# Lecture
# Chapter 10
# Algorithm Efficiency & Sorting

## Measuring Algorithm Efficiency

- **not coding, i.e., implementation**
- **not platforms, i.e., computer systems**
- **not the data sets**

| |
|---|
| • **but the execution time of the algorithm as measured by the number of operations required by the algorithm** |

**Algorithm A is of the order f(n), i.e., O(f(n))**
**If there exists constants k & $n_0$ such that**
**A produces a solution to a problem of size n >= $n_0$**
**within k * f(n) time units**

| |
|---|
| **growth rate function** |

**e.g., $f(n) = n^2 - 3*n - 10$**

**if k = 3 & $n_0$ = 2 then**
**for all n >= 2**
$\underline{3*n^2} > n^2 - 3*n - 10$

**hence f(n) is of order $O(n^2)$**

**e.g., Linked List**
**displaying/searching the first n items requires**
**(n + 1)*(a + c) + n*w time units**

**for n >= 1**
$\underline{(2*a + 2*c + w)*n} >= (n + 1)*(a + c) + n*w$

**hence the task is of order O(n)**

**e.g., Towers of Hanoi**
**solution requires $(2^n + 1) * m$ time units**
**for n >= 1**
$\underline{m * 2^n} > (2^n + 1) * m$

**hence the solution is $O(2^n)$**

## Order of Growth Rates

| |
|---|
| $O(1) < O(\log n) < O(n) < O(n * \log n) < O(n^2) < O(n^3) < O(2^n)$ |

- **largest order term absorbs smaller order terms**
- **multiplicative and additive constants are absorbed by higher order terms**
- **O(f(n)) + O(g(n)) = O(f(n) + g(n))**

**Worst Case Analysis**

**Average Case Analysis**

**An application's**

- **<u>structure</u>**
- **<u>size</u>**
- **<u>execution time</u> requirements**
- **<u>memory  size</u> requirements**

**will often dictate the appropriate solution**

**algorithms used to solve large problems ➔ order of magnitude analysis**