Recurrence Relations

Birgit Vogtenhuber

TU Graz

Overview

- What are recurrence relations?
- Recall asymptotic notations: $\mathcal{O}$, Ω, Θ
- Depict recurrence relations via recursion trees
- Three methods for solving recurrence relations
- Exercises

What are recurrence relations?

Recurrence relation: equation (or inequality) that describes a function in terms of its values on smaller inputs, plus one or more initial terms.

Example: factorial function
$f(n) = n \cdot f(n-1)$ for $n > 1$, $f(1) = 1$

Example: fibonacci numbers
$f(n) = f(n-1) + f(n-2)$ for $n > 2$, $f(1) = f(2) = 1$

Applications:

- representation of combinatorial relations
- analysis of memory consumption and running time of algorithms

What are recurrence relations?

Recurrence relations in the context of runtime and memory consumption analysis:

Example: runtime of Merge-Sort

$$T(n) = \begin{cases} \Theta(1) & \text{for } n = 1 \\ 2T(n/2) + \Theta(n) & \text{for } n \geq 2 \end{cases}$$

Wanted: explicit asymptotic bounds for $T(n)$:

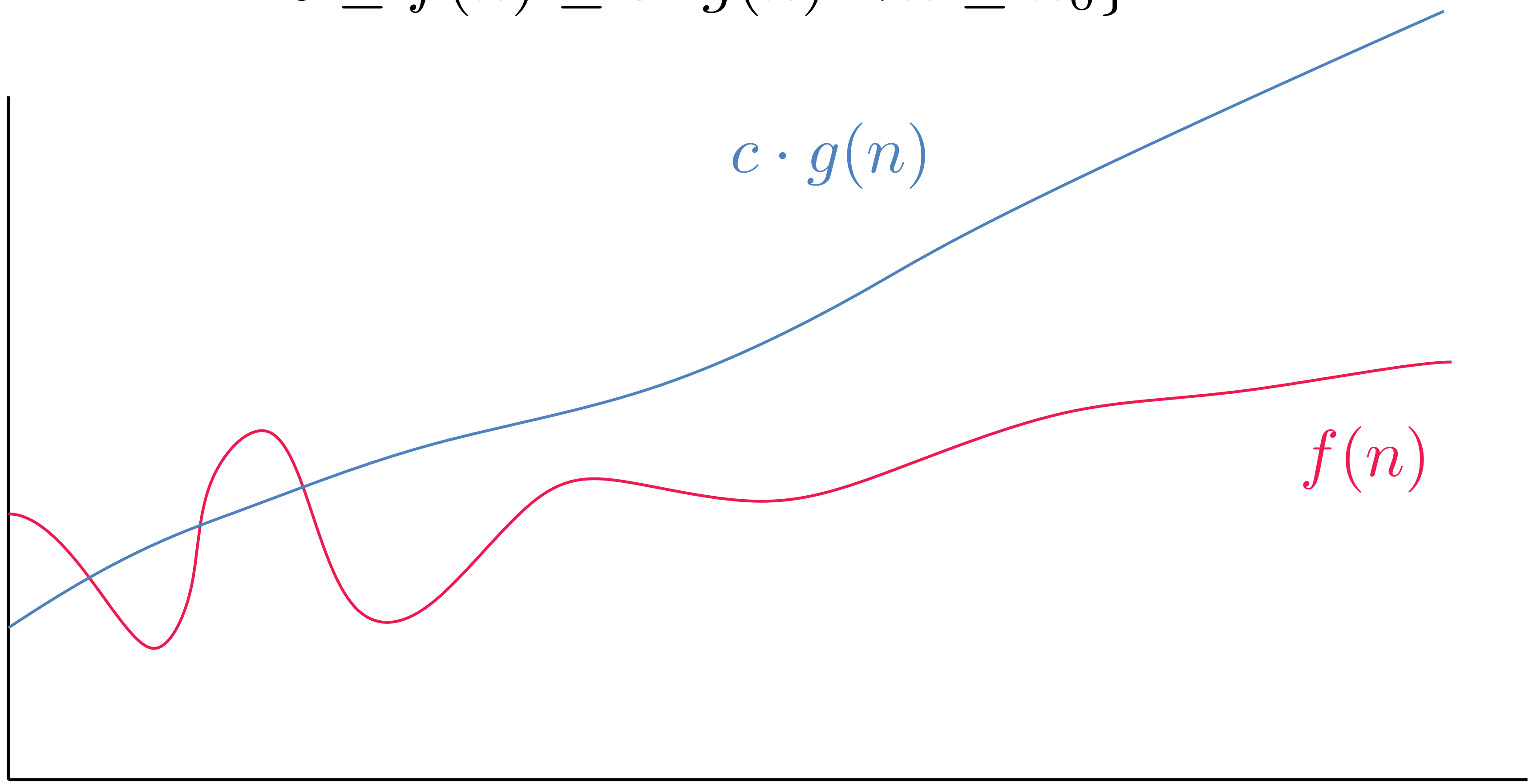
$$T(n) = \Theta(f(n)),\ T(n) = \mathcal{O}(f(n)),\ T(n) = \Omega(f(n))$$

Initial values: constant size input only requires constant runtime and memory. Hence we always have

