

Wie finden wir einen geschlossenen Ausdruck für die Laufzeit?

Dafür müssen wir die Rekursionsgleichung lösen.

Wie finden wir einen geschlossenen Ausdruck für die Laufzeit?

Dafür müssen wir die Rekursionsgleichung **lösen**.

## 1. Raten+Induktion

Man rät die richtige Lösung und beweist die Korrektheit mittels vollständiger Induktion. Man benötigt Erfahrung um richtig zu raten...

## 2. Mastertheorem

Für die meisten Rekurrenzen gibt es ein allgemeines Theorem, dass die asymptotisch korrekte Laufzeit für die jeweilige Rekurrenz angibt.

# Raten+Induktion

Zuerst müssen wir die  $\mathcal{O}$ -Notation entfernen:

$$T(n) \leq \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + c_1 n & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

# Raten+Induktion

Zuerst müssen wir die  $\mathcal{O}$ -Notation entfernen:

$$T(n) \leq \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + c_1 n & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**für  $n = 2^k$ :**

Für diesen Fall können wir stattdessen die folgende Rekursionsgleichung betrachten:

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 n & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

# Raten+Induktion

Zuerst müssen wir die  $\mathcal{O}$ -Notation entfernen:

$$T(n) \leq \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + c_1 n & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**für  $n = 2^k$ :**

Für diesen Fall können wir stattdessen die folgende Rekursionsgleichung betrachten:

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 n & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

Man rät die richtige Lösung und beweist, dass durch Einsetzen, dass diese Lösung korrekt ist.

## Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

## Raten+Induktion

Ansatz:  $T(n) \leq a \log n + b$ .

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

## Raten+Induktion

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

## Raten+Induktion

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ):

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

## Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n - 1 \rightarrow n$ :

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n - 1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n - 1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n - 1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n - 1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

$$T(n) \leq T\left(\frac{n}{2}\right) + c_1$$

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n-1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n-1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

$$\begin{aligned} T(n) &\leq T\left(\frac{n}{2}\right) + c_1 \\ &\leq \left(a \log \frac{n}{2} + b\right) + c_1 \end{aligned}$$

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n - 1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n - 1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

$$\begin{aligned} T(n) &\leq T\left(\frac{n}{2}\right) + c_1 \\ &\leq \left(a \log \frac{n}{2} + b\right) + c_1 \\ &= a(\log n - 1) + b + c_1 \end{aligned}$$

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n - 1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n - 1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

$$\begin{aligned} T(n) &\leq T\left(\frac{n}{2}\right) + c_1 \\ &\leq \left(a \log \frac{n}{2} + b\right) + c_1 \\ &= a(\log n - 1) + b + c_1 \\ &= a \log n + (c_1 - a) + b \end{aligned}$$

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n-1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n-1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

$$\begin{aligned} T(n) &\leq T\left(\frac{n}{2}\right) + c_1 \\ &\leq \left(a \log \frac{n}{2} + b\right) + c_1 \\ &= a(\log n - 1) + b + c_1 \\ &= a \log n + (c_1 - a) + b \\ &\leq a \log n + b \end{aligned}$$

# Raten+Induktion

$$T(n) \leq \begin{cases} T(\frac{n}{2}) + c_1 & n > 1 \\ c_2 & \text{sonst} \end{cases}$$

**Ansatz:**  $T(n) \leq a \log n + b$ .

**Beweis.** (durch Induktion)

- ▶ **Anfang** ( $n = 1$ ): **wahr** falls  $b \geq c_2$ .
- ▶ **Induktionsschritt**  $1, \dots, n-1 \rightarrow n$ :

Angenommen Aussage wahr für  $n' \in \{1, \dots, n-1\}$ , und  $n > 1$ . Wir beweisen sie für  $n$ :

$$\begin{aligned} T(n) &\leq T\left(\frac{n}{2}\right) + c_1 \\ &\leq \left(a \log \frac{n}{2} + b\right) + c_1 \\ &= a(\log n - 1) + b + c_1 \\ &= a \log n + (c_1 - a) + b \\ &\leq a \log n + b \end{aligned}$$

Gilt falls  $a \geq c_1$ .

# Mastertheorem

## Lemma 1

Seien  $a \geq 1, b \geq 1$  und  $\epsilon > 0$  **Konstanten**. Betrachte die Rekurrenz

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

### 1. Fall:

Falls  $f(n) = \mathcal{O}(n^{\log_b(a)-\epsilon})$  gilt  $T(n) = \Theta(n^{\log_b a})$ .

### 2. Fall:

Falls  $f(n) = \Theta(n^{\log_b(a)} \log^k n)$  gilt  $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$ ,  
 $k \geq 0$ .

# Mastertheorem

Wir beweisen das Mastertheorem für den Fall  $n = b^l$ , und nehmen an, dass der nichtrekursive Fall für Problemgröße  $1$  Kosten  $1$  verursacht.

# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:

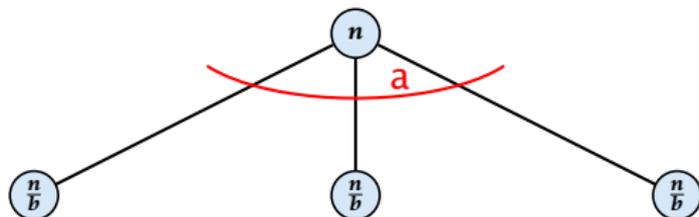
# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



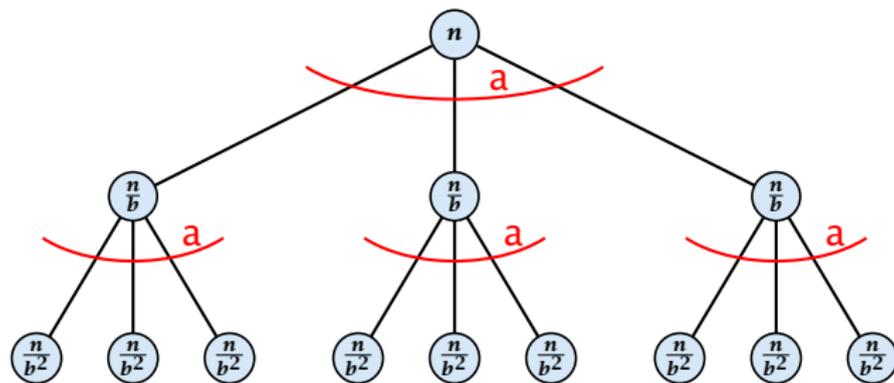
# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



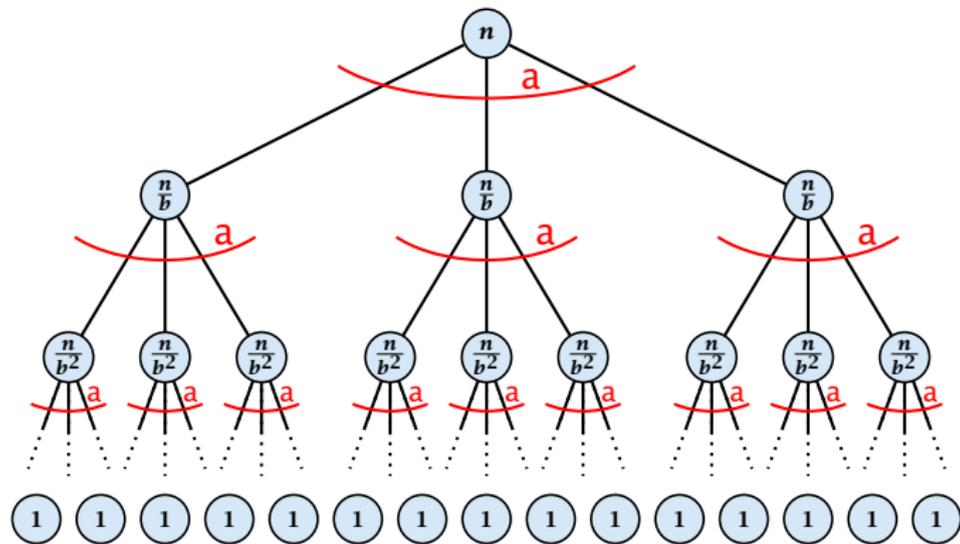
# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



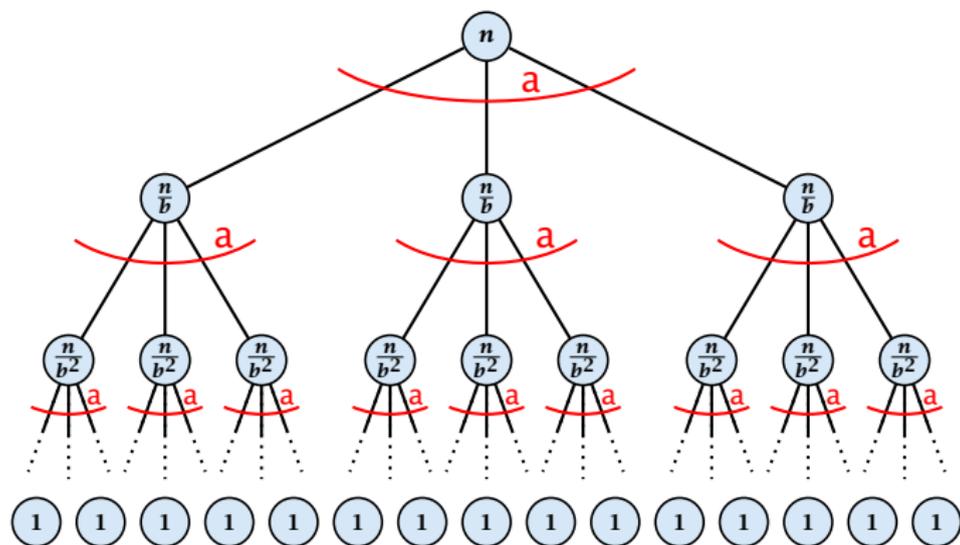
# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



