Termine

Termine für mündliche Prüfungen

- **Montag 25. Februar**
- **Dienstag 01. April**

Wiederholung

Matroide

- Greedy Algorithmus für MST
- MST Algorithmus berechnet Kreismatroid.
- Def. Matroid M=(S,U)
  - Vererbbarkeit
  - Ergänzungseigenschaft
- Basen eines Matroids haben gleiche Kardinalität
- Greedy-Matroid liefert Basis minimalen Gewichts
  - Greedy-Wahl: $\exists$ optimale Basis mit Element x kleinsten Gewichts
  - Subprobleme: Löse mit x verträgliche Submatroide

Rekursionsgleichungen

Ziel: Lösen von Rekursionsgleichungen

- $T(n) = a*T(n/b) + f(n)$ bzw.
- $x_n = a_1*x_{n-1} + \ldots + a_k*x_{n-k} + b_k$

Def.: $x_n = a_1*x_{n-1} + \ldots + a_k*x_{n-k} + b_k$
mit Anfangsbedingungen

$x_i = b_i$ für $0,\ldots,k-1$

heißt lineare Rekursionsgleichung k-ter Ordnung.
Rekursionsgleichung ist homogen $\Leftrightarrow$ $b_k=0$.

Lineare homogene Gleichung 1. Ordnung

Sei $x_n = ax_{n-1}$ für $n \geq 1$ und $x_0=b_0$.
$\Rightarrow x_n = a^2x_{n-2} = a^3x_{n-3} = \ldots = a^nx_0 = a^nb_0$

Bsp.: Verzinse 1000€ mit jährlich 4%.

- $b_0=1000$, $a=1,04$.
  $\Rightarrow x_n = (1,04)^n*1000$.
- Nach wieviel Jahren besitzt man 1 Million?
  $(1,04)^n*1000 \geq 10^6 \Rightarrow n \geq \log_{1,04}10^3 \approx 176$.

Inhomogene Rekursionen 1. Ordnung

Satz: Sei $x_n = a*x_{n-1}+b_1$ und $x_0=b_0$. Dann gilt

$$x_n = \begin{cases} b_0 a^n + b_1 \frac{a^n-1}{a-1} & \text{falls } a \neq 1 \\ b_0 + n b_1 & \text{sonst} \end{cases}$$

Iterationsmethode liefert

$x_n = ax_{n-1} + b_1$

$= a(ax_{n-2} + b_1) + b_1 = a^2x_{n-2}+b_1(1+a)$

$= a^3x_{n-3} + b_1(1+a+a^2)$

$\vdots$

$= a^nb_0 + b_1(1+a+a^2+\ldots+a^{n-1})$

$1+a+\ldots+a^{n-1} = (a^n-1)/(a-1)$ für $a\neq1$

n sonst

Millionär durch Zusatzzahlung?

- Startkapital 1000 €, jährlich 4% Zinsen
- Zusätzliche jährliche Zahlung 100€
- $a=1{,}04$, $b_1=100$, $b_0=1000$

  $\Rightarrow x_n = 1000*(1{,}04)^n + 100*(1{,}04^n-1)/(0{,}04)$
  $= (1{,}04)^n(10^3+10^4/4) - 10^4/4$
- $x_n \geq 10^6$ für $n>144$.

- Rekursionsgleichungen höheren Grades:
  Nächste Vorlesung mittels Erzeugendenfunktionen.

Das Master-Theorem

Satz (Master-Theorem):
Sei $T(n) = a*T(n/b) + f(n)$, wobei $a,b \geq 1$. Dann gilt:

1. **Falls $f(n)=\mathcal{O}(n^{\log_b a - \epsilon})$, $\epsilon>0$:**
   - **$T(n) = \Theta(n^{\log_b a})$**
2. **Falls $f(n)=\Theta(n^{\log_b a})$:**
   - **$T(n) = \Theta(n^{\log_b a} \log n)$**
3. **Falls $f(n)=\Omega(n^{\log_b a + \epsilon})$ und $a*f(n/b) \leq cf(n)$, $c<1$**
   - **$T(n) = \Theta(f(n))$.**

Man kann zeigen:
n/b kann als $\lfloor n/b \rfloor$ oder $\lceil n/b \rceil$ interpretiert werden.

Bereits bekannte Anwendungen

Mergesort, FFT:

$$T(n) = 2*T(n/2) + cn$$

- a=b=2, f(n) = cn
- $n^{\log_b a} = n$, d.h. Fall 2: $f(n) = \mathcal{O}(n^{\log_b a})$

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

Multiplikation großer Zahlen:

$$T(n) = 3*T(n/2) + cn$$

- a=3, b=2, f(n) = cn
- $n^{\log_b a} = n^{\log_2 3} \approx n^{1.58}$ d.h.

  Fall 1: $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ für $0<\epsilon<\log_2 3-1$.