$$T(c) = \Theta(1) \text{ for } c \text{ constant}$$

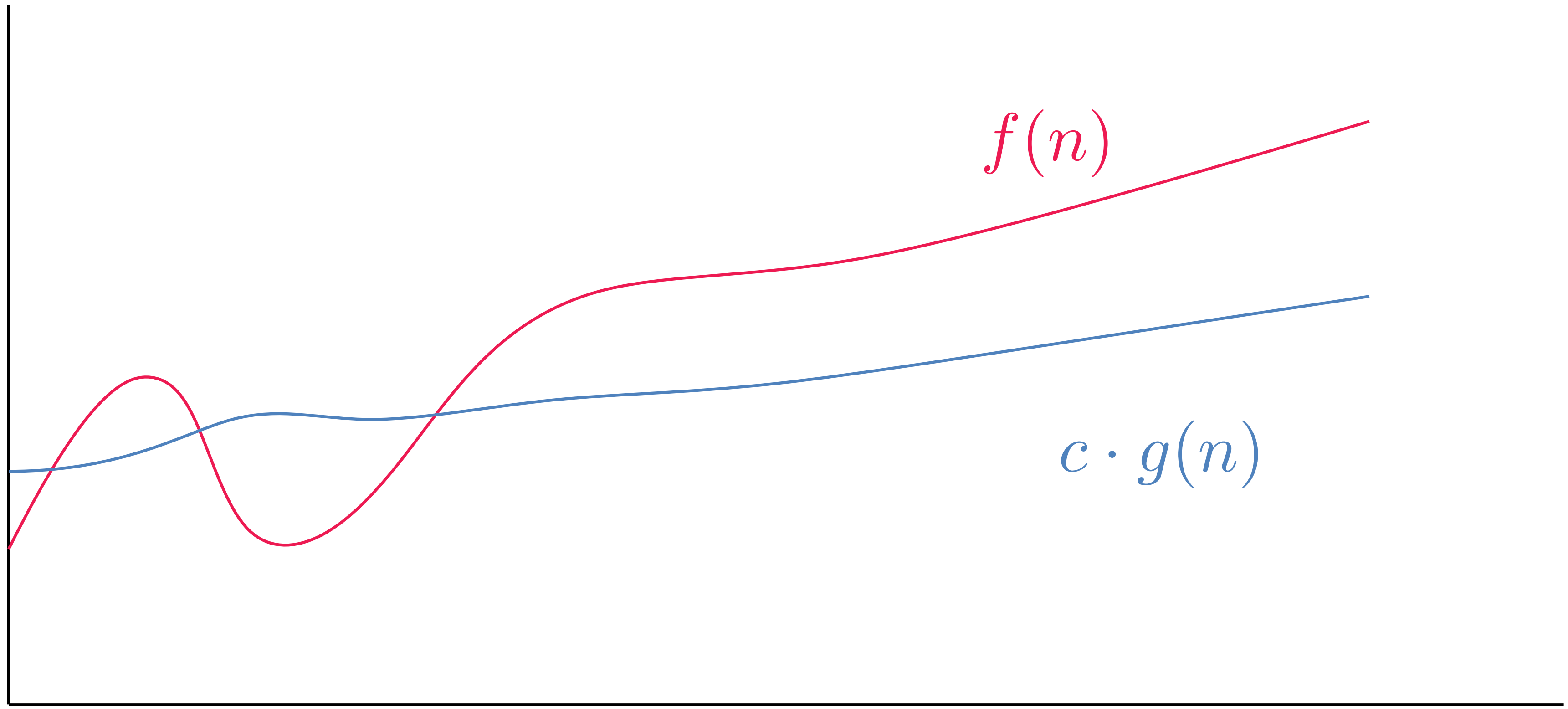
Asymptotic notation

$$\mathcal{O}(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \mid \exists\, c \in \mathbb{R}^+,\ n_0 \in \mathbb{N} : 0 \leq f(n) \leq c \cdot g(n) \ \ \forall n \geq n_0\}$$



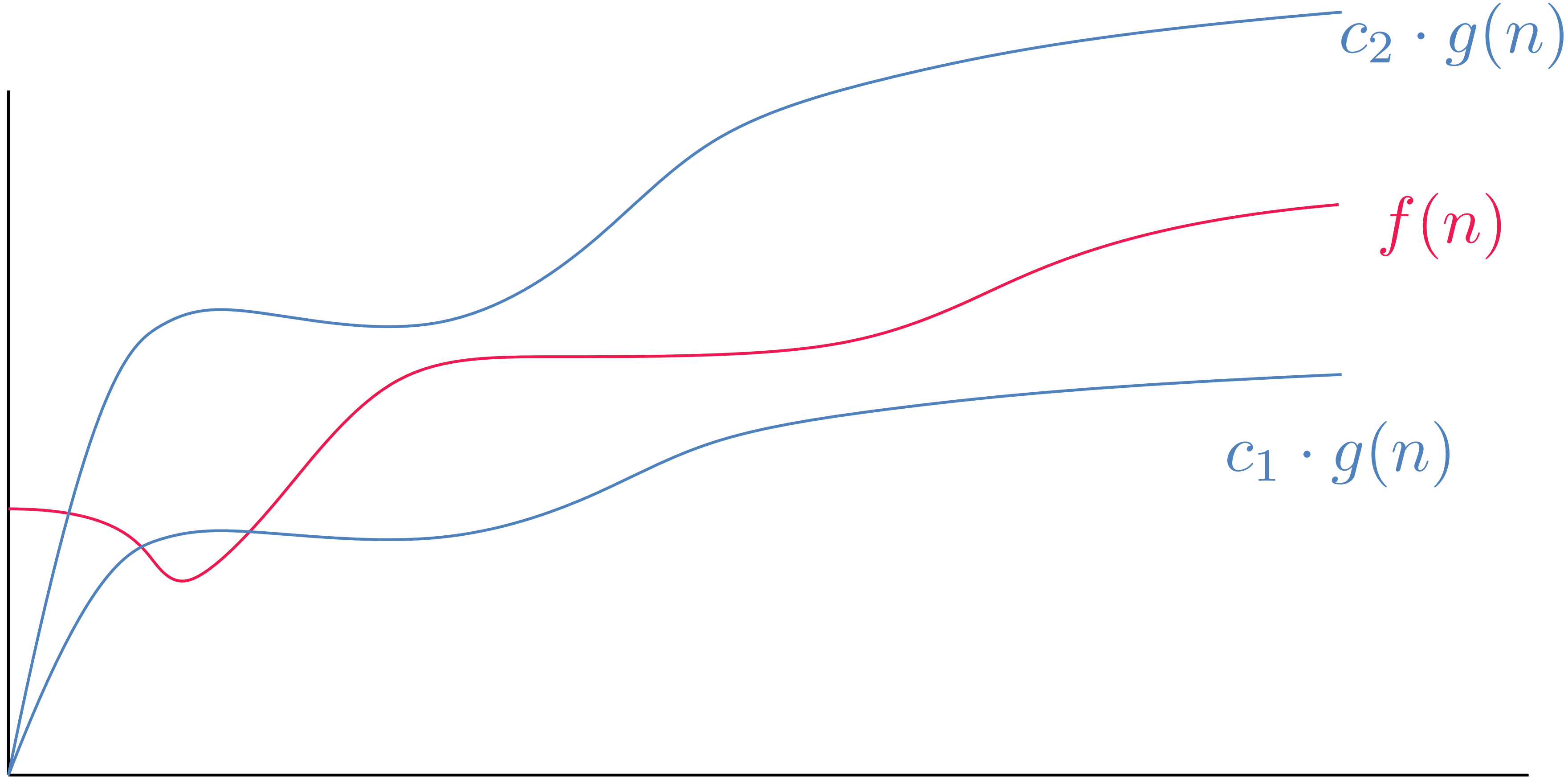
Asymptotic notation

$$\Omega(g(n)) = \{f : \mathbb{N} \to \mathbb{R}^+ \mid \exists\, c \in \mathbb{R}^+,\ n_0 \in \mathbb{N} : 0 \le c \cdot g(n) \le f(n) \ \ \forall n \ge n_0\}$$



