Algorithms and Data Structures
A9. Runtime Analysis: Application

Gabriele Röger and Patrick Schnider

University of Basel

March 11, 2026
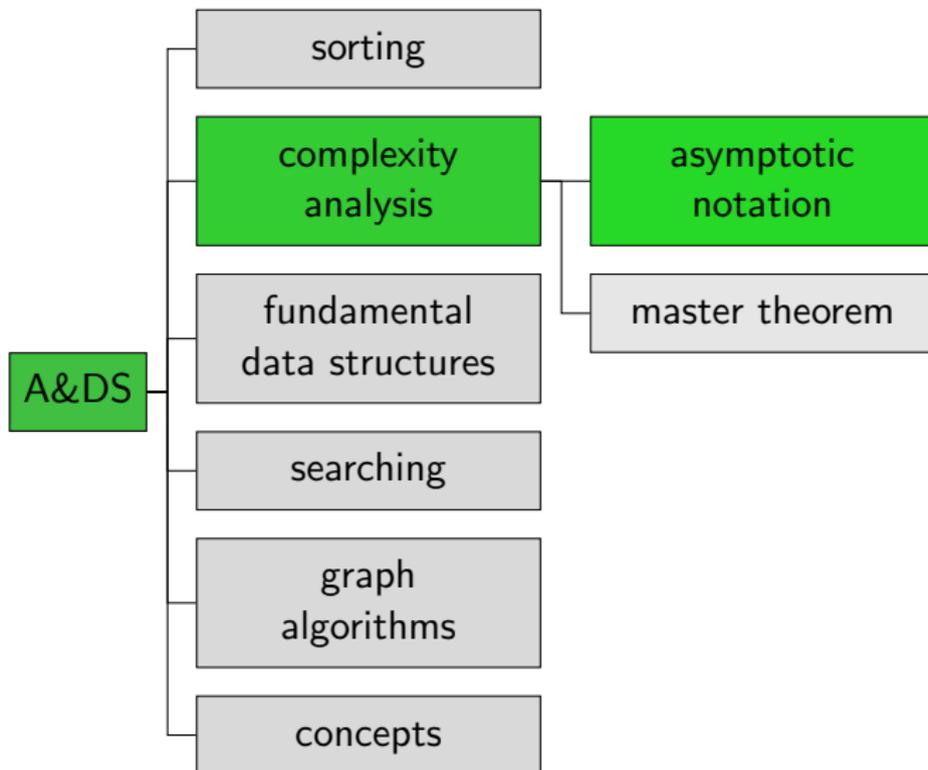
A9.1 Application

A9.2 Summary

Content of the Course

A9.1 Application

Quick *O*-Analysis for Common Code Patterns I

▶ Constant-time operation:

var = 4	$O(1)$

▶ Sequence of constant-time operations:

var1 = 4	$O(1)$	
var2 = 4	$O(1)$	$O(123 \cdot 1) = O(1)$
...	...	
var123 = 4	$O(1)$	

Quick O-Analysis for Common Code Patterns II

▶ Loop:

`for i in range(n):`	$O(n)$
`  res += i * m`	$O(1)$

$O(n \cdot 1) = O(n)$

`for i in range(n):`	$O(n)$	$O(n)$	
`  for j in range(i):`	$O(n)$	$O(n)$	$O(n^2)$
`    res += i * (m - j)`	$O(1)$		

i depends on n.

Quick O-Analysis for Common Code Patterns III

► if-then-else

`if var < bound:`	$O(1)$	$O(1)$
`  res += var`	$O(1)$	$O(1)$
`else:`		
`  for i in range(n):`	$O(n)$	$O(n \cdot 1)$
`    res += i * n`	$O(1)$	$= O(n)$

$O(1 + \max\{1, n\})$
$= O(n)$

Attention: Can lead to unnecessarily loose bound
if the expensive case only occurs with small n
(bound by a constant).

Example: Worst Case for Insertion Sort

```python
1  def insertion_sort(array):
2      n = len(array)
3      for i in range(1, n):  # i = 1, ..., n - 1
4          # move array[i] to the left until it is
5          # at the correct position.
6          for j in range(i, 0, -1):  # j = i, ..., 1
7              if array[j] < array[j-1]:
8                  array[j], array[j-1] = array[j-1], array[j]
9              else:
10                 break
```

▶ Worst case: `break` never happens.

▶ $O(1 + n \cdot n \cdot 1) = O(n^2)$

▶ Over-estimated?
   No, each of the two loops has $\Omega(n)$ iterations.

Example: Best Case for Insertion Sort

```
1  def insertion_sort(array):
2      n = len(array)
3      for i in range(1, n):  # i = 1, ..., n - 1
4          # move array[i] to the left until it is
5          # at the correct position.
6          for j in range(i, 0, -1):  # j = i, ..., 1
7              if array[j] < array[j-1]:
8                  array[j], array[j-1] = array[j-1], array[j]
9              else:
10                 break
```

▶ Best case: break always immediately with $j = i$

▶ $O(1 + n \cdot 1 \cdot 1) = O(n)$

▶ Over-estimated?
  No, the outer loop has $\Omega(n)$ iterations.

Exam Question from 2019

Consider the following code fragment.
Specify the asymptotic running time (depending on $n \in \mathbb{N}$)
in Θ notation and justify your answer (1-2 sentences).

```
1   int result = 0;
2   if (n > 23) {
3       return result;
4   }
5   for (int i = 0; i < n; i++) {
6       for (int j = 0; j < n; j++) {
7           result += j;
8       }
9   }
10  return result;
```

Why are we Interested in All This?

▶ Because algorithms/data structures with bad runtime complexity strike back!
▶ Example: for several years, GTA online took several minutes to load.
  ▶ Several minutes for parsing 10 megabyte of JSON data!
  ▶ Probably bad library for parsing
  ▶ Unsuitable data structure for duplication check
  ▶ After fix: 70% less loading time
  ▶ https://nee.lv/2021/02/28/
    How-I-cut-GTA-Online-loading-times-by-70/index.
    html

A9.2 Summary

Summary

- In practice, we quite quickly can get an impression of the running time of an algorithm with simple "cookbook recipes".
- Insertion sort has
  - in the best case running time $\Theta(n)$.
  - in the worst case running time $\Theta(n^2)$.