

Zusammenfassung Informatik III

© Tim Baumann, <http://timbaumann.info/uni-spicker>

Abkürzung. WC/BC/AC steht für Worst/Average/Best Case.

Algorithmus (Insertion Sort). BC: $O(n)$; AC, WC: $O(n^2)$

Notation. $\mathcal{F} := \{f : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}\}$. Für $f \in \mathcal{F}$ ist

$$\begin{aligned} O(f) &:= \{g \in \mathcal{F} \mid \exists c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\} \\ \Omega(f) &:= \{g \in \mathcal{F} \mid \exists c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\} \\ o(f) &:= \{g \in \mathcal{F} \mid \forall c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\} \\ \omega(f) &:= \{g \in \mathcal{F} \mid \forall c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\} \\ \Theta(f) &:= \{g \in \mathcal{F} \mid \exists c_1, c_2 > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : \\ &\quad c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)\} = O(f) \cap \Omega(f) \end{aligned}$$

Satz. Seien $0 < \alpha < \beta$, $0 < a < b$ und $1 < A < B$. Betrachte

- $f_1(n) := \log \log n$
- $f_2(n) := (\log n)^\alpha$
- $f_3(n) := (\log n)^\beta$
- $f_4(n) := n^a$
- $f_5(n) := n^\alpha (\log n)^\alpha$
- $f_6(n) := n^b (\log n)^\alpha$
- $f_7(n) := n^b$
- $f_8(n) := n^a$
- $f_9(n) := A^n \cdot n^a$
- $f_{10}(n) := A^n \cdot n^b$
- $f_{11}(n) := B^n$

Es gilt: $f_i \in o(f_{i+1})$ für $i = 1, \dots, 10$.

Def (RAM). Die Random Access Access Machine besitzt eine unendlich lange Liste von aufsteigend nummerierten Speicherzellen $R[0], R[1], \dots$, die jeweils eine ganze Zahl beinhalten und einen Programmzähler. Sie kann mittels der folgenden Sprache programmiert werden:

$\langle \text{Zieladresse} \rangle ::= \langle \text{Adresse} \rangle \mid R[\langle \text{Adresse} \rangle]$

$\langle \text{Operand} \rangle ::= \langle \text{Literal} \rangle \mid R[\langle \text{Adresse} \rangle]$

$\langle \text{Befehl} \rangle ::= \langle \text{Zieladresse} \rangle \text{ ‘:=’ } \langle \text{Operand} \rangle \odot \langle \text{Operand} \rangle$
| ‘if’ $\langle \text{Operand} \rangle \bowtie \langle \text{Operand} \rangle$ ‘goto’ $\langle \text{Label} \rangle$

$\langle \text{Programm} \rangle ::= \langle \text{Befehl} \rangle \text{ ‘;’ } \langle \text{Programm} \rangle \mid \text{‘End’}$

wobei $\odot \in \{+, -, *, \div\}$ und $\bowtie \in \{<, \leq, =, \geq, >, \neq\}$. Diese einfache Grammatik lässt sich auch für unbedingte Sprünge nutzen (mittels Bedingung $0 = 0$). Ein Sprung über das Ende des Programms hinaus lässt das Programm anhalten. Per Konvention steht die Größe der Eingabe in der Speicherzelle $R[1]$, während die tatsächliche Eingabe in $R[2], \dots, R[R[1] + 1]$ abgelegt wird.

Def. Ein **Graph** ist ein Tupel (V, E) , wobei V eine endliche Mengen von **Knoten** und $E \subset V \times V$ die Menge der **Kanten** ist.

Def. Eine Zufallsvariable auf einem Wahrscheinlichkeitsraum $(\Omega, \mathfrak{A}, \mathbb{P})$ ist eine Borel-messbare Funktion $X : \Omega \rightarrow \mathbb{R}$.

Def. Der Erwartungswert einer Zufallsvariable X auf einem (diskreten) Wahrscheinlichkeitsraum $(\Omega, \mathfrak{A}, \mathbb{P})$ ist

$$\mathbb{E} := \int_{\Omega} X d\mathbb{P} = \sum_{\omega \in \Omega} \mathbb{P}(\{\omega\}) \cdot X(\omega).$$

Bemerkung. Es gilt für $|x| < 1$: $\sum_{k=1}^{\infty} kx^{k-1} = \frac{1}{(1-x)^2}$.

Algorithmus. Zwei sortierte Folgen der Gesamtlänge n können in $O(n)$ Zeit gemischt werden.