Asymptotic notation

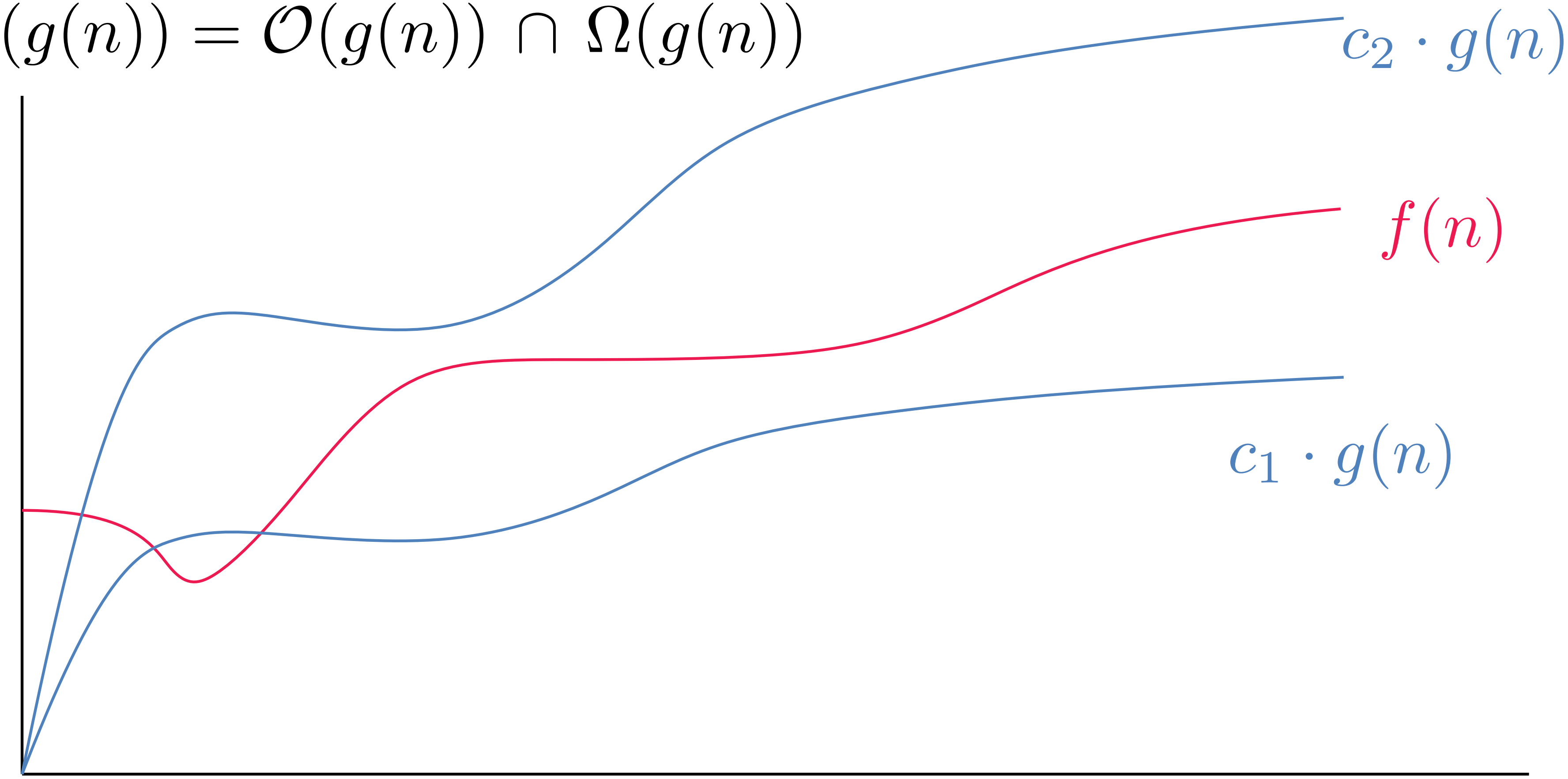
$$\Theta(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \mid \exists\, c_1,\ c_2 \in \mathbb{R}^+,\ n_0 \in \mathbb{N} : \\ 0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \ \ \forall n \ge n_0\}$$



Asymptotic notation

$$\Theta(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \mid \exists\, c_1,\ c_2 \in \mathbb{R}^+,\ n_0 \in \mathbb{N} : \\ 0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \ \ \forall n \ge n_0\}$$

$$\Theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$



Asymptotic notation

$$\mathcal{O}(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \,|\, \exists\, c \in \mathbb{R}^+,\ n_0 \in \mathbb{N} :$$
$$0 \leq f(n) \leq c \cdot g(n)\ \ \forall n \geq n_0\}$$

$$\Omega(g(n)) = \{f : \mathbb{N} \to \mathbb{R}^+ \,|\, \exists\, c \in \mathbb{R}^+,\ n_0 \in \mathbb{N} :$$
$$0 \leq c \cdot g(n) \leq f(n)\ \ \forall n \geq n_0\}$$

$$\Theta(g(n)) = \{f : \mathbb{N} \to \mathbb{R} \,|\, \exists\, c_1,\ c_2 \in \mathbb{R}^+,\ n_0 \in \mathbb{N} :$$
$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\ \ \forall n \geq n_0\}$$

$$\Theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$

Asymptotic notation

Asymptotic notation and limits

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) = \mathcal{O}(g(n)) \\ 0 < c < \infty & f(n) = \Theta(g(n)), \\ & \quad \mathcal{O}(g(n)), \Omega(g(n)) \\ \infty & f(n) = \Omega(g(n)) \end{cases}$$

$$\liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0 \qquad f(n) = \Omega(g(n))$$

$$\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty (\text{and } \geq 0) \qquad f(n) = \mathcal{O}(g(n))$$

if both are true: $\quad f(n) = \Theta(g(n))$

Asymptotic notation

Computing with asymptotic notation

- Addition:

  $$\Theta(f(n)) + \Theta(g(n)) = \Theta(\max\{f(n), g(n)\})$$

  $$\sum_i \Theta(f_i(n)) = \Theta\Big(\sum_i f_i(n)\Big)$$

  $\boxed{\text{Attention: not iteratively!}}$

- Mulitplication:

  $$c \cdot \Theta(f(n)) = \Theta(f(n)) \quad \text{for constant } c > 0$$

  $$\Theta(f(n)) \cdot \Theta(g(n)) = \Theta(f(n) \cdot g(n))$$

- Attention: equations always "from left to right":

  $$f(n) = \Theta(g(n)) \quad \text{vs.} \quad \xcancel{\Theta(g(n)) = f(n)}$$

Methods to solve recurrence relations

- The iterative method
- The master method
- The substitution method

The iterative method

Idea: repeatedly plug in recurrence

Example: $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

$$\begin{aligned}
&= 2\left(2T\left(\tfrac{n}{4}\right) + \left(\tfrac{n}{2}\right)^2\right) + n^2 \\
&= 4T\left(\tfrac{n}{4}\right) + \frac{n^2}{2} + n^2 \\
&= 4\left(2T\left(\tfrac{n}{8}\right) + \left(\tfrac{n}{4}\right)^2\right) + \frac{n^2}{2} + n^2 \\
&= 8T\left(\tfrac{n}{8}\right) + \frac{n^2}{4} + \frac{n^2}{2} + n^2 = \ldots \\
\ldots &= 2^k T\left(\tfrac{n}{2^k}\right) + \frac{n^2}{2^{k-1}} + \cdots + \frac{n^2}{2} + n^2 \\
&= 2^k T\left(\tfrac{n}{2^k}\right) + n^2 \sum_{i=0}^{k-1} \frac{1}{2^i}
\end{aligned}$$

The iterative method

$$T(n) = 2^k T\left(\tfrac{n}{2^k}\right) + n^2 \sum_{i=0}^{k-1} \tfrac{1}{2^i}$$

$$\tfrac{n}{2^k} = 1 \text{ implies } k = \log_2 n$$

$$\Rightarrow T(n) = 2^{\log_2(n)} T(1) + n^2 \sum_{i=0}^{\log_2(n)-1} \tfrac{1}{2^i}$$

