

- Rekursion:
 - Algorithmen rufen sich selbst (rekursiv) auf.
- Rekurrenz:
 - Das Laufzeitverhalten bzw. der Speicherplatzbedarf von rekursiven Algorithmen kann in der Regel durch eine Rekursionsformel (recurrence, RF) beschrieben werden.

- Beispiel: MergeSort.
 - Sei $T(n)$ die (asymptotische) Laufzeit des MergeSort-Algorithmus für die Eingabegröße n :

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & n > 1 \end{cases}$$

Rekurrenz

3

- Bei der Lösung solcher Rekursionsformeln vernachlässigen wir oft gewisse technische Details:
 - Z.B. das Auf- und Abrunden der Größen der rekursiven Teilprobleme:
 - Beispiel: MergeSort.

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

Rekurrenz

4

- Wenn die Laufzeit für die Eingabegröße 1 in $\Theta(1)$ ist (also eine Konstante), schreiben wir in der Regel:

$$T(n) = \begin{cases} c_1 & n = 1 \\ 2T(n/2) + \Theta(n) & n > 1 \end{cases}$$

- Oder noch einfacher:

$$T(n) = 2T(n/2) + \Theta(n)$$

Rekurrenz

5

- Wir werden im folgenden drei Verfahren zum Lösen von RF kennen lernen:
 - Substitutions-Methode (Ersetzungsmethode)
 - Rekursionsbäume
 - Master-Methode (Master-Theorem)

Substitutions-Methode

6

- Die Substitutions-Methode funktioniert wie folgt:
 - Man „errate“ die Form der Lösung.
 - Man verwende (vollständige) Induktion, um die Konstanten zu finden und um zu zeigen, dass es die Lösung ist.
- Beispiel:
$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$
- Wir „glauben“, dass
$$T(n) \in O(n \log(n)) \text{ die Lösung der RF ist.}$$
- Daher versuchen wir zu zeigen, dass
$$T(n) \leq c n \log(n) \text{ für eine geeignet gewählte Konstante } c > 0 \text{ und für hinreichend große } n \text{ (} n \geq n_0 \text{)}$$

Substitutions-Methode

7

- Wir nehmen an, dass die Aussage für $\lfloor n/2 \rfloor$ gilt:

$$T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)$$

- Induktionsschluss:

$$T(n) \leq 2(c \lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n$$

$$\leq cn \log(n/2) + n$$

$$\leq cn \log(n) - cn \log(2) + n$$

$$\leq cn \log(n) - cn + n$$

$$\leq cn \log(n) \quad \text{falls } c > 1$$

Es fehlt der Induktionsanfang $n = 1$ (siehe nächste Seite)!

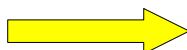
Substitutions-Methode

8

- Da die Laufzeit $T(1) = c_1$ für $n = 1$ immer größer als 0 ist, ist die Grenzbedingung für jede Konstante $c > 0$ verletzt:

$$T(1) = c_1 > c \log(1) = c * 0 = 0$$

- Der Induktionsanfang ist also nicht erfüllt.
- Man beachte, dass das asymptotische Laufzeitverhalten aber nur für alle n größer gleich einer vorgegebenen ganzen Zahl n_0 gelten muss!



Substitutions-Methode

9

- Wir müssen nur zeigen, dass

$$T(n) \leq cn \log(n) \quad \text{für alle } n \geq n_0 \text{ gilt.}$$

- Wir können also den Fall $T(2) = c_2$ (oder $T(3) = c_3$ oder $T(4) = c_4$ usw.) als Basisfall (Induktionsstart) für die vollständige Induktion wählen.
- Dann haben wir kein Problem eine geeignete Konstante anzugeben:

$$c = \max\{1, c_2\}$$

Substitutions-Methode

10

- Manchmal funktioniert die Abschätzung mittels der vollständigen Induktion nicht, obwohl man sich bezüglich der Schätzung sehr sicher ist.

- Beispiel:

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

- Wir raten, dass $T(n) = O(n)$, und wir wollen zeigen, dass $T(n) \leq cn$ ist:

$$T(n) \leq c \lfloor n/2 \rfloor + c \lceil n/2 \rceil + 1 = cn + 1$$



Substitutions-Methode

11

- Wir können das Problem lösen, in dem wir

$$T(n) \leq cn - b$$

zeigen, wobei $b > 0$ eine Konstante ist.

$$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil - 2b + 1 \\ &\leq cn - 2b + 1 \\ &\leq cn - b \quad \text{falls } b \geq 1 \end{aligned}$$

Substitutions-Methode

12

In einigen Beispielen kann man die RF erst dann lösen, wenn man eine geeignete Variablensubstitution durchführt!

$$T(n) = 2T(\sqrt{n}) + \log(n)$$

Setze $m = \log(n)$:

$$T(2^m) = 2T(2^{m/2}) + m$$

Setze $S(m) = T(2^m)$:

$$\begin{aligned} S(m) &= 2S(m/2) + m \Rightarrow S(m) \in O(m \log(m)) \\ &\Rightarrow T(n) \in O(\log(n) \log \log(n)) \end{aligned}$$

Substitutions-Methode

13

- Wie kann man die Lösung der RF „erraten“?
 - Erfahrung: Ähnlichkeit mit bereits bekannten Rekursionsformeln.
 - Man verwende Rekursionsbäume, um die asymptotische Größenordnung abzuschätzen.
 - Man verwende untere und obere Schranken, um die asymptotische Größenordnung immer weiter einzuschränken.

