



CRICOS PROVIDER 00123M

School of Computer Science

# COMP SCI 1103/2103 Algorithm Design & Data Structure

## Complexity – Linked lists

[adelaide.edu.au](http://adelaide.edu.au)

*seek* LIGHT

# Overview

- Today we will have
  - Review on the topic of last session
  - Formal definition of Big O
  - If we have time: Linked list!
  - If we have time: formal definition of Big Omega, Theta, little o

# Review

- We've talked about complexity in general terms.
- Assumptions:
  - The complexity analysis focuses on algorithms
  - The input size is taken as argument.
  - The machine model is used to eliminate the influence of hardware
- The running time complexity of an algorithm matters in
  - the worst case
  - the average case



# Review

- Is it always easy to find the complexity of an algorithm?
  - No!
- We provide a range for the complexity that we are after
  - Upper bound
  - Lower bound
- Usually used for the worst case or the average case.
- When you introduce an instance of the problem as the worst case, is it always possible to prove that it is the worst case?
  - No. (but for the simple search example it was obvious!)
  - Formally, it is usually called a hard instance.
  - It gives a lower bound on the worst case complexity, but for proving an upper bound we need to go generally for the proof

# Questions! (with focus on worst case)

- As a customer
  - Is an upper bound on the worst case important?
  - Is a lower bound on the worst case important?
- As an analyst
  - Do you try to find a high upper bound or low?
  - Do you try to find a high lower bound or low?
  - How can you prove that a worst case upper bound is tight?
    - How can you prove that a worst case lower bound is tight?
- As the algorithm designer
  - Are you happy if an analyst finds a high upper bound or low?
  - Are you happy if an analyst finds a high lower bound or low?
  - Are you happy if an analyst finds high/low upper/lower bounds for your opponent's algorithm?

# Example: A simple search function

```
// pseudo code
search(list, item)
{
    for(i=1 to n)
        if list[i]==item
            return i
}
```

Required time:

- Set up time
- For loop

# Example

- The for loop will take approximately  $n$  times the effort to make decision on one item.
- Set-up for this algorithm is constant.
- Mathematically, the execution time is equal to  $cn + s$ .
  - Where  $c$  and  $s$  are constants and represents the overhead per iteration, and the set-up costs, respectively.
- $f(n) = cn + s$  represents the actual execution time of this searching process.

# Asymptotic Behaviour

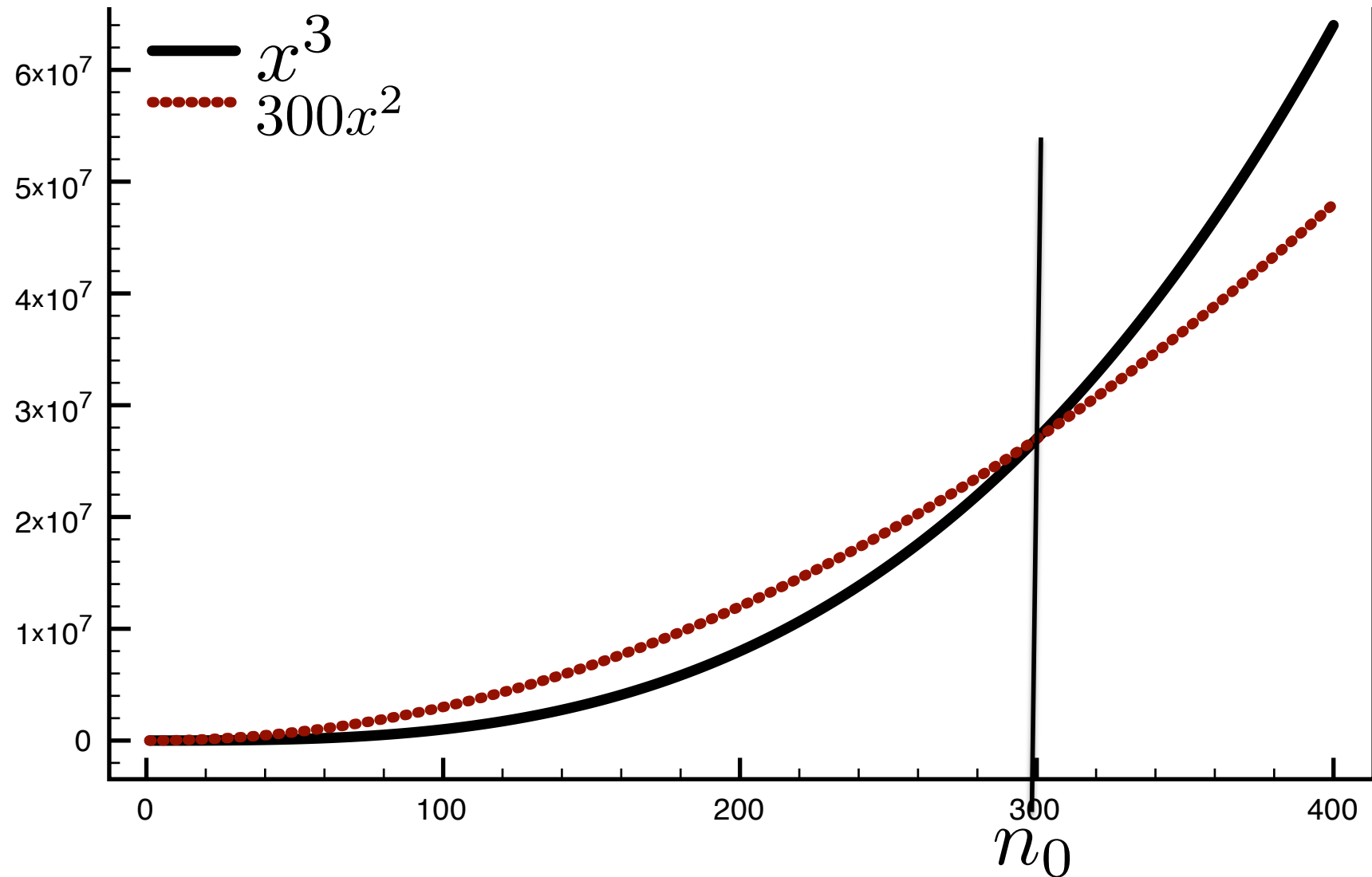
- We measure **runtime** as a **function** of the **input size  $n$** .
- Define **complexity** depending on the **asymptotic behaviour**.
- **Want** to have **algorithms** that solve a given problem and have **low complexity**.