Algorithmus (Mergesort). BC, AC, WC: $O(n \log n)$

Satz (Master-Theorem). Seien a, b, c, k, N reelle Zahlen mit $a, c > 0$, $k \geq 0$, $b, N \in \mathbb{N}$ und $b \geq 2$ und sei $T : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ eine Funktion, die folgende Rekursionsungleichung erfüllt:

$$T(n) \leq \begin{cases} c, & \text{für } n \leq N \\ cn^k + aT(\lceil n/b \rceil), & \text{für } n > N \end{cases}$$

Sei ferner $\lambda := \log_b a$. Dann gilt

$$T(n) = \begin{cases} O(n^k), & \text{falls } \lambda < k \\ O(n^k \log n), & \text{falls } \lambda = k \\ O(n^\lambda), & \text{falls } \lambda > k. \end{cases}$$

Satz. Seien a, b, c, k, N reelle Zahlen mit $a, c > 0$, $k \geq 0$, $b, N \in \mathbb{N}$ und $b \geq 2$ und sei $T : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ eine Funktion, die folgende Rekursionsungleichung erfüllt:

$$T(n) \geq \begin{cases} c, & \text{für } n \leq N \\ cn^k + aT(\lceil n/b \rceil), & \text{für } n > N \end{cases}$$

Sei ferner $\lambda := \log_b a$. Dann gilt

$$T(n) = \begin{cases} \Omega(n^k), & \text{falls } \lambda < k \\ \Omega(n^k \log n), & \text{falls } \lambda = k \\ \Omega(n^\lambda), & \text{falls } \lambda > k. \end{cases}$$

Satz (Karatsuba und Ofman). Zwei n -stellige Zahlen können in $O(n^{\log_2 3})$ Zeit multipliziert werden.

Def. Für $\beta \in [\frac{1}{2}, 1)$ ist ein β -Splitter eine Funktion, die aus einer List von n Schlüsseln einen Schlüssel auswählt, sodass höchstens je βn Schlüssel der Liste größer bzw. kleiner sind.

Satz (Selektion). Gegeben seien eine Menge X von n Elementen aus einem total geordneten Universum und eine ganze Zahl k mit $1 \leq k \leq n$. Dann können wir (deterministisch) in $O(n)$ Zeit das k -kleinste Element aus X bestimmen.

Algorithmus (Quicksort). BC, AC: $O(n \log n)$, WC: $O(n^2)$

Algorithmus (Heapsort). Inplace, BC/AC/WC: $O(n \log n)$

Satz. Jeder deterministische vergleichsbasierte Sortieralgorithmus hat im RAM-Modell eine Worst-Case-Laufzeit von $O(n \log n)$. Wenn alle Permutation mit gleicher Wkt. auftreten, gilt dies auch für die mittlere Laufzeit.

Satz. Jeder randomisierte vergleichsbasierte Sortieralgorithmus hat im RAM-Modell eine erwartete Laufzeit von $\Omega(n \log n)$ auf WC-Eingaben der Länge n .

Satz (Sortieren durch Zählen). n ganze Zahlen im Bereich $\{0, \dots, m-1\}$ können in Zeit $O(n+m)$ sortiert werden.

Satz (Radix-Sort). n ganze Zahlen im Bereich $\{0, \dots, 10^k - 1\}$ können in Zeit $O(nk)$ sortiert werden.

Satz. n Strings mit insgesamt N Zeichen aus dem Alphabet $\{0, \dots, m-1\}$ können in $O(n+m+N)$ Zeit sortiert werden.

Satz (0-1-Prinzip). Sortiert ein Vergleichsnetzwerk für n Schlüssel alle 0-1-Tupel der Länge n korrekt, dann sortiert es alle Tupel der Länge n korrekt, ist also ein Sortiernetzwerk.

Satz. Für jede Zweierpotenz n gibt es ein Sortiernetzwerk für n Schlüssel mit Tiefe $\log_2 n(1 + \log_2 n)/2$.

	BinHeap	FibHeap
insert	$O(\log n)$	$O(1)$
delete	$O(\log n)$	$O(\log n)$
find_min	$O(1)$	$O(1)$
decrease_key	$O(\log n)$	$O(1)$

Satz. Der Grad jedes Knoten in einem Fibonacci-Heap mit n Knoten ist $O(\log n)$

Satz (amortisierte Kosten des Fibonacci-Heaps). Eine Folge von r insert-, find_min- und decrease_key- und $n \leq r$ delete-Operationen auf einem am Anfang leeren Fibonacci-Heap können in $O(r + n \log r)$ Zeit ausgeführt werden.

Satz. AVL-Bäume unterstützen alle Operationen einer Prioritätswarteschlange und eines Wörterbuchs in Zeit $O(\log n)$, wobei n die Anzahl der gespeicherten Tupel ist.