Rekursionsbäume

14

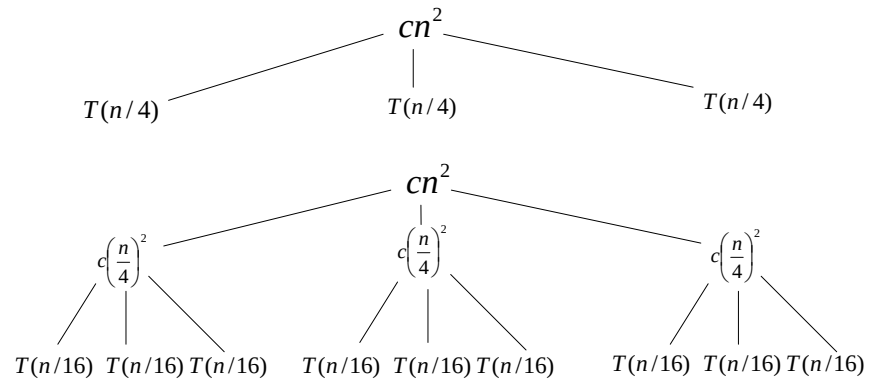
- Rekursive Algorithmen zerlegen das zu lösende Problem in Teilprobleme.
- Die Gesamtlösung wird aus den Lösungen der Teilprobleme zusammengesetzt.
- Die Zerlegung in Teilprobleme kann mittels eines Rekursionsbaumes dargestellt werden.
- In einem solchen Baum repräsentiert jeder Knoten ein Teilproblem bzw. die Kosten eines Teilproblems.
- Rekursionsbäume sind hervorragend geeignet, um die asymptotischen Laufzeiten eines rekursiven Verfahrens abzuschätzen.
- Die asymptotische Laufzeit kann dann mittels der Substitutionsmethode bewiesen werden.

Rekursionsbäume

15

- Als Beispiel betrachten wir die Rekursion:

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \Rightarrow T(n) = 3T(n/4) + cn^2$$

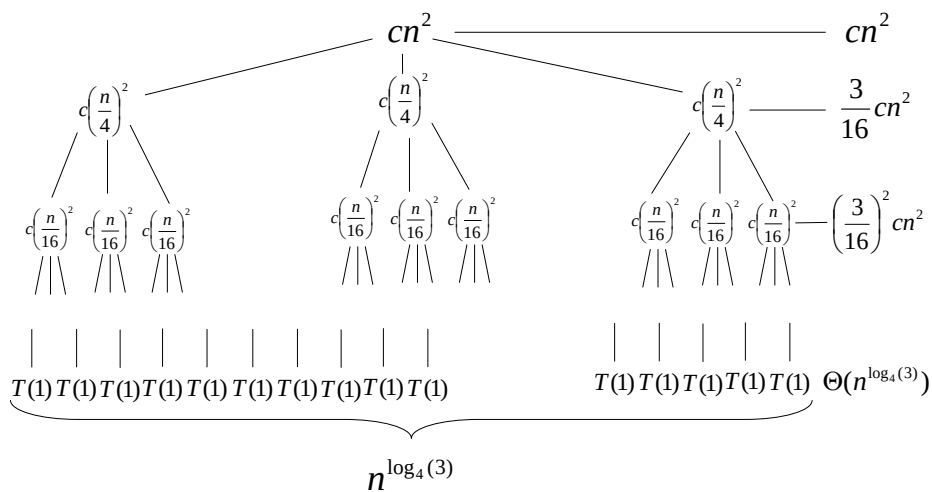


Rekursionsbäume

16

- Als Beispiel betrachten wir die Rekursion:

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \Rightarrow T(n) = 3T(n/4) + cn^2$$



Rekursionsbäume

17

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \Lambda + \left(\frac{3}{16}\right)^{\log_4(n)-1} cn^2 + \Theta(n^{\log_4(3)})$$

$$T(n) = cn^2 \sum_{i=0}^{\log_4(n)-1} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4(3)})$$

$$T(n) = \frac{(3/16)^{\log_4(n)} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4(3)})$$

$$T(n) \leq 2cn^2 + \Theta(n^{\log_4(3)}) \quad \text{für } n \geq 4$$

$$T(n) \in O(n^2)$$

Wir vermuten daher, dass $T(n) = O(n^2)$ ist, und zeigen nun mittels der Substitutionsmethode, dass $T(n) \leq dn^2$ für ein geeignetes $d > 0$.

