جامـــــعة المـــــستـقـبـل
AL MUSTAQBAL UNIVERSITY

قســم الامـــــن الـــــــــسيبرانـــــــــي

Department of Cyber Security

Subject:

Algorithms and Complexity

Class:

First

Lecturer:

Asst. lect. Mustafa Ameer Awadh

Lecture: (10)

Lecture Objectives:

By the end of this lecture, students will be able to:

- Understand what an algorithm is and its properties.
- Analyze the time and space complexity of algorithms.
- Distinguish between different types of algorithmic complexity (Big-O, Big-Ω, Big-Θ).
- Apply asymptotic notation to common algorithms.
- Recognize the impact of algorithm efficiency in computer science.

---

1. What is an Algorithm?

Definition:
An algorithm is a finite, well-defined sequence of instructions for solving a problem or performing a task.

Properties of an Algorithm:

- **Input**: Takes zero or more inputs.
- **Output**: Produces at least one output.
- **Finiteness**: Must terminate after a finite number of steps.
- **Effectiveness**: All operations must be basic enough to be carried out.
- **Definiteness**: Each step must be precisely defined.

---

2. Algorithm Efficiency

Two main resources considered:

- **Time** (how long it takes to run)
- **Space** (how much memory it uses)

3. Asymptotic Notation

Used to describe the performance of an algorithm as the input size grows.

Notation	Description	Example
Big-O (O)	Worst-case	$O(n^2)$, $O(\log n)$
Big-Ω (Omega)	Best-case	$\Omega(n)$, $\Omega(1)$
Big-Θ (Theta)	Tight bound	$\Theta(n \log n)$, $\Theta(n)$

Examples:

- Linear Search: $O(n)$
- Binary Search: $O(\log n)$
- Bubble Sort: $O(n^2)$

4. Complexity Classes

- **Constant Time** – $O(1)$: independent of input size
- **Logarithmic Time** – $O(\log n)$: e.g., Binary Search
- **Linear Time** – $O(n)$: e.g., Linear Search
- **Quadratic Time** – $O(n^2)$: e.g., Bubble Sort
- **Exponential Time** – O(2^n): brute-force for some NP problems

5. Comparison of Algorithms

Algorithm	Time Complexity	Space Complexity
Bubble Sort	$O(n^2)$	$O(1)$

Algorithm	Time Complexity	Space Complexity
Merge Sort	O(n log n)	O(n)
Binary Search	O(log n)	O(1)

---

6. Practical Example

Problem: Find the maximum element in an unsorted array.

Algorithm (Pseudocode):

arduino
CopyEdit

```
max ← A[0]
for i = 1 to n-1
    if A[i] > max
        max ← A[i]
return max
```

Time Complexity: O(n)
Space Complexity: O(1)

---

7. Key Takeaways

- Efficient algorithms are crucial in modern computing.
- Big-O helps predict the scalability of algorithms.
- Choice of algorithm affects performance in real applications.
- Discrete structures (graphs, sets, relations) form the basis for algorithm analysis.