# Der Rekursionsbaum

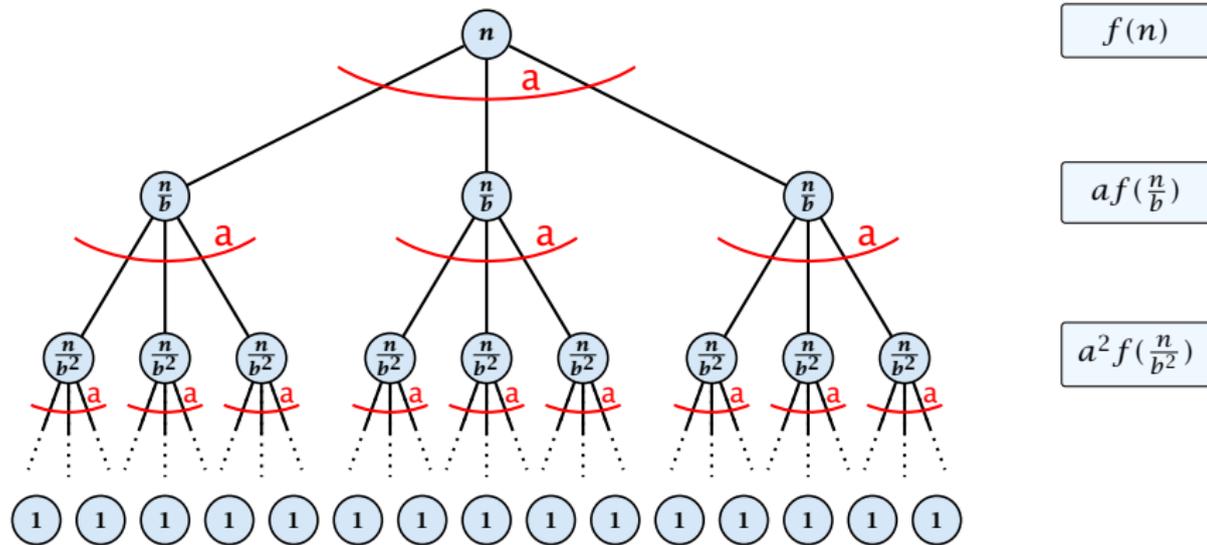
Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:





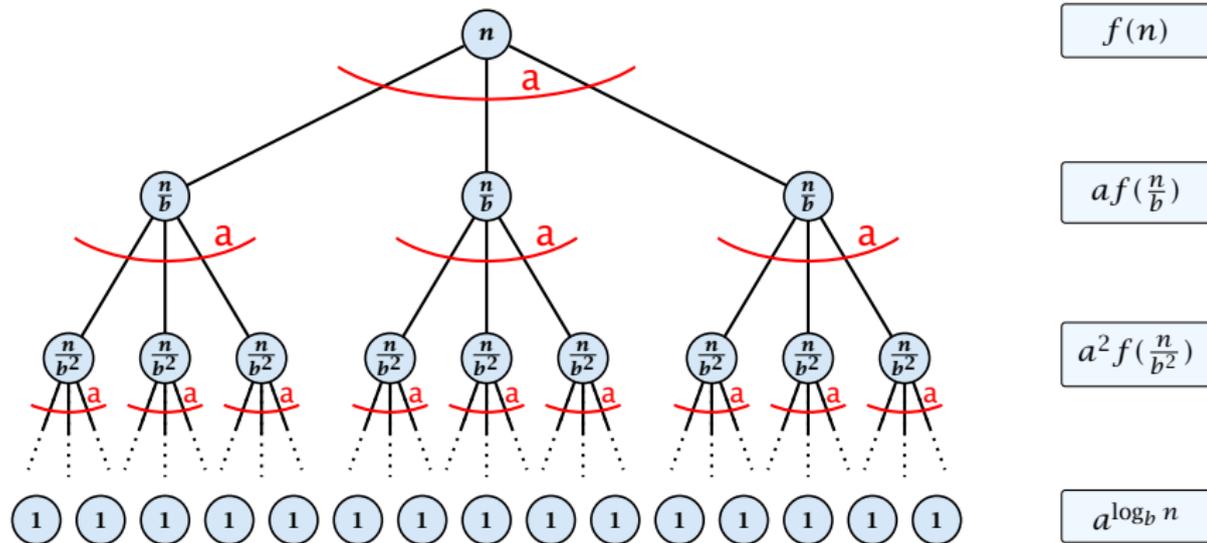
# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



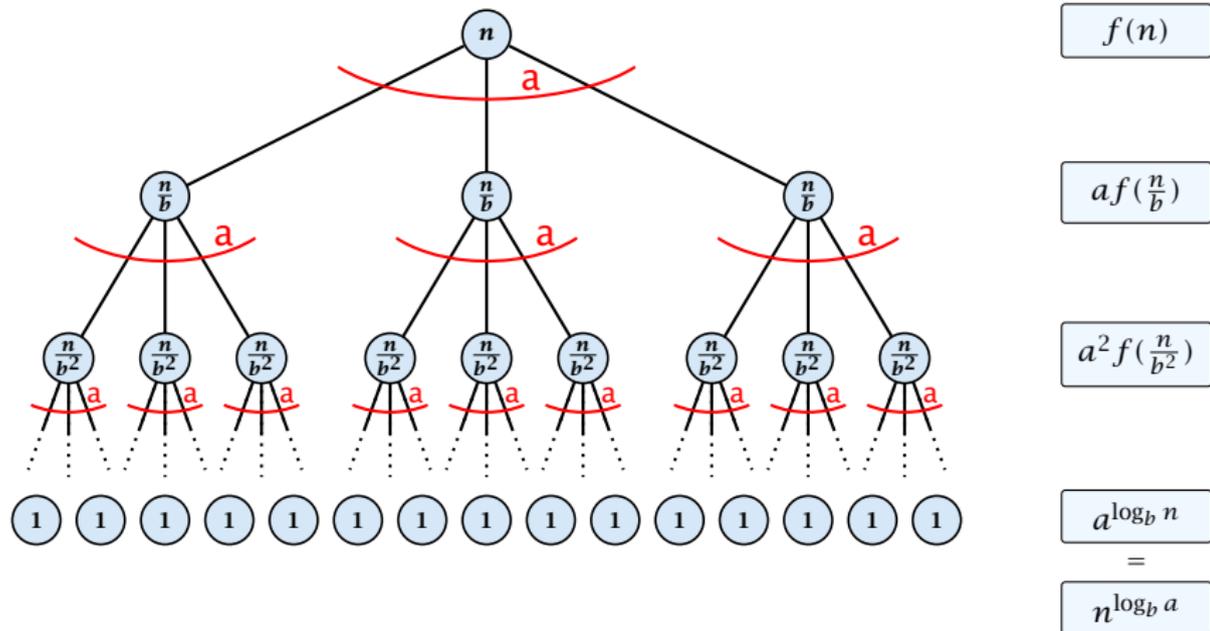
# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