Satz. Seien a und b ganzzahlige Konstanten mit $a \geq 2$ und $b \geq 2a - 1$. Dann unterstützen (a, b) -Bäume alle Operationen einer Prioritätswarteschlange und eines Wörterbuchs in Zeit $O(\log n)$, wobei n die Anzahl der gespeicherten Tupel ist.

Def. Der Belegungsfaktor einer Hashtabelle ist $\alpha := n/s$, wobei n die Anzahl der Schlüssel und s die Größe der Hashtabelle ist.

Satz. Eine Wörterbuchoperation auf einer Hashtabelle mit Belegungsfaktor α kann unter den Annahmen (A) und (B) in mittlerer Zeit $O(t + \alpha)$ ausgeführt werden, wobei t die Auswertungszeit der Hashfunktion ist.

Def. Sei $s \in \mathbb{N}$, U eine Menge und \mathcal{H} eine endliche Klasse von Funktionen von U nach $\{0, \dots, s-1\}$. Die Klasse \mathcal{H} heißt c -universell, wobei $c > 0$, falls

$$|\{h \in \mathcal{H} \mid h(x) = h(y)\}| \quad \text{für alle } x, y \in U \text{ mit } x \neq y.$$

Satz. Eine Wörterbuchoperation auf einer Hashtabelle mit Belegungsfaktor α kann in erwarteter Zeit $O(t + \alpha)$ ausgeführt werden, wobei t die Auswertungszeit der Hashfunktion ist, wenn die Hashfunktion zufällig aus einer universellen Klasse \mathcal{H} von Hashfunktionen gewählt wird.

Lemma. Sei $s \in \mathbb{N}$ und p eine Primzahl. Für $a \in \{1, \dots, p-1\}$ sei $h_a : \{0, \dots, p-1\} \rightarrow \{0, \dots, s-1\}$, $x \mapsto (ax \bmod p) \bmod s$.

Dann ist $\mathcal{H} = \{h_a \mid 1 \leq a \leq p-1\}$ eine 2-universelle Klasse von Hashfunktionen von $\{0, \dots, p-1\}$ nach $\{0, \dots, s-1\}$.

Lemma. Sei $r \in \mathbb{N}$, s eine Primzahl und $\Sigma = \{0, \dots, s-1\}$. Für jedes r -Tupel $a = (a_1, \dots, a_r) \in \Sigma^r$ sei

$$h_a : \Sigma^r \rightarrow \Sigma, \quad (x_1, \dots, x_r) \mapsto \left(\sum_{i=1}^r a_i x_i \right) \bmod s.$$

Dann ist $\mathcal{H} = \{h_a \mid a \in \Sigma^r\}$ eine 1-universelle Klasse von Funktionen von Σ^r nach Σ .

Satz. Eine Operationsfolge bestehend aus *initialize*(n) und m union- und *find*-Operationen kann in $O(m + n \log n)$ Zeit ausgeführt werden.

Satz. Eine Operationsfolge bestehend aus *initialize*(n) gefolgt von m union- und *find*-Operationen kann in $O(n + m\alpha(n, \frac{m}{n}))$ Zeit ausgeführt werden, wobei α die inverse Ackermann-Funktion bezeichnet.

Lemma. Sei $T = (V, E)$ ein ungerichteter Graph. Dann ist T genau dann ein Baum, wenn beliebige zwei der folgenden Bedingungen erfüllt sind. Dann gilt auch die dritte Bedingung.

- T ist zusammenhängend.
- T ist azyklisch.
- $|E| = |V| - 1$

Satz. Eine topologische Sortierung eines gerichteten azyklischen Graphen kann in $O(n + m)$ Zeit berechnet werden.

Satz. Sei W ein DFS-Wald eines ungerichteten Graphen $G = (V, E)$ und seien $u, v \in V$. Dann gehören u und v genau dann zur selben Zusammenhangskomponente von G , wenn sie Knoten im selben Baum von W sind.

Def. Eine **starke Zusammenhangskomponente** in einem gerichteten Graphen ist eine maximale Gruppe von Knoten, sodass zwischen je zwei Knoten der Gruppe ein Pfad existiert.

Satz. Die starken Zusammenhangskomponenten eines gerichteten Graphen mit n Knoten und m Kanten können in $O(n + m)$ Zeit berechnet werden.

Lemma (Δ -Ungleichung). Für jede Kante $(u, v) \in E$ ist $\delta(v) \leq \delta(u) + c(u, v)$

Algorithmus (Bellman-Ford). Relaxiere n mal je alle Kanten im Graphen.

Satz. Das Single-Source-Shortest-Paths-Problem mit n Knoten und m Kanten kann in Zeit $O(nm)$ gelöst werden.