  $\Rightarrow T(n) = \Theta(n^{\log_2 3})$

Weitere Anwendungen

Binäre Suche:

$$T(n) = T(n/2) + c$$

- a=1, b=2, f(n)=c.
- $n^{\log_b a} = 1$. Fall 2: $f(n) = \mathcal{O}(n^0) = \mathcal{O}(n^{\log_b a})$

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

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

- $n^{\log_b a} = n^{\log_4 3} \approx n^{0.79}$, d.h.
  - Fall 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ für $0<\epsilon\leq 1-\log_4 3$.
  - $a*f(n/b) = 3* n/4 * \log(n/4) \leq ¾*n \log n = c(n \log n)$ für $c=¾<1$

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

Nicht-Anwendbarkeit Master-Theorem

$$T(n) = 4*T(n) + n \log n$$

- $n^{\log_b a} = n$
- $f(n) = \Omega(n^{\log_b a}) = \Omega(n)$, aber **nicht** $f(n) = \Omega(n^{\log_b a + \epsilon})$
- Beachte: $f(n)/n = \log n = o(n^{\epsilon})$ für jedes $\epsilon > 0$
  - Fall 3 nicht anwendbar.
  - Fall 2 ebenfalls nicht, da $f(n) \neq \mathcal{O}(n)$

- Lücken zwischen Fall 1 und 2, bzw. Fall 2 und 3
- Master-Theorem nicht für jede Gleichung der Form

$$T(n) = a*T(n/b) + f(n) \text{ anwendbar.}$$

Beweis des Master-Theorems

Vereinfachende Annahme: $n=b^i$.

- Kann auch für allgemeine n gezeigt werden.

Satz: Sei T(n) = a*T(n/b) + f(n) und T(1) = Θ(1). Dann gilt:

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

$$\begin{aligned} aT(\frac{n}{b}) + f(n) &= a(aT(\frac{n}{b^2}) + f(\frac{n}{b})) + f(n) = a^2 T(\frac{n}{b^2}) + af(\frac{n}{b}) + f(n) \\ &= a^3 T(\frac{n}{b^3}) + a^2 f(\frac{n}{b^2}) + af(\frac{n}{b}) + f(n) \\ &\vdots \\ &= a^{\log_b n} T(\frac{n}{b^{\log_b n}}) + \sum_{i=0}^{\log_b n - 1} a^i f(\frac{n}{b^i}) \\ &= \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(\frac{n}{b^i}) \end{aligned}$$

Interpretation des Satzes

Rekursionsbaum:

- Wurzel hat Kosten f(n) und *a* Kinder mit Kosten jeweils f(n/b).
- Jedes dieser Kinder
  - a Kinder mit Kosten $f(n/b^2)$.
  - Insgesamt a^2 Kinder mit Kosten insgesamt $a^2*f(n/b^2)$
- In Tiefe i:
  - a^i Kinder mit Kosten $a^i*f(n/b^i)$
  - Summe über die Kosten der inneren Knoten:
    Kosten: Aufsplittens in Subprobleme & Kombination der Lösungen
- Blätter in Tiefe $i=\log_b n$
  - Kosten für Blätter $\Theta(n^{\log_b a})$: Kosten des Lösens der Subprobleme

Drei Fälle des Master-Theorems: Kosten werden dominiert von

(1) Kosten in den Blättern: $\Theta(n^{\log_b a})$
(2) gleichverteilt über alle Ebenen: $\Theta(n^{\log_b a})*\log_b n = \Theta(n^{\log_b a} \log n)$
(3) Kosten in der Wurzel: $\Theta(f(n))$

Beschränken der Summe

Lemma: Sei $g(n) = \sum_{i=0}^{\log_b n - 1} a^i f(\frac{n}{b^i})$

1. **Falls f(n)=$\mathcal{O}(n^{\log_b a - \epsilon})$, ϵ>0:**
   - **g(n) = $\mathcal{O}(n^{\log_b a})$**
2. **Falls f(n)=$\Theta(n^{\log_b a})$:**
   - **g(n) = $\Theta(n^{\log_b a} \log n)$**
3. **Falls a*f(n/b) $\leq$ cf(n), c<1**
   - **g(n) = Θ(f(n)).**

Korollar Master-Theorem:

1. **T(n) = $\Theta(n^{\log_b a})$ + g(n) = $\Theta(n^{\log_b a})$**
2. **T(n) = $\Theta(n^{\log_b a})$ + g(n) = $\Theta(n^{\log_b a} \log n)$**
3. **T(n) = $\Theta(n^{\log_b a})$ + g(n) = Θ(f(n)) wegen f(n)=$\Omega(n^{\log_b a + \epsilon})$ (Übung)**

Beweis des Lemmas

Beweisen nur Fall 2+3, Fall 1 ist Übungsaufgabe.
Fall 2:

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

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

Fall 3: g(n) = Ω(f(n)) folgt für i=0.

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

$$g(n) \leq \sum_{i=0}^{log_b n-1} c^i f(n) \leq f(n) \sum_{i=0}^{\infty} c^i = f(n) \frac{1}{1-c} = \mathcal{O}(f(n)).$$