# Der Rekursionsbaum

Die Laufzeit eines rekursiven Algorithmus kann durch einen Rekursionsbaum veranschaulicht werden:



# Mastertheorem

Das heißt unsere Kosten sind

$$T(n) = n^{\log_b a} + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) .$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

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

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$T(n) - n^{\log_b a} = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} = cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^{\epsilon})^i$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} = cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^{\epsilon})^i$$

$$\boxed{\sum_{i=0}^k q^i = \frac{q^{k+1} - 1}{q - 1}}$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} = cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^{\epsilon})^i$$

$$\boxed{\sum_{i=0}^k q^i = \frac{q^{k+1} - 1}{q - 1}} = cn^{\log_b a - \epsilon} (b^{\epsilon \log_b n} - 1) / (b^{\epsilon} - 1)$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} = cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^\epsilon)^i$$

$$\begin{aligned} \boxed{\sum_{i=0}^k q^i = \frac{q^{k+1} - 1}{q - 1}} &= cn^{\log_b a - \epsilon} (b^{\epsilon \log_b n} - 1) / (b^\epsilon - 1) \\ &= cn^{\log_b a - \epsilon} (n^\epsilon - 1) / (b^\epsilon - 1) \end{aligned}$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\begin{aligned} \boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} &= cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^\epsilon)^i \\ \boxed{\sum_{i=0}^k q^i = \frac{q^{k+1} - 1}{q - 1}} &= cn^{\log_b a - \epsilon} (b^{\epsilon \log_b n} - 1) / (b^\epsilon - 1) \\ &= cn^{\log_b a - \epsilon} (n^\epsilon - 1) / (b^\epsilon - 1) \\ &= \frac{c}{b^\epsilon - 1} n^{\log_b a} (n^\epsilon - 1) / (n^\epsilon) \end{aligned}$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\begin{aligned} \boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} &= cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^{\epsilon})^i \\ \boxed{\sum_{i=0}^k q^i = \frac{q^{k+1} - 1}{q - 1}} &= cn^{\log_b a - \epsilon} (b^{\epsilon \log_b n} - 1) / (b^{\epsilon} - 1) \\ &= cn^{\log_b a - \epsilon} (n^{\epsilon} - 1) / (b^{\epsilon} - 1) \\ &= \frac{c}{b^{\epsilon} - 1} n^{\log_b a} (n^{\epsilon} - 1) / (n^{\epsilon}) \end{aligned}$$

Also,

$$T(n) \leq \left( \frac{c}{b^{\epsilon} - 1} + 1 \right) n^{\log_b(a)}$$

Fall 1. Sei  $f(n) \leq cn^{\log_b a - \epsilon}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \epsilon} \end{aligned}$$

$$\begin{aligned} \boxed{b^{-i(\log_b a - \epsilon)} = b^{\epsilon i} (b^{\log_b a})^{-i} = b^{\epsilon i} a^{-i}} &= cn^{\log_b a - \epsilon} \sum_{i=0}^{\log_b n - 1} (b^\epsilon)^i \\ \boxed{\sum_{i=0}^k q^i = \frac{q^{k+1} - 1}{q - 1}} &= cn^{\log_b a - \epsilon} (b^{\epsilon \log_b n} - 1) / (b^\epsilon - 1) \\ &= cn^{\log_b a - \epsilon} (n^\epsilon - 1) / (b^\epsilon - 1) \\ &= \frac{c}{b^\epsilon - 1} n^{\log_b a} (n^\epsilon - 1) / (n^\epsilon) \end{aligned}$$

Also,

$$T(n) \leq \left( \frac{c}{b^\epsilon - 1} + 1 \right) n^{\log_b(a)} \quad \Rightarrow T(n) = \mathcal{O}(n^{\log_b a}).$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

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

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

$$T(n) - n^{\log_b a} = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \end{aligned}$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\ &= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \end{aligned}$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

$$\begin{aligned}T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\&\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\&= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \\&= cn^{\log_b a} \log_b n\end{aligned}$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

$$\begin{aligned}T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\&\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\&= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \\&= cn^{\log_b a} \log_b n\end{aligned}$$

Also,

$$T(n) = \mathcal{O}(n^{\log_b a} \log_b n)$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a}$ .