Rekursionsbäume

18

$$T(n) \leq 3T(\lfloor n/4 \rfloor) + cn^2$$

$$\leq 3d \lfloor n/4 \rfloor^2 + cn^2$$

$$\leq 3d(n/4)^2 + cn^2$$

$$= \frac{3}{16}dn^2 + cn^2$$

$$\leq dn^2$$

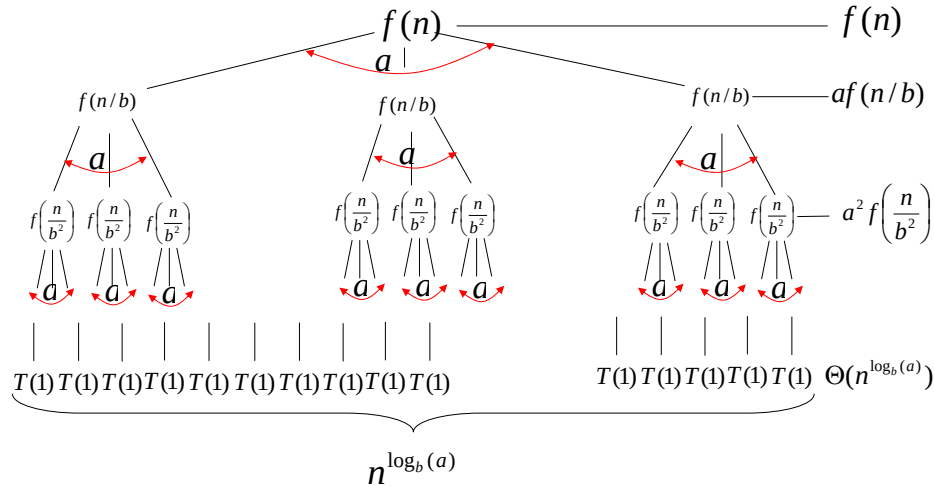
Die letzte Abschätzung gilt für alle d mit $d \geq (16/13)c$.

Master-Methode

19

- Wir betrachten im folgenden Rekursionen der Form

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \quad b > 1 \quad \forall n \geq n_0 : f(n) \geq 0$$



Master-Theorem

20

Summieren wir nun die Kosten aller Schichten des Baumes auf, so erhalten wir die folgenden Gesamtkosten.

$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \dots + a^{\log_b(n)-1} f\left(\frac{n}{b^{\log_b(n)-1}}\right) + \Theta(n^{\log_b(a)})$$

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

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

Wir haben implizit in den obigen Berechnungen immer vorausgesetzt, dass n durch die Potenzen von b teilbar ist, d.h., dass n eine Potenz von b ist. Diese Voraussetzung behalten wir für die folgenden Rechnungen und Beweise bei. Die Berechnungen und Beweise für beliebige ganze Zahlen $n \in \mathbb{N}$ können Sie im Lehrbuch „Introduction to Algorithms“ von Cormen et al. nachlesen.

Master-Theorem

21

Wir diskutieren im folgenden die potentiellen Lösungen von Rekurrenzen der Form $T(n) = a T(n/b) + f(n)$ in Abhängigkeit vom asymptotischen Verhalten der Funktion $f(n)$:

Hierbei betrachten wir zuerst den Fall, dass

$$f(n) = O(n^{\log_b(a)-\varepsilon}) \quad \text{für eine Konstante} \quad \varepsilon > 0$$

$$\Rightarrow f\left(\frac{n}{b^i}\right) = O\left(\left(\frac{n}{b^i}\right)^{\log_b(a)-\varepsilon}\right)$$

$$\Rightarrow g(n) = \sum_{i=0}^{\log_b(n)-1} a^i f\left(\frac{n}{b^i}\right) \in O\left(\sum_{i=0}^{\log_b(n)-1} a^i \left(\frac{n}{b^i}\right)^{\log_b(a)-\varepsilon}\right)$$

Master-Theorem

22

$$\begin{aligned} \sum_{i=0}^{\log_b(n)-1} a^i \left(\frac{n}{b^i}\right)^{\log_b(a)-\varepsilon} &= n^{\log_b(a)-\varepsilon} \sum_{i=0}^{\log_b(n)-1} \left(\frac{ab^\varepsilon}{b^{\log_b(a)}}\right)^i \\ &= n^{\log_b(a)-\varepsilon} \sum_{i=0}^{\log_b(n)-1} b^{\varepsilon i} \\ &= n^{\log_b(a)-\varepsilon} \frac{b^{\varepsilon \log_b(n)} - 1}{b^\varepsilon - 1} \\ &= cn^{\log_b(a)-\varepsilon} \frac{n^\varepsilon - 1}{b^\varepsilon - 1} \in O(n^{\log_b(a)}) \end{aligned}$$