$$= n \cdot \Theta(1) + n^2 \sum_{i=0}^{\log_2(n)-1} \tfrac{1}{2^i}$$

$$= n \cdot \Theta(1) + n^2 \cdot \Theta(1) = \Theta(n) + \Theta(n^2)$$

$$\Rightarrow T(n) = \Theta(n^2)$$

Recursion trees

Visualization of a recurrence: recursion tree

Example: again $T(n) = 2T(\frac{n}{2}) + n^2$

$$T(n)$$

Recursion trees

Visualization of a recurrence: recursion tree

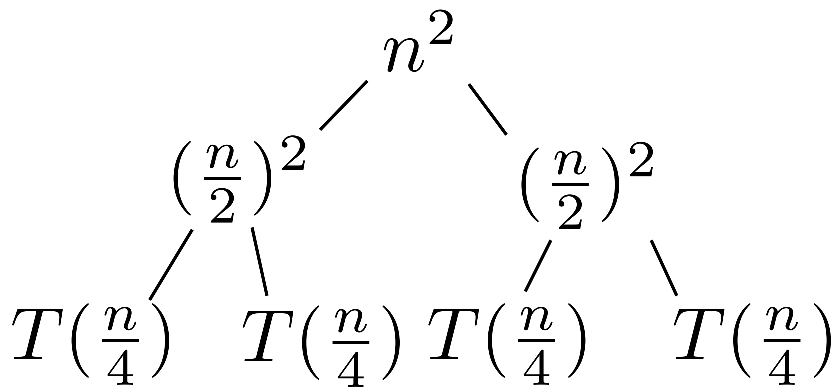
Example: again $T(n) = 2T(\frac{n}{2}) + n^2$

$$
\begin{array}{ccc}
 & n^2 & \\
 / & & \backslash \\
T(\frac{n}{2}) & & T(\frac{n}{2})
\end{array}
$$

Recursion trees

Visualization of a recurrence: recursion tree

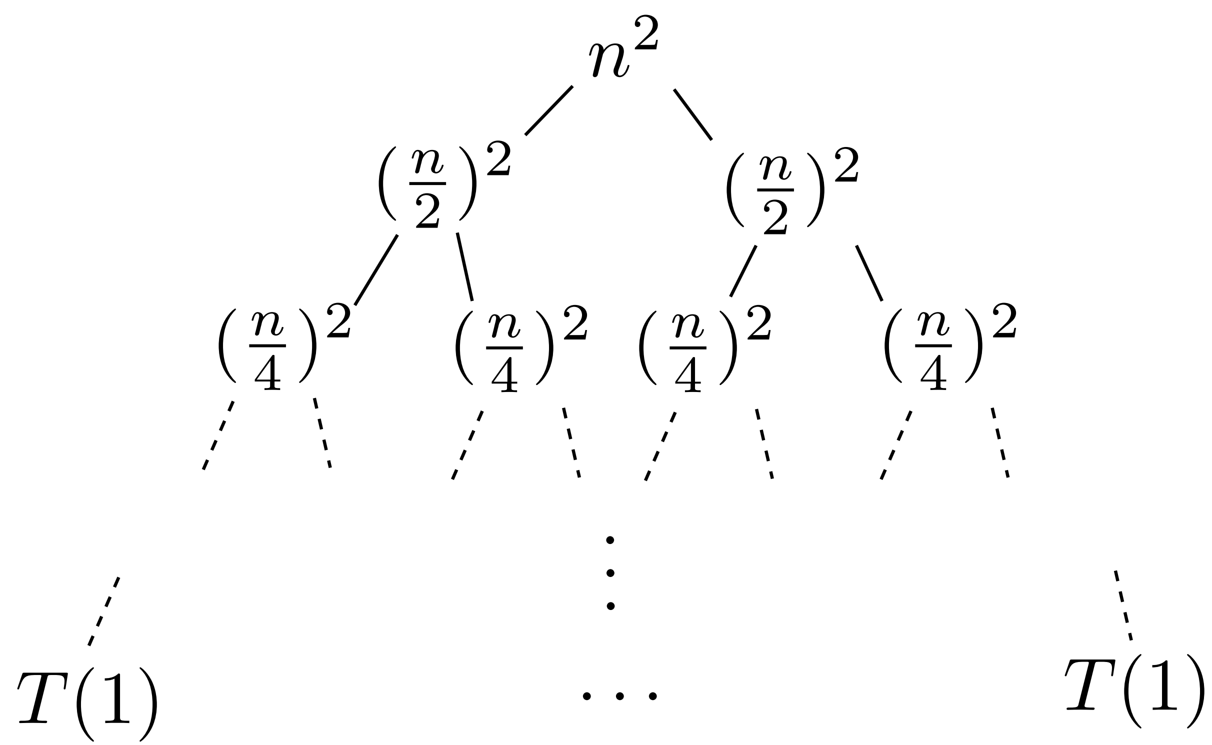
Example: again $T(n) = 2T(\frac{n}{2}) + n^2$



Recursion trees

Visualization of a recurrence: recursion tree

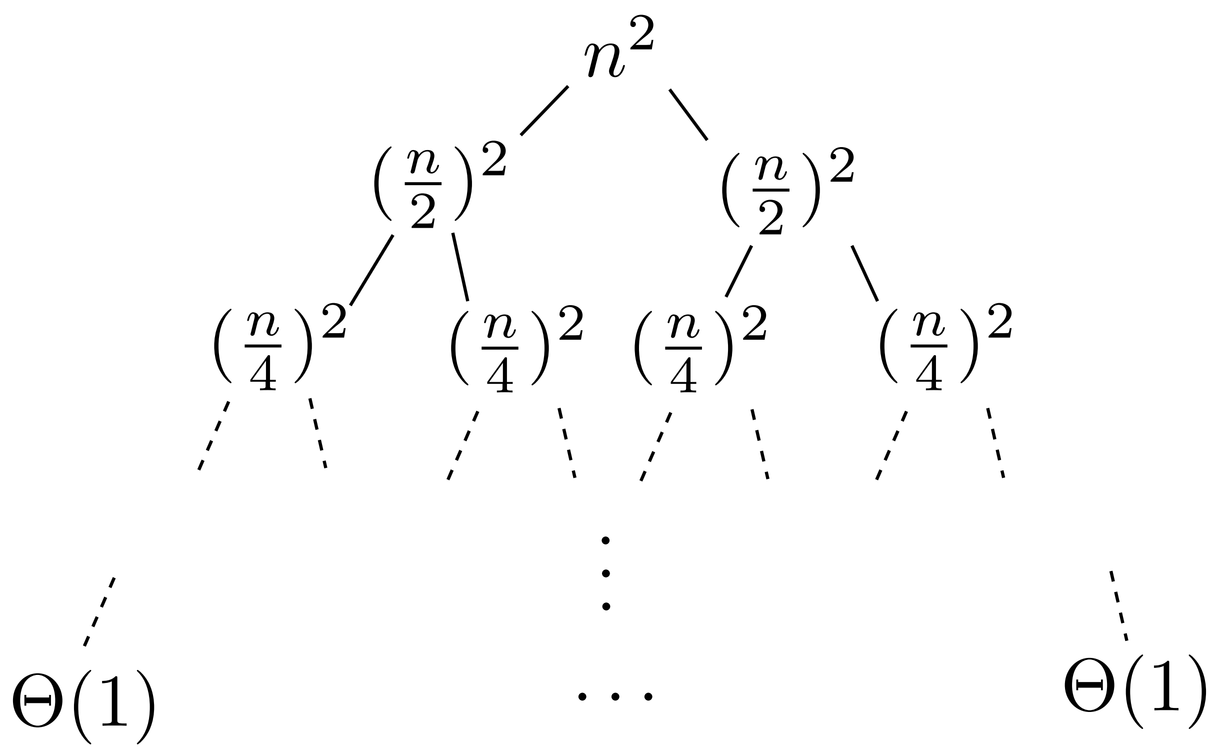
Example: again $T(n) = 2T(\frac{n}{2}) + n^2$