$$\begin{aligned}T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\&\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\&= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \\&= cn^{\log_b a} \log_b n\end{aligned}$$

Also,

$$T(n) = \mathcal{O}(n^{\log_b a} \log_b n)$$

$$\Rightarrow T(n) = \mathcal{O}(n^{\log_b a} \log n).$$

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$T(n) - n^{\log_b a}$$

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$T(n) - n^{\log_b a} = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)$$

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\geq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \end{aligned}$$

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\geq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\ &= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \end{aligned}$$

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$\begin{aligned}T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\geq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\ &= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \\ &= cn^{\log_b a} \log_b n\end{aligned}$$

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$\begin{aligned}T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\geq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\ &= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \\ &= cn^{\log_b a} \log_b n\end{aligned}$$

Also,

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

Fall 2. Sei  $f(n) \geq cn^{\log_b a}$ .

$$\begin{aligned}T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\geq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \\ &= cn^{\log_b a} \sum_{i=0}^{\log_b n - 1} 1 \\ &= cn^{\log_b a} \log_b n\end{aligned}$$

Also,

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

$$\Rightarrow T(n) = \Omega(n^{\log_b a} \log n).$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde. Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

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

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde. Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$T(n) - n^{\log_b a} = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde. Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k \end{aligned}$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde. Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k \end{aligned}$$

$$n = b^\ell \Rightarrow \ell = \log_b n$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k \end{aligned}$$

$$\boxed{n = b^\ell \Rightarrow \ell = \log_b n} = cn^{\log_b a} \sum_{i=0}^{\ell-1} \left(\log_b\left(\frac{b^\ell}{b^i}\right)\right)^k$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k \end{aligned}$$

$$n = b^\ell \Rightarrow \ell = \log_b n$$

$$\begin{aligned} &= cn^{\log_b a} \sum_{i=0}^{\ell-1} \left(\log_b\left(\frac{b^\ell}{b^i}\right)\right)^k \\ &= cn^{\log_b a} \sum_{i=0}^{\ell-1} (\ell - i)^k \end{aligned}$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k \end{aligned}$$

$$n = b^\ell \Rightarrow \ell = \log_b n$$

$$\begin{aligned} &= cn^{\log_b a} \sum_{i=0}^{\ell-1} \left(\log_b\left(\frac{b^\ell}{b^i}\right)\right)^k \\ &= cn^{\log_b a} \sum_{i=0}^{\ell-1} (\ell - i)^k \\ &= cn^{\log_b a} \sum_{i=1}^{\ell} i^k \end{aligned}$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$T(n) - n^{\log_b a} = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ \leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k$$

$$n = b^\ell \Rightarrow \ell = \log_b n$$

$$= cn^{\log_b a} \sum_{i=0}^{\ell-1} \left(\log_b\left(\frac{b^\ell}{b^i}\right)\right)^k \\ = cn^{\log_b a} \sum_{i=0}^{\ell-1} (\ell - i)^k$$

$$= cn^{\log_b a} \sum_{i=1}^{\ell} i^k \approx \frac{1}{k} \ell^{k+1}$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$\begin{aligned} T(n) - n^{\log_b a} &= \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ &\leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k \end{aligned}$$

$$n = b^\ell \Rightarrow \ell = \log_b n$$

$$\begin{aligned} &= cn^{\log_b a} \sum_{i=0}^{\ell-1} \left(\log_b\left(\frac{b^\ell}{b^i}\right)\right)^k \\ &= cn^{\log_b a} \sum_{i=0}^{\ell-1} (\ell - i)^k \end{aligned}$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

$$\begin{aligned} &= cn^{\log_b a} \sum_{i=1}^{\ell} i^k \\ &\approx \frac{c}{k} n^{\log_b a} \ell^{k+1} \end{aligned}$$

Fall 2. Sei  $f(n) \leq cn^{\log_b a} (\log_b(n))^k$ .

$$T(n) - n^{\log_b a} = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\ \leq c \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a} \cdot \left(\log_b\left(\frac{n}{b^i}\right)\right)^k$$

$$n = b^\ell \Rightarrow \ell = \log_b n$$

$$= cn^{\log_b a} \sum_{i=0}^{\ell-1} \left(\log_b\left(\frac{b^\ell}{b^i}\right)\right)^k$$

$$= cn^{\log_b a} \sum_{i=0}^{\ell-1} (\ell - i)^k$$

Beachte, dass  $\sum_{i=1}^{\ell} i^k \leq \ell^{k+1}$  trivial ist, und für diese obere Schranke ausreichen würde.

Für eine untere Schranke reicht auch  $\sum_{i=1}^{\ell} i^k \geq \sum_{i=\ell/2}^{\ell} i^k \geq (\ell/2)^{k+1}$ .

$$= cn^{\log_b a} \sum_{i=1}^{\ell} i^k$$

$$\approx \frac{c}{k} n^{\log_b a} \ell^{k+1}$$

$$\Rightarrow T(n) = \mathcal{O}(n^{\log_b a} \log^{k+1} n).$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integern konstanter Länge ausführen.