Satz (Dijkstras Algorithmus). Das Single-Source-Shortest-Paths-Problem kann in $O(n \log n + m)$ Zeit gelöst werden.

Satz. Das Single-Source-Shortest-Paths-Problem kann in Netzwerken mit n Knoten, m Kanten und allen Kantenkosten 1 in Zeit $O(n + m)$ gelöst werden.

Algorithmus (Floyd-Warshall). Verwende Tabelle mit aktuell berechneter Entfernung zwischen je zwei Knoten (dynamische Programmierung). Betrachte dann alle Tripel von Knoten, wende Dreiecksungleichung an.

Satz. Das All-Pairs-Shortest-Paths-Problem mit n Knoten kann in Zeit $O(n^3)$ gelöst werden.

Algorithmus (Kruskal). Für immer diejenige Kante zum Spannbaum hinzu, die die geringsten Kosten hat und durch die kein Zirkel entsteht. Laufzeit: $O(m \log n)$

Algorithmus (Prim). Wir lassen einen minimalen Baum zwischen einer Gruppe durch Knoten durch Hinzunahme der jeweils günstigsten Kante nach „draußen“ wachsen.

Satz. Ein minimal aufspannender Wald eines ungerichteten Netzwerks mit n Knoten und m Kanten kann in Zeit $O(n \log n + m)$ berechnet werden.

Def. Eine (deterministische) **Turing-Maschine** (DTM) ist ein Tupel $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r, \sqcup)$ mit

- Einer Zustandsmenge Q (endlich)
- Einem Eingabealphabet Σ (endlich)
- Einem **Bandalphabet** Γ enthält Σ
- Einer Übergangsfunktion $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$
- Einem Startzustand $q_0 \in Q$
- Einem **akzeptierenden Zustand** $q_a \in Q$
- Einem **verwerfenden Zustand** $q_r \in Q$
- Einem Symbol $\sqcup \in \Gamma \setminus \Sigma$

Satz. Jede Sprache, die von einer RAM mit dem logarithmischen Kostenmaß in Zeit $T(n) \geq n$ entschieden wird, wird von einer Turing-Maschine in Zeit $O(T(n)^4)$ entschieden.

Notation. • Die Menge aller von einer DTM in Polynomialzeit entscheidbaren Sprachen ist P.

- Die Menge aller von einer NTM in Polynomialzeit entscheidbaren Sprachen ist NP.

Frage. $N = NP$?

Problem (Independent Set). Frage: Enthält ein gegebener Graph eine unabhängige Menge der Größe k , also k Knoten, von denen keine zwei benachbart sind.

Problem (Clique). Frage: Enthält ein gegebener Graph eine Clique der Größe k , also einen vollständigen Untergraphen mit k Knoten?

Problem (Vertex Cover). Eine Knotenüberdeckung ist eine Teilmenge aller Knoten in einem Graph, sodass jede Kante im Graph mindestens einen dieser Knoten als Randpunkt besitzt. Frage: Enthält ein gegebener Graph eine Knotenüberdeckung der Größe k ?

Def. Seien L_1 und L_2 Sprachen über Σ_1 bzw. Σ_2 . Eine **Polynomialzeit-Reduktion** von L_1 auf L_2 ist eine Funktion $f : \Sigma_1^* \rightarrow \Sigma_2^*$ mit folgenden Eigenschaften:

- Es gibt eine DTM M und ein Polynom p , sodass M auf jeder Eingabe $w \in \Sigma_1^*$ den Wert $f(w)$ in maximal $p(|w|)$ Schritten berechnet.
- Für alle $w \in \Sigma_1^*$ gilt: $w \in L_1 \iff f(w) \in L_2$

Notation. Wenn es eine Polynomialzeit-Reduktion von L_1 auf L_2 gibt, so schreiben wir $L_1 \leq_p L_2$

Lemma. Die Relation \leq_p ist reflexiv und transitiv.

Lemma. Es gibt Polynomialzeit-Reduktion zwischen den Problemen Independent Set, Clique und Vertex Cover.

Satz. Gilt $L_1 \leq_p L_2$ und ist $L_2 \in P$, dann ist auch $L_1 \in P$.

Def. Eine Sprache L heißt **NP-hart** oder **NP-schwer**, wenn $L' \leq_p L$ für alle $L' \in NP$. Ist L NP-schwer und gilt zugleich $L \in NP$, so heißt L **NP-vollständig**.

Notation. $SAT := \{\langle F \rangle \mid F \text{ ist eine erfüllbare Boolesche Formel in CNF}\}$

Satz (Cook). SAT ist NP-vollständig.

Satz. Independent Set (und somit auch Clique und Vertex Cover) sind NP-vollständig.