Algorithmentheorie

Daniel Neuen (Universität Bremen)

WiSe 2023/24

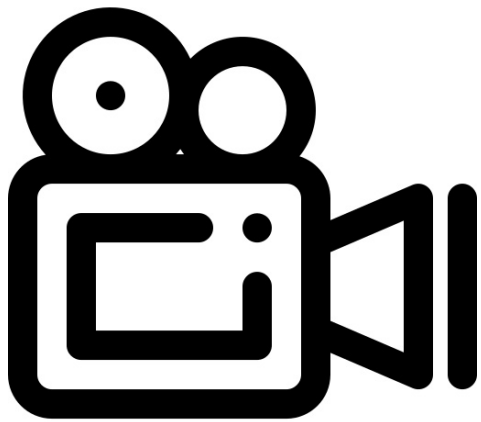
Wiederholung und Klausur

14. Vorlesung

Aufzeichnung der Vorlesung

Diese Vorlesung wird aufgezeichnet und live gestreamt.

- Aufzeichnungen nur der Lehrenden durch sich selbst.
- Bei Rückfragen aus dem Auditorium und Diskussion bitte deutlich anzeigen, falls das Mikro stumm geschaltet werden soll.



Organisatorisches

Evaluation: Die Evaluation läuft bis zum 29.02.2024.

Bitte Teilnehmen!

Rückblick

Motivation

Algorithmus

Eine eindeutige Handlungsvorschrift zur Lösung eines Problems, besteht aus endlich vielen, wohldefinierten Einzelschritten.

Algorithmen sind allgegenwärtig:

- Wegsuche im Navigationsgerät,
- Anfragen an Datenbanken und Suchmaschinen,
- Kompression,
- ...

Was ist Algorithmentheorie?

Algorithmus

Eine eindeutige Handlungsvorschrift zur Lösung eines Problems, besteht aus endlich vielen, wohldefinierten Einzelschritten.

Eine vollständige Analyse eines Algorithmus besteht aus 4 Komponenten:

- **Was wird gelöst:** Spezifikation des Problems.
- **Wie wird es gelöst:** Beschreibung des Algorithmus zur Lösung des Problems.
- **Warum funktioniert das:** Beweis der Korrektheit des Algorithmus.
- **Wie schnell:** Analyse der Laufzeit des Algorithmus.

Ziele der Algorithmentheorie

- Möglichst effiziente Algorithmen (und Datenstrukturen) für die verschiedensten Probleme entwerfen.
- Techniken und Paradigmen zum Algorithmenentwurf bereitstellen.
- Korrektheit der Algorithmen beweisen und Laufzeit analysieren.
- Implementierungen und Evaluationen für Standardalgorithmen bereitstellen.

Ziele der Vorlesung

- Breite Auswahl an wichtigen Problemen und Algorithmen zur Lösung vorstellen.
- Wichtigste Paradigmen zum Algorithmenentwurf vermitteln.
  - Die Vorlesung ist methodenbasiert aufgebaut.
- Soll euch in die Lage versetzen, Probleme die in euren Anwendungen auftreten, zu formalisieren und effiziente Lösungen zu finden.
- Außerdem sollt ihr in der Lage sein die Korrektheit der Algorithmen zu beweisen und ihre Laufzeit zu analysieren.

Themenüberblick

- Problembegriff, Algorithmenbegriff, Schreibweise (Pseudocode), Beispiel Insertion Sort
- Größenordnungen ($\mathcal{O}$-Notation)
- Methoden Rekursion, Divide & Conquer, Backtracking, Beispiele Sortieralgorithmen
- Methode Greedy Algorithmen, Beispiele Interval Scheduling und Fractional Knapsack
- Methode Dynamische Programmierung, Beispiele Knapsack und Gewichtetes Interval Scheduling

Themenüberblick

- Grundlegende Begriffe Graphentheorie
- Minimum Spanning Trees
- Kürzeste Wege: Dijkstra, Bellman-Ford, Floyd-Warshall
- Maximale Netzwerkflüsse
- Kardinalitätsmaximale Matchings (in bipartiten Graphen)
- Ausblick: Approximationsalgorithmen, parametrisierte Komplexität

Klausur

Termine

Klausurtermin: 19. Februar, 14:15-15:45 Uhr, HS 1010 und HS 2010

Wiederholungstermin: 08. März, 10:15-11:45 Uhr, MZH 1380/1400

Die Klausur dauert **90 Minuten**.