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integern konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ A \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ B \\ \hline \end{array}$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1 \\ \hline \end{array}$$

The diagram shows two 9-bit integers, A and B, aligned for addition. A vertical light blue box highlights the rightmost bit (the least significant bit) of both numbers, which are both 1. The label 'A' is to the right of the top row, and 'B' is to the right of the bottom row. A horizontal line is drawn under the bottom row of bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

1	1	0	1	1	0	1	0	1	$A$
1	0	0	0	1	0	0	1	1	$B$
<hr/>									
								0	

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ A \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ B \\ \hline \phantom{1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ } 1\ 0 \end{array}$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ A \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ B \\ \hline 0\ 0 \end{array}$$

The diagram illustrates the addition of two 10-bit integers, A and B. The numbers are written in binary. A horizontal line is drawn under the numbers. The result of the addition is shown below the line. The first two bits of the result are 0 and 0, which are highlighted in a light blue box. The rest of the result is not shown.

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{rcccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & A \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & B \\ \hline & & & & & & 1 & 1 & & \\ & & & & & & & 0 & 0 & \end{array}$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

1	1	0	1	1	0	1	0	1	$A$
1	0	0	0	1	0	0	1	1	$B$
<hr/>						0	0	0	

The diagram illustrates the addition of two 9-bit integers, A and B. A vertical line is drawn between the 6th and 7th bits. A light blue vertical box highlights the 7th bit of A (1), the 7th bit of B (0), and the resulting 7th bit (0). Small '1' characters are placed below the 6th, 7th, and 8th bits of the second row, indicating carry bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{rcccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & A \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & B \\ \hline & & & & & 1 & 1 & 1 & & \\ & & & & & & 0 & 0 & 0 & \end{array}$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{rcccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & A \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & B \\ \hline & & & & 0 & 1 & 1 & 1 & & \\ & & & & & 1 & 0 & 0 & 0 & \end{array}$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{rcccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & A \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & B \\ \hline & & & & 0 & 1 & 1 & 1 & & \\ & & & & & 1 & 0 & 0 & 0 & \end{array}$$

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{rcccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & A \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & B \\ \hline & & & & 0 & 1 & 0 & 0 & 0 & \end{array}$$

The diagram illustrates the addition of two 9-bit integers, A and B, to produce a 9-bit result. A vertical box highlights the 5th bit position (index 4 from the right). In this position, the bits from A and B are 1 and 1, respectively, which sum to 2 (binary 10). The carry-out from this position is 1, which is added to the 6th bit position. The carry-in to the 5th position is 0. The resulting bit in the 5th position is 0.

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

1	1	0	1	1	0	1	0	1	$A$
1	0	0	0	1	0	0	1	1	$B$
<hr/>									
			1	0	1	1	1		
			0	1	0	0	0		

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{rcccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & A \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & B \\ \hline & & & 0 & 0 & 1 & 0 & 0 & 0 & \end{array}$$

The diagram illustrates the addition of two 9-bit integers, A and B, to produce a 9-bit result. The numbers are aligned to the right. A vertical light blue box highlights the 4th bit position from the right. In this position, the bits from A and B are 1 and 0, respectively, and the resulting bit is 0. Small subscripts '1' are placed below the 4th and 5th bits of the result row, indicating carry propagation.



## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

1	1	0	1	1	0	1	0	1	$A$
1	0	0	0	1	0	0	1	1	$B$
<hr/>									
		1	0	0	1	0	0	0	

*Note: In the original image, a vertical box highlights the third bit (index 2) of both A and B, and the carry bit 1 in the result row below it.*



## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

	1	1	0	1	1	0	1	0	1	$A$
	1	0	0	0	1	0	0	1	1	$B$
	<hr/>									
	1	1	0	0	1	0	0	0		

0 0 1 1 0 1 1 1

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

1	1	0	1	1	0	1	0	1	$A$
1	0	0	0	1	0	0	1	1	$B$
<hr/>									
1	1	0	0	1	0	0	0	0	

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

	1	1	0	1	1	0	1	0	1	$A$
	1	0	0	0	1	0	0	1	1	$B$
1	0	0	1	1	0	1	1	1		
	0	1	1	0	0	1	0	0	0	

## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

	1	1	0	1	1	0	1	0	1	$A$
	1	0	0	0	1	0	0	1	1	$B$
	<hr/>									
1	0	1	1	0	0	1	0	0	0	



## Beispiel: Integermultiplikation

Angenommen wir möchten zwei  $n$ -bit Integer multiplizieren, aber unsere Register können nur Operationen auf Integer konstanter Länge ausführen.

Dafür müssen wir zunächst zwei Zahlen  $A$  and  $B$  addieren können:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ A \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ B \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Das heißt wir können zwei  $n$ -bit Integer in Zeit  $\mathcal{O}(n)$  addieren.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 10001 \\ \times 1011 \\ \hline \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 10001 \\ \times 101 \\ \hline \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 10001 \times 1011 \\ \hline 10001 \\ \phantom{10001}0000 \\ \phantom{10001}0000 \\ \phantom{10001}10001 \\ \hline \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.



## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ \boxed{1}\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\ \hline \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 10001 \times 1011 \\ \hline 10001 \\ 100010 \\ 0000000 \\ 10001000 \\ \hline 10111011 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\ \hline 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

**Laufzeit:**

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \times 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\ \hline 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

### Laufzeit:

- ▶ Zwischenergebnisse berechnen:  $\mathcal{O}(nm)$ .

