12)

1. Algorithm Complexity

Algorithm complexity measures the resources required by an algorithm as a function of input size. The two main resources analyzed are time and space.

Time Complexity

Time complexity expresses how the execution time grows with the input size n . The courses use asymptotic notations O , Θ , and Ω .

$\Theta(f(n))$: tight bound (exact growth rate)

$O(f(n))$: upper bound (worst case)

$\Omega(f(n))$: lower bound (best case)

Example: $\Theta(n)$

```
def array_sum(a):  
    s = 0  
    for x in a:  
        s += x  
    return s
```

The loop runs once per element $\rightarrow \Theta(n)$.

Example: $\Theta(n^2)$

```
def count_pairs(a):  
    n = len(a)  
    c = 0  
    for i in range(n):  
        for j in range(n):  
            c += 1  
    return c
```

Two nested loops $\rightarrow \Theta(n^2)$.

Space Complexity

Space complexity measures additional memory usage excluding input. Recursive algorithms also use stack space.

Example: $\Theta(1)$ space

```
def max_value(a):  
    m = a[0]  
    for x in a:  
        if x > m:  
            m = x  
    return m
```

Uses constant extra memory $\rightarrow \Theta(1)$.

2. Recurrence Relations

Recurrence relations describe execution time of recursive algorithms.

Example: Binary search recurrence $T(n)=T(n/2)+1 \rightarrow \Theta(\log n)$.

```
def binary_search(a, x):
    left, right = 0, len(a) - 1
    while left <= right:
        mid = (left + right) // 2
        if a[mid] == x:
            return True
        if x < a[mid]:
            right = mid - 1
        else:
            left = mid + 1
    return False
```

3. Algorithm Correctness

Correctness proves that an algorithm always produces the expected output.

Preconditions and Postconditions

Preconditions must hold before execution; postconditions must hold after.

Example: Precondition – array non-empty. Postcondition – returned value is maximum.

Loop Invariants

A loop invariant is true before and after each iteration.

Example: after iteration i , subarray $a[0..i-1]$ is sorted.

Induction for Recursive Algorithms

```
def factorial(n):  
    if n <= 1:  
        return 1  
    return n * factorial(n - 1)
```

Correctness follows by induction on n .