

Complexity notes

Priit Järv, TUT 2015

How long does this function execute?

```
def sum(list):  
    sum = 0  
    for elem in list:  
        sum += elem  
    return sum
```

(this function computes $\sum a_i$ for $\{a_1, \dots, a_n\}$)

How long does this function execute?

```
def sum(list):  
    sum = 0  
    for elem in list:  
        sum += elem  
    return sum
```

We can count the number of code lines:

$2n + 2$ if the length of list is n

Complexity of algorithms

Estimate of time complexity $T(n) = 2n+2$ for input n

We use the big-O notation (asymptotic complexity)

$T(n)$ is $O(f(n))$ if $T(n) \leq kf(n)$ for some k and $n > n_0$

In this case, $O(n)$ because we can choose (for example) $k=3$ and $n_0=1$. Then $2n+2 \leq 3n$ for $n > 1$.

Complexity of algorithms

The practical usefulness of the big-O notation:

	problem size increase	time increase
$O(n)$	5x	5x
$O(n^2)$	5x	25x
$O(2^n)$	5x	A LOT*

*- for example, about 10^{12} times if increasing from 10 to 50

Polynomial time

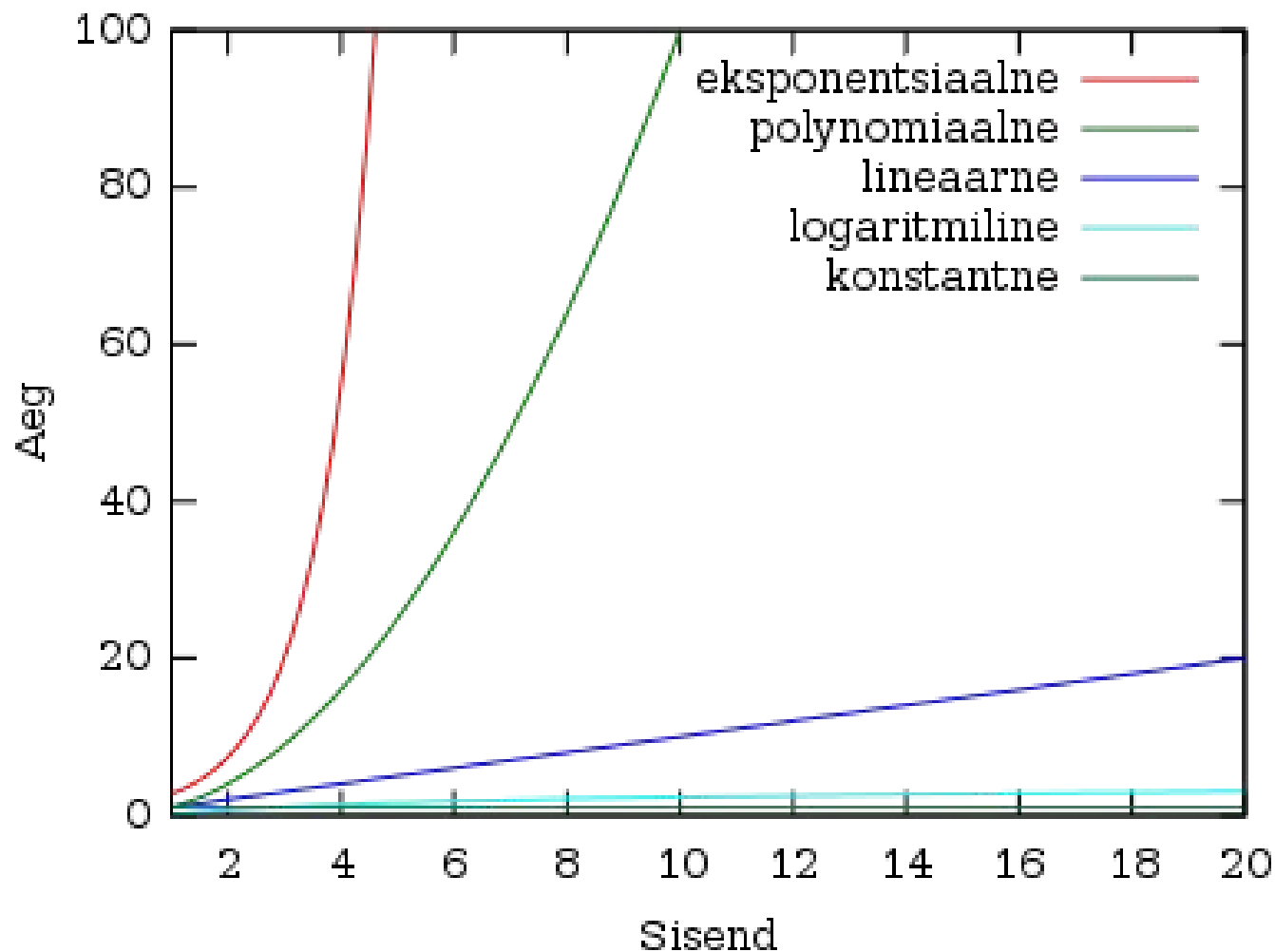
$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ is n-th order polynomial

These are also polynomials:

$2x, 6, x^3$

So $O(n^3)$ is polynomial complexity and means the algorithm takes input size cubed time to run.

Common complexity classes



Complexity of problems

There are two main classes: P and NP

Class P implies that the problem is solvable in polynomial time.

More specifically, this is true on a deterministic Turing machine (sort of equivalent to a contemporary computer)

Complexity of problems

Problems in NP are *ALSO* solvable in polynomial time, but:

- require a special nondeterministic Turing machine
- it is a random computer with non-zero probability that it actually accidentally solves the problem

Complexity of problems

How do we know that a problem is in class NP

- Some problems have been proven to be NP-complete (means “in NP and hard to solve”)
- We take a known NP-complete problem X and convert it (in polynomial time) to a new problem Y
- If this succeeds, then Y is also in NP

Complexity of problems

Practical implication:

There are no polynomial time algorithms for problems that are NP-something.