Recursion trees

Visualization of a recurrence: recursion tree

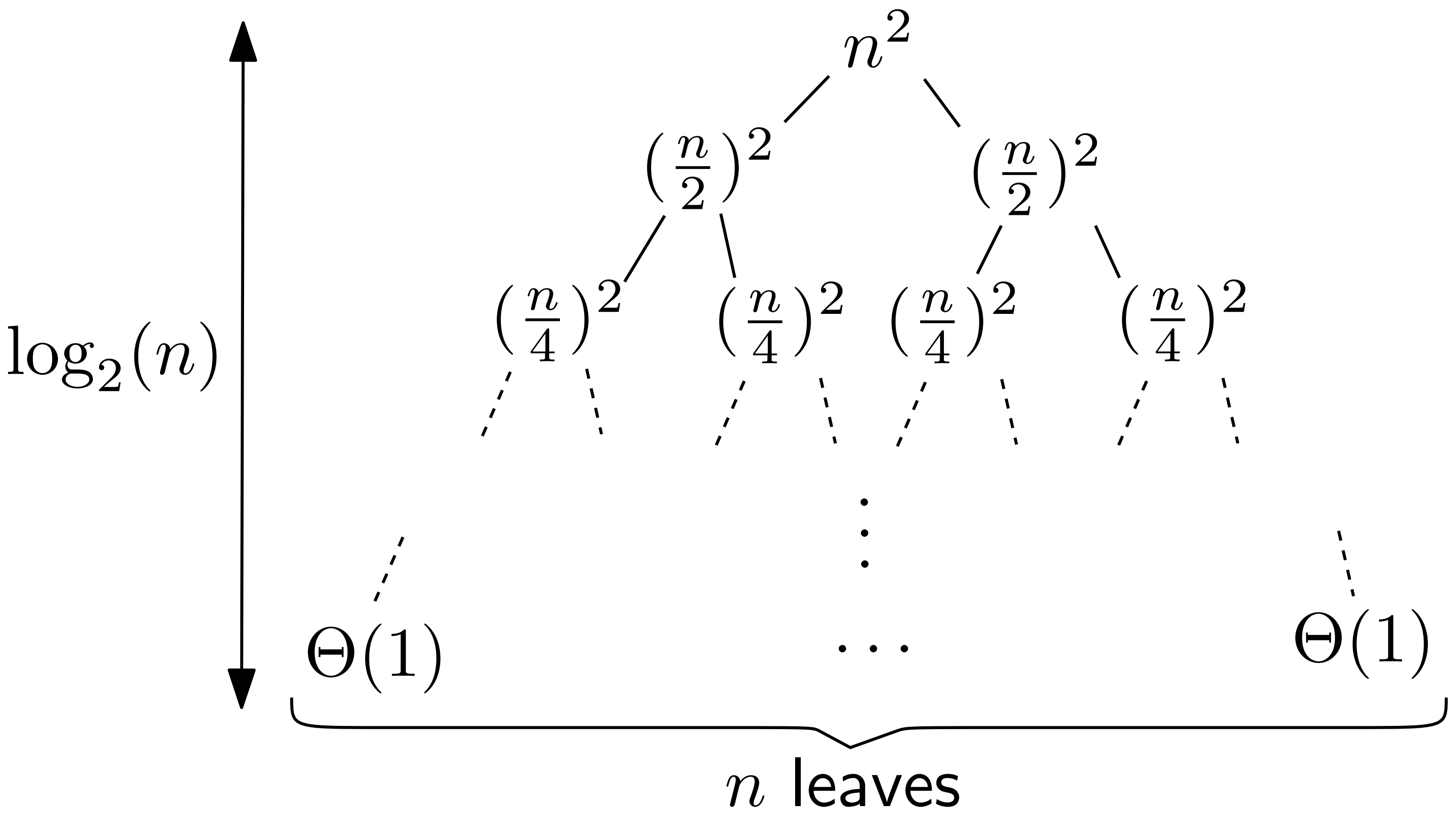
Example: again $T(n) = 2T(\frac{n}{2}) + n^2$



Recursion trees

Visualization of a recurrence: recursion tree

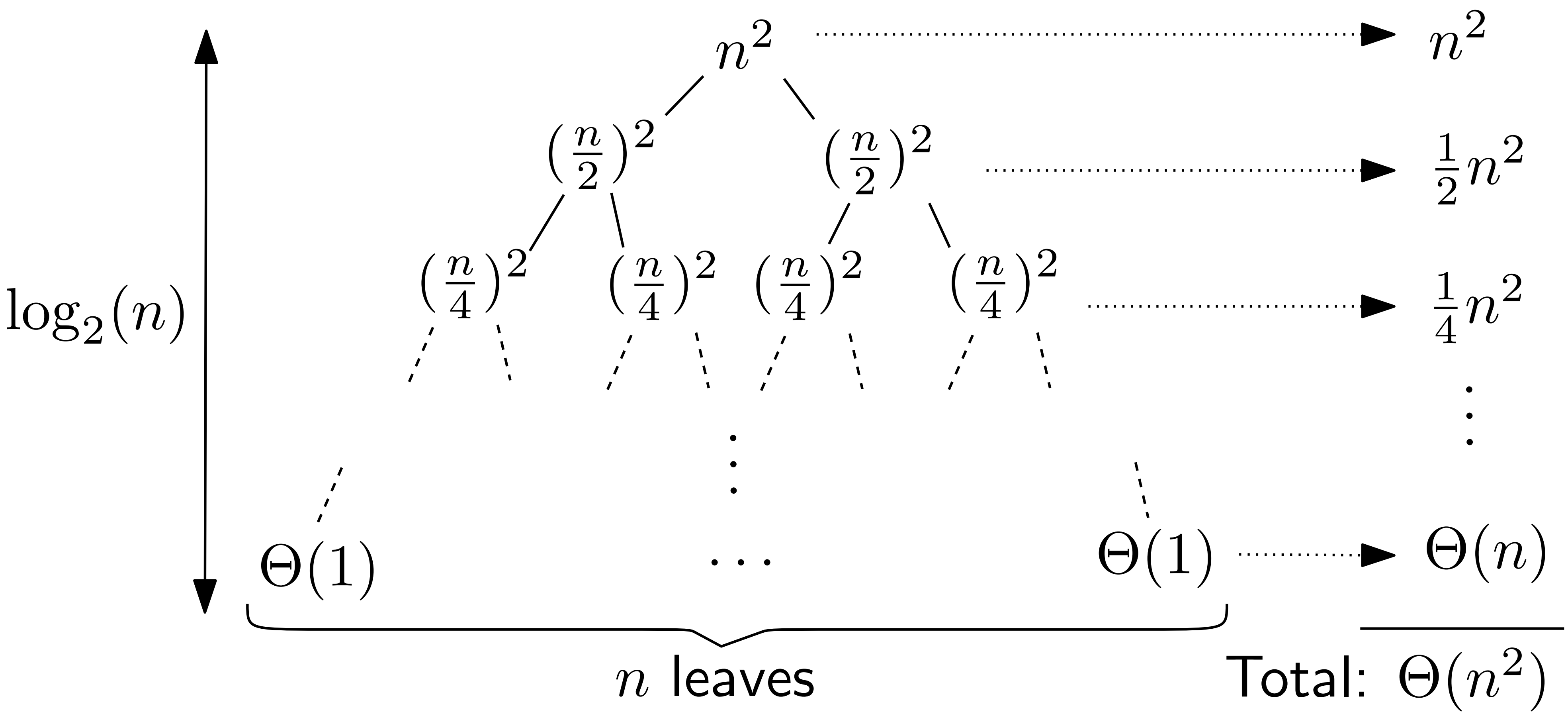
Example: again $T(n) = 2T(\frac{n}{2}) + n^2$



Recursion trees

Visualization of a recurrence: recursion tree

Example: again $T(n) = 2T(\frac{n}{2}) + n^2$



The master method

"cooking recipe" to solve recurrences of the form

$$T(n) = aT(\frac{n}{b}) + f(n)$$

with $a \geq 1$ and $b > 1$.