# Big O [ $O(g(n))$ ]

- If  $f(n)$  represents the *actual* execution time of an algorithm, then, as we don't always know the details of  $f(n)$ , we approximate it.
- $f(n) = O(g(n))$  if there exist positive constants  $c$  and  $n_0$  such that  $f(n) \leq c \cdot g(n)$  when  $n \geq n_0$ .
- We refer to this as Big-Oh( $g(n)$ ),  $O(g(n))$  or “order  $g(n)$ ”

# Asymptotic Behavior



# Example

- $n+1 = O(n)$ ?
- $n+2 = O(n)$ ?
- $2n+2 = O(n)$ ?
- $n^2 = O(n)$ ?
  - No!
- $\sqrt{n} = O(n)$ ?
- $1 = O(n)$ ?
- $\log n = O(n)$ ?
- $n \log n = O(n)$ ?
  - No!

# Example

- $n = O(n^2)$ ?
  - Correct but not tight
- $n^2 = O(n^2)$ ?
- $n^3 = O(n^2)$ ?
  - No
- ✱ We want our Big Oh bounds to be:
  - Tight: We want  $g(n)$  to be as close to  $f(n)$  as it can be.
  - Simple: we can drop low order terms and constants. (why?)
- Growth rates are important

# Landau Symbols

We want to measure computation times asymptotically

$$O(f(n)) = \{g(n) : \exists c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\},$$

$$\Omega(f(n)) = \{g(n) : \exists c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\},$$

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n)),$$

$$o(f(n)) = \{g(n) : \forall c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\},$$

$$\omega(f(n)) = \{g(n) : \forall c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}.$$

Mehlhorn, Sanders (page 21)

We often write  $h = O(f)$  instead of  $h \in O(f)$  and  $O(h) = O(f)$  instead of  $O(h) \subseteq O(f)$ .

# Examples

$$5n : O(n), \Omega(n), \Theta(n), o(n \log n), \omega(\sqrt{n})$$

$$n^2 - n \log n : O(n^2), \Omega(n^2), \Theta(n^2), o(n^3), \omega(n \log n)$$

$$100n : O(n^2), \Omega(\sqrt{n}), \Theta(n), o(n \log n), \omega(\sqrt{n})$$



# Right or Wrong

$$5n \log n \in O(n \log n)$$

Right

$$5n \log n \in O(n^2)$$

Right

$$5n \log n \in \Omega(n^2)$$

Wrong

$$5n \log n \in o(n^2)$$

Right

$$5n \log n + n^2 \in O(n \log n)$$

Wrong

$$5n \log n + n^2 \in O(n^2)$$

Right



THE UNIVERSITY  
*of* ADELAIDE