## Beispiel: Integermultiplikation

Angenommen wir möchten ein  $n$ -bit Integer  $A$  und ein  $m$ -bit Integer  $B$  ( $m \leq n$ ) multiplizieren.

$$\begin{array}{r} 10001 \times 1011 \\ \hline 10001 \\ 100010 \\ 0000000 \\ 10001000 \\ \hline 10111011 \end{array}$$

- Dies nennt man auch die „Schulmethode“ für die Integermultiplikation.
- Die Zahlen der Zwischenergebnisse haben höchstens  $m + n \leq 2n$  bits.

### Laufzeit:

- ▶ Zwischenergebnisse berechnen:  $\mathcal{O}(nm)$ .
- ▶ Addieren von  $m$  Zahlen der Länge  $\leq 2n$ :  
 $\mathcal{O}((m + n)m) = \mathcal{O}(nm)$ .

# Beispiel: Integermultiplikation

**Ein rekursiver Ansatz:**

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .

# Beispiel: Integermultiplikation

Ein rekursiver Ansatz:

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .



# Beispiel: Integermultiplikation

**Ein rekursiver Ansatz:**

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .

$$\begin{array}{|c|c|c|c|} \hline b_{n-1} & \dots & & b_0 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|} \hline a_{n-1} & \dots & & a_0 \\ \hline \end{array}$$

# Beispiel: Integermultiplikation

**Ein rekursiver Ansatz:**

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .

$$\boxed{b_{n-1} \quad \cdots \quad b_{\frac{n}{2}} \quad b_{\frac{n}{2}-1} \quad \cdots \quad b_0} \times \boxed{a_{n-1} \quad \cdots \quad a_{\frac{n}{2}} \quad a_{\frac{n}{2}-1} \quad \cdots \quad a_0}$$

# Beispiel: Integermultiplikation

Ein rekursiver Ansatz:

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .

$$\begin{array}{|c|c|} \hline B_1 & B_0 \\ \hline \end{array} \times \begin{array}{|c|c|} \hline A_1 & A_0 \\ \hline \end{array}$$

# Beispiel: Integermultiplikation

Ein rekursiver Ansatz:

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .



Dann gilt

$$A = A_1 \cdot 2^{\frac{n}{2}} + A_0 \text{ und } B = B_1 \cdot 2^{\frac{n}{2}} + B_0$$

# Beispiel: Integermultiplikation

**Ein rekursiver Ansatz:**

Angenommen die Integer  $A$  und  $B$  haben Länge  $n = 2^k$ .



Dann gilt

$$A = A_1 \cdot 2^{\frac{n}{2}} + A_0 \text{ und } B = B_1 \cdot 2^{\frac{n}{2}} + B_0$$

Also,

$$A \cdot B = A_1 B_1 \cdot 2^n + (A_1 B_0 + A_0 B_1) \cdot 2^{\frac{n}{2}} + A_0 B_0$$

## Beispiel: Integermultiplikation

1 **Input:** Zahlen A, B, repräsentiert durch Bitarrays  
2 der Länge n

3 **Output:** Bitarray, das A\*B enthält

```
4  
5 mult(A, B, n)  
6     if (n == 1)  
7         return new int(A[0]*B[0]);  
8     split(A,A0,A1);  
9     split(B,B0,B1);  
10    Z2 = mult(A1,B1,n/2);  
11    Z1 = mult(A0,B1,n/2) + mult(A1,B0,n/2);  
12    Z0 = mult(A0,B0,n/2);  
13    return Z2*2n + Z1*2n/2 + Z0;
```

Die Addition + addiert Bitarrays. Das funktioniert in C++ nicht direkt, aber man kann stattdessen z.B. eine Funktion add(A,B) benutzen. Wenn man den Algorithmus nach C++ übertragen möchte ist das Speichermanagement etwas unübersichtlich, da man jedes Zwischenergebnis vor Funktionsende wieder von Hand löschen muss.

Wir erhalten folgende Rekurrenzgleichung:

$$T(n) = 4T\left(\frac{n}{2}\right) + \mathcal{O}(n) .$$

# Beispiel: Integermultiplikation

**Mastertheorem:** Rekurrenz:  $T[n] = aT(\frac{n}{b}) + f(n)$ .

- ▶ Fall 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Fall 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

# Beispiel: Integermultiplikation

**Mastertheorem:** Rekurrenz:  $T[n] = aT(\frac{n}{b}) + f(n)$ .

- ▶ Fall 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Fall 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

In unserem Fall  $a = 4$ ,  $b = 2$ , und  $f(n) = \Theta(n)$ . Also, Fall 1, da  $n = \mathcal{O}(n^{2-\epsilon}) = \mathcal{O}(n^{\log_b a - \epsilon})$ .

# Beispiel: Integermultiplikation

**Mastertheorem:** Rekurrenz:  $T[n] = aT(\frac{n}{b}) + f(n)$ .

- ▶ Fall 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Fall 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

In unserem Fall  $a = 4$ ,  $b = 2$ , und  $f(n) = \Theta(n)$ . Also, Fall 1, da  $n = \mathcal{O}(n^{2-\epsilon}) = \mathcal{O}(n^{\log_b a - \epsilon})$ .

Wir erhalten Laufzeit  $\mathcal{O}(n^2)$ .