Da b und ε Konstanten sind, folgt aus der obigen Abschätzung:

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

Master-Theorem

23

Wir betrachten nun den zweiten Fall und nehmen an, dass

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

$$\Rightarrow f\left(\frac{n}{b^i}\right) = \Theta\left(\left(\frac{n}{b^i}\right)^{\log_b(a)}\right)$$

$$\Rightarrow g(n) \in \Theta\left(\sum_{i=0}^{\log_b(n)-1} a^i \left(\frac{n}{b^i}\right)^{\log_b(a)}\right)$$

Master-Theorem

24

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

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

Master-Theorem

25

Wir diskutieren im folgenden den dritten Fall und nehmen an, dass

$$af(n/b) \leq cf(n) \quad \text{für eine Konstante } c < 1 \text{ und für alle } n \geq b$$

$$\Rightarrow f(n/b) \leq \frac{c}{a} f(n) \quad \text{Wenden wir diese Abschätzung } i \text{ mal iterativ an, so erhalten wir}$$

$$\Rightarrow f(n/b^i) \leq \left(\frac{c}{a}\right)^i f(n)$$

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

Nehmen wir jetzt ferner an, dass

$$f(n) \in \Omega(n^{\log_b(a)+\varepsilon}) \quad \text{für ein } \varepsilon > 0, \text{ so gilt: } T(n) \in \Theta(f(n))$$

Master-Theorem

26

Satz [1]:

Seien $a \geq 1$ und $b > 1$ Konstanten, sei $f(n)$ eine auf den exakten Potenzen von b definierte positive Funktion und sei $T(n)$ die Rekursion der Form

$$T(n) = aT(n/b) + f(n)$$

wobei n/b entweder als $\lfloor n/b \rfloor$ oder als $\lceil n/b \rceil$ interpretiert werden kann.

- (1) Ist $f(n) = O(n^{\log_b(a)-\varepsilon})$ für ein $\varepsilon > 0$, so gilt: $T(n) = \Theta(n^{\log_b(a)})$
- (2) Ist $f(n) = \Theta(n^{\log_b(a)})$ dann ist: $T(n) = \Theta(n^{\log_b(a)} \log(n))$
- (3) Ist $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ für ein $\varepsilon > 0$, und ist $af(n/b) \leq cf(n)$ für eine Konstante $c < 1$ und genügend großes n , so gilt:

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

Anwendung des MTs auf MergeSort

27

Die Laufzeit des MergeSort-Algorithmus wird durch die folgende RF beschrieben:

$$T(n) = 2T(n/2) + \Theta(n)$$

Die Konstanten a und b sind gleich 2 und die Funktion f(n), die den Zeitaufwand für das Mischen (Merge) der sortierten Teilfolgen angibt, ist in $\Theta(n)$.

Da $\log_b(a) = \log_2(2) = 1$ ist,

$$f(n) \in \Theta(n^{\log_b(a)}) = \Theta(n^1) = \Theta(n)$$

folgt aus Fall (2) des Master-Theorems

Satz [2]:

Die Laufzeit T(n) des Algorithmus MergeSort ist in $\Theta(n \log n)$:

$$T(n) \in \Theta(n \log(n))$$

Beispielanwendung des MTs

28

Als zweites Beispiel betrachten wir die Rekursion

$$T(n) = 9T(n/3) + n$$

Es gilt: $a = 9$, $b = 3$, $f(n) = n$, $\log_b(a) = \log_3(9) = 2$.

$$n^{\log_b(a)} = n^2 \in \Theta(n^2)$$

$$f(n) \in O(n^{\log_b(a) - \varepsilon}) \quad \text{wobei } \varepsilon = 1 \text{ ist.}$$

Dann folgt aus Fall (1) des Master-Theorems

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

Beispielanwendung des MTs

29

Als drittes Beispiel betrachten wir die Rekursion

$$T(n) = 3T(n/4) + n \log(n)$$

Es gilt: $a = 3$, $b = 4$, $f(n) = n \log(n)$, $\log_b(a) = \log_4(3) = 0.793$.

$$f(n) \in \Omega(n) = \Omega(n^{\log_4(3)+\varepsilon}) \quad \text{wobei } \varepsilon \approx 0.207 \text{ ist.}$$

Da für große n gilt: $3 f(n/4) = 3 (n/4) \log(n/4) \leq (3/4) n \log(n) = (3/4) f(n)$,
folgt aus Fall (3) des Master-Theorems

$$T(n) \in \Theta(f(n)) = \Theta(n \log(n))$$