In the following cases we directly get the solution ...

Case 1: $f(n) = \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ for some $\varepsilon > 0$

$$\Rightarrow T(n) = \Theta(n^{\log_b a})$$

Case 2: $f(n) = \Theta(n^{\log_b a})$ $\qquad \Rightarrow T(n) = \Theta(n^{\log_b a} \log n)$

Case 3: $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ for some $\varepsilon > 0$
and $\exists c < 1$ such that $a \cdot f(\frac{n}{b}) \leq c \cdot f(n), n \geq n_0$

$$\Rightarrow T(n) = \Theta(f(n))$$

The master method

Example 1: $T(n) = 4T(\frac{n}{2}) + \Theta(n)$

parameter: $a = 4, \;\; b = 2, \;\; f(n) = \Theta(n) = \Theta(n^1)$
$\log_b(a) = \log_2(4) = 2, \;\; n^{\log_b(a)} = n^2$

compare $f(n)$ with $n^{\log_b(a)}$:

$$f(n) = \Theta(n^1) \stackrel{?}{=} \mathcal{O}(n^{2-\varepsilon})$$

for $2 - \varepsilon \geq 1, \; 1 \geq \varepsilon$
$\Rightarrow$ Yes: $\varepsilon = 1$

$$\stackrel{?}{=} \Theta(n^2)$$

$\Rightarrow$ No

$$\stackrel{?}{=} \Omega(n^{2+\varepsilon})$$

for $2 + \varepsilon \leq 1, \; \varepsilon \leq -1$
$\Rightarrow$ No

$\Rightarrow$ Case 1 $\quad \Rightarrow T(n) = \Theta(n^{\log_b(a)}) = \Theta(n^2)$

The master method

Example 2: $T(n) = 4T(\frac{n}{2}) + \Theta(n^2)$

parameter: $a = 4,\ \ b = 2,\ \ f(n) = \Theta(n^2)$
$\log_b(a) = \log_2(4) = 2,\ \ n^{\log_b(a)} = n^2$

compare $f(n)$ with $n^{\log_b(a)}$:

$$f(n) = \Theta(n^2) \stackrel{?}{=} \mathcal{O}(n^{2-\varepsilon})$$ for $2 - \varepsilon \geq 2$, $0 \geq \varepsilon$ $\Rightarrow$ No

$$\stackrel{?}{=} \Theta(n^2)$$ $\Rightarrow$ Yes

$$\stackrel{?}{=} \Omega(n^{2+\varepsilon})$$ for $2 + \varepsilon \leq 2$, $\varepsilon \leq 0$ $\Rightarrow$ No

$\Rightarrow$ Case 2 $\quad \Rightarrow T(n) = \Theta(n^{\log_b(a)} \log(n)) = \Theta(n^2 \log(n))$

The master method

Example 3: $T(n) = 3T(\frac{n}{2}) + \Theta(n^2)$

parameter: $a = 3, \;\; b = 2, \;\; f(n) = \Theta(n^2)$

$\log_b(a) = \log_2(3), \;\; 1 < \log_2(3) < 2$

$n^{\log_b(a)} = n^{\log_2(3)}$

compare $f(n)$ with $n^{\log_b(a)}$:

$f(n) = \Theta(n^2) \stackrel{?}{=} \mathcal{O}(n^{log_2(3)-\varepsilon})$ for $\log_2(3) - \varepsilon \geq 1$, $\log_2(3) - 2 \geq \varepsilon$ $\Rightarrow$ No

$\stackrel{?}{=} \Theta(n^{log_2(3)})$ $\Rightarrow$ No

$\stackrel{?}{=} \Omega(n^{log_2(3)+\varepsilon})$ for $\log_2(3) + \varepsilon \leq 2$, $\varepsilon \leq 2 - \log_2(3)$ $\Rightarrow$ Yes $\Rightarrow$ Case 3?

The master method

Example 3: $T(n) = 3T(\frac{n}{2}) + \Theta(n^2)$

parameter: $a = 3, \;\; b = 2, \;\; f(n) = \Theta(n^2)$
$\log_b(a) = \log_2(3), \;\; 1 < \log_2(3) < 2$
$n^{\log_b(a)} = n^{\log_2(3)}$

$f(n) = \Theta(n^2) = \Omega(n^{\log_2(3)+\varepsilon})$

Check additional condition:
$\exists c < 1$ such that $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ for $n \geq n_0$?

$$a \cdot f(\tfrac{n}{b}) = 3 \cdot (\tfrac{n}{2})^2 = \tfrac{3}{4} \cdot n^2 \leq c \cdot n^2$$

$\Rightarrow$ Yes for $n \geq 1$ and $\frac{3}{4} \leq c < 1$

$\Rightarrow$ Case 3 $\quad \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^2)$

Recursion tree for the master method

Some intuition why the master method is actually correct

$$T(n) = aT(\tfrac{n}{b}) + f(n)$$

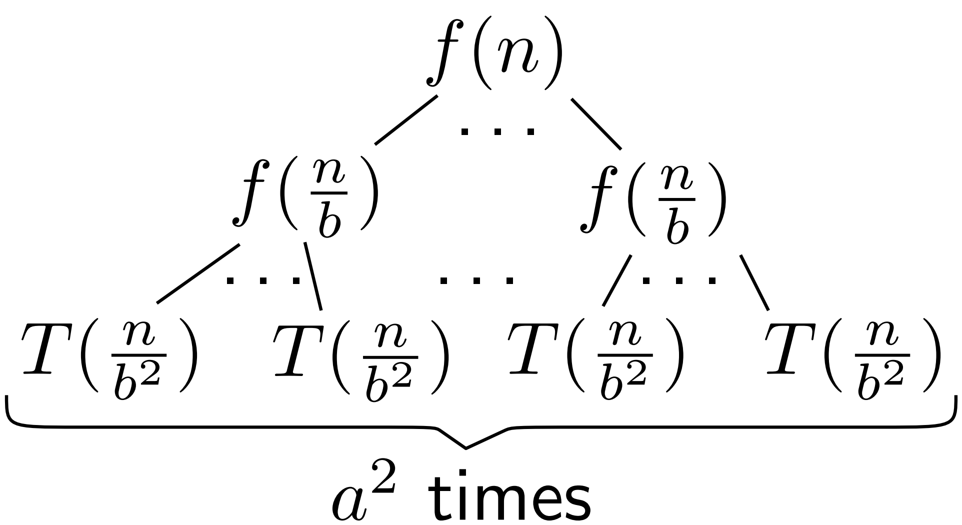
$$f(n)$$

$$\underbrace{T(\tfrac{n}{b}) \quad \cdots \quad T(\tfrac{n}{b})}_{a \text{ times}}$$