# Beispiel: Integermultiplikation

**Mastertheorem:** Rekurrenz:  $T[n] = aT(\frac{n}{b}) + f(n)$ .

- ▶ Fall 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Fall 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

In unserem Fall  $a = 4$ ,  $b = 2$ , und  $f(n) = \Theta(n)$ . Also, Fall 1, da  $n = \mathcal{O}(n^{2-\epsilon}) = \mathcal{O}(n^{\log_b a - \epsilon})$ .

Wir erhalten Laufzeit  $\mathcal{O}(n^2)$ .

⇒ Nicht besser als „Schulmethode“.

## Beispiel: Integermultiplikation

We can use the following identity to compute  $Z_1$ :

## Beispiel: Integermultiplikation

We can use the following identity to compute  $Z_1$ :

$$Z_1 = A_1B_0 + A_0B_1$$

## Beispiel: Integermultiplikation

We can use the following identity to compute  $Z_1$ :

$$\begin{aligned}Z_1 &= A_1B_0 + A_0B_1 \\ &= (A_0 + A_1) \cdot (B_0 + B_1) - A_1B_1 - A_0B_0\end{aligned}$$

## Beispiel: Integermultiplikation

We can use the following identity to compute  $Z_1$ :

$$\begin{aligned} Z_1 &= A_1 B_0 + A_0 B_1 && = Z_2 && = Z_0 \\ &= (A_0 + A_1) \cdot (B_0 + B_1) - \underbrace{A_1 B_1} && - \underbrace{A_0 B_0} \end{aligned}$$

## Beispiel: Integermultiplikation

We can use the following identity to compute  $Z_1$ :

$$\begin{aligned} Z_1 &= A_1B_0 + A_0B_1 && = Z_2 && = Z_0 \\ &= (A_0 + A_1) \cdot (B_0 + B_1) - \underbrace{A_1B_1}_{Z_2} - \underbrace{A_0B_0}_{Z_0} \end{aligned}$$

```
1 Input: Zahlen A, B, repraesentiert durch Bitarrays
2       der Laenge n
3 Output: Bitarray, das A*B enthaelt
4
5 mult(A, B, n)
6     if (n == 1)
7         return new int(A[0]*B[0]);
8     split(A,A0,A1);
9     split(B,B0,B1);
10    Z2 = mult(A1,B1,n/2);
11    Z1 = mult(A0+A1,B0+B1,n/2)-Z0-Z2;
12    Z0 = mult(A0,B0,n/2);
13    return Z2*2^n + Z1*2^{n/2} + Z0;
```

## Beispiel: Integermultiplikation

Zeile 11 ist leider nicht korrekt, da  $A_0 + A_1$  bzw.  $B_0 + B_1$  eventuell  $n/2 + 1$  bits haben können. Wenn man eine  $n/2 + 1$ -bit Zahl  $X$  in das höchstwertige Bit  $X_{n/2}$  und die restlichen Bits  $\tilde{X}$  zerlegt kann man folgendes nutzen:

$$\begin{aligned} & \dots \\ X &= A_0 + A_1; \\ Y &= B_0 + B_1; \\ Z_1 &= X_{n/2} * Y_{n/2} * 2^n + (X_{n/2} * \tilde{Y} + Y_{n/2} * \tilde{X}) * 2^{n/2} + \text{mult}(\tilde{X}, \tilde{Y}) - Z_0 - Z_2; \\ & \dots \end{aligned}$$

Laufzeit hierfür ist  $T(n/2) + \mathcal{O}(n)$ .

# Beispiel: Integermultiplikation

Wir erhalten folgende Rekurrenz:

$$T(n) = 3T\left(\frac{n}{2}\right) + \mathcal{O}(n) .$$

Master Theorem: Rekurrenz:  $T[n] = aT\left(\frac{n}{b}\right) + f(n)$ .

- ▶ Case 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Case 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

Wir sind im Fall 1. Laufzeit  $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$ .

Eine deutliche Verbesserung der „Schulmethode“.

# Beispiel: Integermultiplikation

Wir erhalten folgende Rekurrenz:

$$T(n) = 3T\left(\frac{n}{2}\right) + \mathcal{O}(n) .$$

**Master Theorem:** Rekurrenz:  $T[n] = aT\left(\frac{n}{b}\right) + f(n)$ .

- ▶ Case 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Case 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

Wir sind im Fall 1. Laufzeit  $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$ .

Eine deutliche Verbesserung der „Schulmethode“.

# Beispiel: Integermultiplikation

Wir erhalten folgende Rekurrenz:

$$T(n) = 3T\left(\frac{n}{2}\right) + \mathcal{O}(n) .$$

**Master Theorem:** Rekurrenz:  $T[n] = aT\left(\frac{n}{b}\right) + f(n)$ .

- ▶ Case 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Case 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

Wir sind im Fall 1. Laufzeit  $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$ .

Eine deutliche Verbesserung der „Schulmethode“.

## Beispiel: Integermultiplikation

Wir erhalten folgende Rekurrenz:

$$T(n) = 3T\left(\frac{n}{2}\right) + \mathcal{O}(n) .$$

**Master Theorem:** Rekurrenz:  $T[n] = aT\left(\frac{n}{b}\right) + f(n)$ .

- ▶ Case 1:  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$        $T(n) = \Theta(n^{\log_b a})$
- ▶ Case 2:  $f(n) = \Theta(n^{\log_b a} \log^k n)$        $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

Wir sind im Fall 1. Laufzeit  $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$ .

Eine deutliche Verbesserung der „Schulmethode“.