Recursion tree for the master method

Some intuition why the master method is actually correct

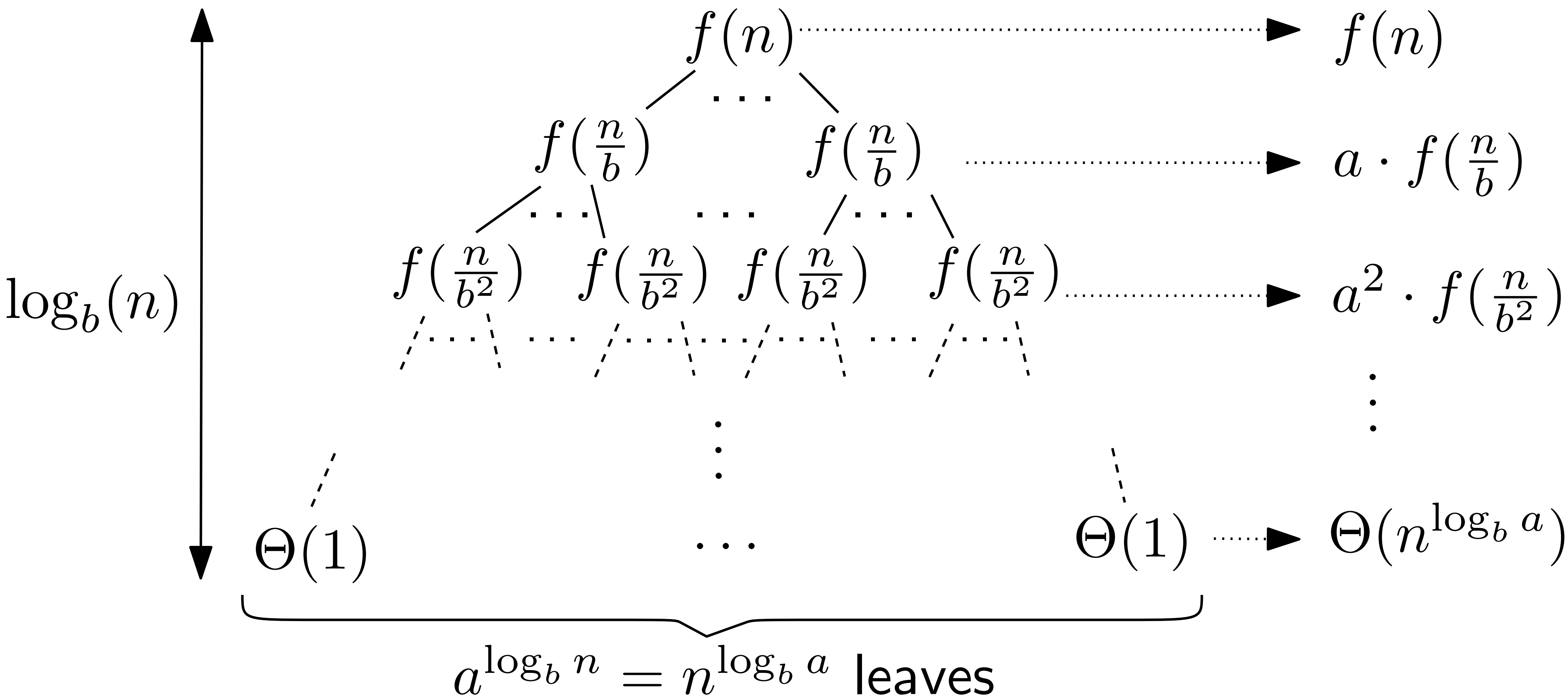
$T(n) = aT(\frac{n}{b}) + f(n)$



Recursion tree for the master method

Some intuition why the master method is actually correct

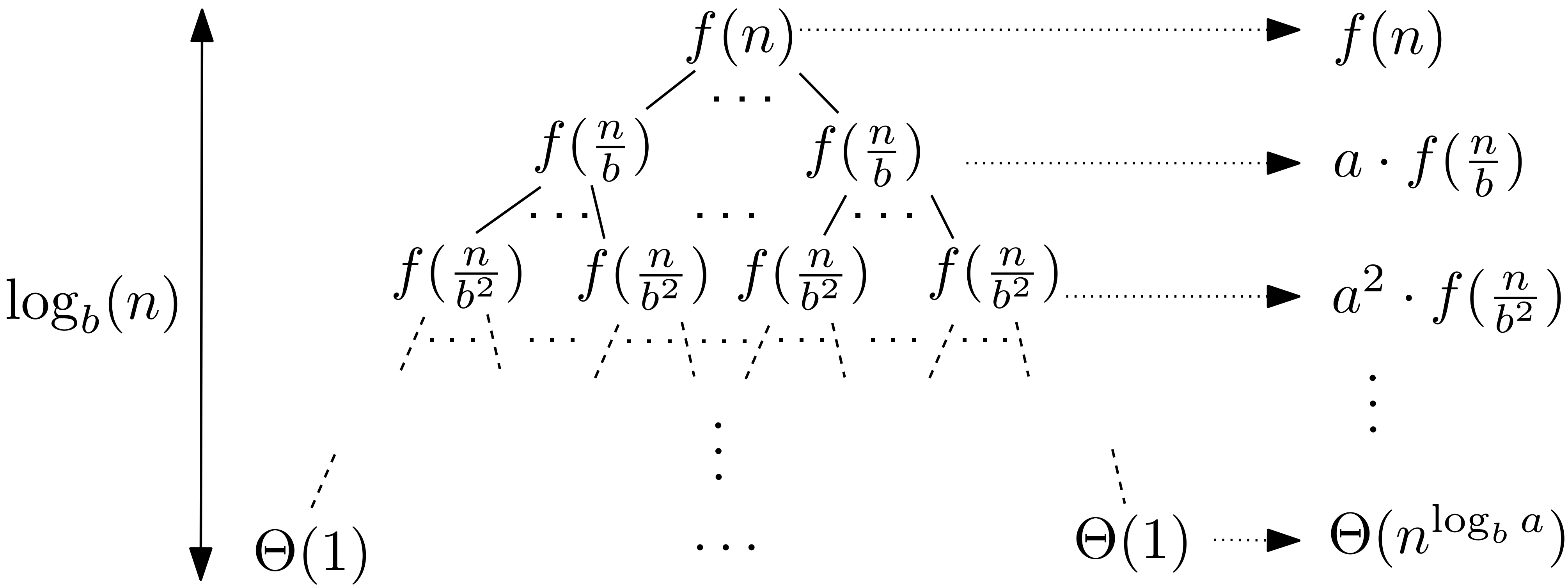
$T(n) = aT(\frac{n}{b}) + f(n)$



Recursion tree for the master method

Some intuition why the master method is actually correct

$T(n) = aT(\frac{n}{b}) + f(n)$



Total: $\Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b(n)-1} a^i \cdot f(\frac{n}{b^i})$

The substitution method

Idea: "guess" an asymptotic bound ($\mathcal{O}$, Ω) and prove it by mathematical induction

Example: $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$

Guess: $T(n) = \mathcal{O}(n \log n)$

We have to show: $\exists c > 0, n_0 \in \mathbb{N}$ such that

$T(n) \leq c \cdot n \log n$ for all $n \geq n_0$

Induction hypothesis: $T(n) \leq c \cdot n \log_2 n$ for some const. $c > 0$

Induction base: $T(k) = \Theta(1) \leq d$ for small constant k

and some constant $d > 0$ (by def.)

Induction step: $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$

$$\leq 2 \cdot (c \cdot \lfloor \tfrac{n}{2} \rfloor \log_2 \lfloor \tfrac{n}{2} \rfloor) + n \overset{?}{\leq} c \cdot n \log_2 n$$

The substitution method

Example: $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$, $\quad T(n) \stackrel{?}{=} \mathcal{O}(n \log n)$

$$\begin{aligned}
T(n) &= 2T(\lfloor \tfrac{n}{2} \rfloor) + n \\
&\leq 2 \cdot \big(c\lfloor \tfrac{n}{2} \rfloor \log_2(\lfloor \tfrac{n}{2} \rfloor)\big) + n \\
&\leq c \cdot n \log_2(\tfrac{n}{2}) + n \\
&= c \cdot n \log_2 n - c \cdot n \log_2 2 + n \\
&= c \cdot n \log_2 n - c \cdot n + n \\
&= c \cdot n \log_2 n + (1 - c) \cdot n \\
&\leq c \cdot n \log_2 n \qquad \text{for } c \geq 1
\end{aligned}$$

Choose $k \geq 3$ for induction base, $c \geq \max\{1, d\}$, $n_0 \geq 2$

from induction base

$\Rightarrow T(n) = \mathcal{O}(n \log n)$

The substitution method

Attention:
Don't use asymptotic notation in the induction step!

Example again: $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$
Guess: $T(n) = \mathcal{O}(n)$

$\Rightarrow$ Show: $\exists c > 0, n_0 \in \mathbb{N} : T(n) \leq c \cdot n$ for $n \geq n_0$

Induction hypothesis: $T(n) \leq c \cdot n$ for some const. $c > 0$

Induction step:

$$\begin{aligned} T(n) &= 2T(\lfloor \tfrac{n}{2} \rfloor) + n \\ &\leq 2(c \cdot \lfloor \tfrac{n}{2} \rfloor) + n \\ &\leq c \cdot n + n = (c+1) \cdot n \cancel{= \mathcal{O}(n)} \end{aligned}$$

WRONG !!

The substitution method

Attention:

Sometimes the "obvious" induction hypothesis doesn't work:

Example: $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1$
Guess: $T(n) = \mathcal{O}(n)$

$\Rightarrow$ Show: $\exists c > 0, n_0 \in \mathbb{N} : T(n) \leq c{\cdot}n$ for $n \geq n_0$

Induction hypothesis: $T(n) \leq c{\cdot}n$ for some const. $c > 0$

Induction step:

$$\begin{aligned} T(n) &= T(\lfloor \tfrac{n}{2} \rfloor) \ + \ T(\lceil \tfrac{n}{2} \rceil) \ + 1 \\ &\leq c \cdot \lfloor \tfrac{n}{2} \rfloor \ + \ c \cdot \lceil \tfrac{n}{2} \rceil \ + 1 \\ &= c \cdot n \ + 1 \ > \ c \cdot n \end{aligned}$$

$\Rightarrow$ Induction fails

The substitution method

Attention:
Sometimes the "obvious" induction hypothesis doesn't work:

Example: $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1$
Guess: $T(n) = \mathcal{O}(n)$

$\Rightarrow$ Show: $\exists c > 0, n_0 \in \mathbb{N} : T(n) \leq c \cdot n$ for $n \geq n_0$

Induction hypothesis: $T(n) \leq c \cdot n$ for some const. $c > 0$

Induction step:

$$\begin{aligned} T(n) &= T(\lfloor \tfrac{n}{2} \rfloor) + T(\lceil \tfrac{n}{2} \rceil) + 1 \\ &\leq c \cdot \lfloor \tfrac{n}{2} \rfloor + c \cdot \lceil \tfrac{n}{2} \rceil + 1 \\ &= c \cdot n + 1 > c \cdot n \qquad \Rightarrow \text{Induction fails} \end{aligned}$$

But: weaker hypothesis $T(n) \leq c \cdot n - d$ with $d \in \mathbb{R}$ works

The substitution method

Attention:
Sometimes the “obvious” induction hypothesis doesn’t work:

Example: $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1$
Guess: $T(n) = \mathcal{O}(n)$

$\Rightarrow$ Show: $\exists c > 0, n_0 \in \mathbb{N} : T(n) \leq c{\cdot}n$ for $n \geq n_0$

Induction hypothesis: $\cancel{T(n) \leq c{\cdot}n}$ for some const. $c > 0, d \in \mathbb{R}$
$T(n) \leq c \cdot n - d$

Induction step:

$$
\begin{aligned}
T(n) &= T(\lfloor \tfrac{n}{2} \rfloor) \ + \ T(\lceil \tfrac{n}{2} \rceil) \ + 1 \\
&\leq c \cdot \lfloor \tfrac{n}{2} \rfloor \ \boxed{-d} + \ c \cdot \lceil \tfrac{n}{2} \rceil \ \boxed{-d} + 1 \\
&= c \cdot n \ \boxed{-2 \cdot d} + 1 \ \cancel{> \ c \cdot n \qquad \Rightarrow \text{Induction fails}} \\
&= c \cdot n - d + (1 - d) \ \leq \ c \cdot n - d \quad \text{for } d \geq 1 \qquad \checkmark
\end{aligned}
$$

The substitution method

Remarks:

- advantage: more powerful than the other two methods
- disadvantage: two proofs needed for Θ ($\mathcal{O}$ and Ω)
- How to make the right guess?
  - similarity to known recurrence relations
  - recursion tree
- How to get the right approach?
  - look at additional function
  - in case of doubt, try a second time

Exercises

Resolve the following recurrence relations using the diffrent methods presented in this lecture (recall that $T(c) = \Theta(1)$ for any constant $c \geq 0$).

- $T(n) = 8T(n-2) + \Theta(1)$
- $T(n) = 8T(n-2)$
- $T(n) = T(n-2) + \Theta(n)$
- $T(n) = T(n-2) + \Theta(1)$
- $T(n) = T(n-2)$
- $T(n) = 2T(n-4) + \Theta(1)$
- $T(n) = 3T(n-4) + \Theta(1)$
- $T(n) = T(\frac{n}{2})$
- $T(n) = 2T(\frac{n}{2}) + 1$
- $T(n) = 2T(\frac{n}{2}) + n$
- $T(n) = 2T(\frac{n}{2}) + n^2$
- $T(n) = 6T(\frac{n}{2}) + 2n^3$
- $T(n) = 9T(\frac{n}{3}) + n^2$
- $T(n) = 10T(\frac{n}{6}) + \Theta(1)$

Exercises

Consider the following recurrence relations (recall that $T(c) = \Theta(1)$ for any constant $c \geq 0$). Make a guess on what should be the best possible asymptotic bounds for each of them, based on equations for which you already know the result and/or recursion trees. Try to prove your guesses using the substitution method.

- $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n$
- $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + n$
- $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + 1$
- $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + n^2$
- $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n^3$

Summary

- Goal: learn methods to asymptotically solve recurrence relations for runtime- and memory analysis
- Iterative method: convert recurrence into a summation and bound the summation to solve the recurrence
- Master method: solve recurrences of the form
  $$T(n) = aT(\tfrac{n}{b}) + f(n)$$
  (three different cases depending on $f(n)$, a, and b)
- Substitution method: guess a bound and prove it via induction (upper and lower bound separately)
- Recursion tree for visualization and intuition

Thank you for your attention.