Ablauf:

- Studierenden- und Lichtbildausweis bitte auf den Tisch legen
- Klausurbögen erst anschauen, wenn von der Aufsicht angesagt
- Keine Hilfsmittel, Handys in Tasche verstaut

Prüfungsgegenstand

Inhalt der Klausur sind alle

Begriffe, Definitionen, Probleme, Algorithmen, Sätze, Beweise, Beweisskizzen, Folgerungen, Beobachtungen, Anmerkungen, Beispiele, Zusammenfassungen, etc.,

die in Vorlesung, Tutorium oder Übungsaufgaben behandelt wurden.

Hinweis: Falls das Mastertheorem gebraucht wird, geben wir es auf der Klausur an.

Typen von Fragen

A Wissensabfragen: Definitionen, Sätze, Algorithmen

Beispiel 1

Was bedeutet $f \in \mathcal{O}(g)$? Welche Funktionen sind in der Klasse $\Omega(g)$?

Beispiel 2

Definiere ein Matching/ Baum/ MST/ einen Fluss in einem Netzwerk.

Beispiel 3

Wie berechnet man einen kürzesten Weg/ MST/max. Matching/Fluss, etc.

Beispiel 4

Formuliere den Max-Fluss Min-Schnitt Satz.

Übung 6.1 (8 Punkte)

Telefonanrufe von New York nach Los Angeles werden wie folgt geroutet: Zuerst geht der Anruf entweder nach Chicago oder nach Memphis, dann wird er entweder durch Denver oder Dallas weitergeleitet, bevor er letztendlich in Los Angeles ankommt. Die Anzahl an Telefonleitungen kann unten stehender Tabelle entnommen werden. Die Aufgabe ist es, die maximale Anzahl an Telefongesprächen zwischen New York und Los Angeles zu bestimmen unter der Annahme, dass keine weiteren Telefonate Leitungen belegen.

Städte	Anzahl an Telefonleitungen
New York – Chicago	500
New York – Memphis	400
Chicago – Denver	300
Chicago – Dallas	250
Memphis – Denver	200
Memphis – Dallas	150
Denver – Los Angeles	400
Dallas – Los Angeles	350

(a) Modelliert das Problem als maximales Flussproblem und zeichnet das Netzwerk der Telefonleitungen.

(b) Berechnet mit Hilfe von Ford-Fulkerson einen maximalen Fluss in obigem Netzwerk. Gebt dabei die verwendeten augmentierenden Wege sowie den aktuellen Fluss nach jeder Erhöhung mit an.

(c) Bestimmt den minimalen s-t-Schnitt C des Netzwerks. Gebt dabei die Schnittkapazität $cap(C)$ sowie die Kantenmenge an.

C Algorithmendesign (für unbekanntes Problem)

Übung 2.3 (5 Punkte)

Wir betrachten das Problem des Potenzierens.

Gegeben: Zwei natürliche Zahlen a und b

Ausgabe: a^b

Gebt einen Algorithmus für dieses Problem mit Laufzeit $\mathcal{O}(\log b)$ an, wobei Addition und Multiplikation in konstanter Zeit ausgeführt werden können. Begründet, dass Euer Algorithmus korrekt ist und tatsächlich die gewünschte Laufzeit erzielt.

Präsenzübung 3.3

(a) Gegeben seien natürliche Zahlen $x_1, x_2, \ldots, x_n \in [0, M]$ für ein ebenfalls gegebenes $M \in \mathbb{N}$. Wir suchen eine Teilmenge $S \subseteq [0, M]$ mit kleinstmöglicher Kardinalität, so dass für jedes $i \in \{1, \ldots, n\}$ ein $s \in S$ existiert mit $|s - x_i| \leq 5$. Gebt einen Algorithmus an, der dieses Problem in Zeit $\mathcal{O}(n \cdot \log n)$ löst. Zeigt, dass euer Algorithmus korrekt ist und die gewünschte Laufzeit erzielt.

(b) Gegeben sei eine Menge von natürlichen Zahlen $X \subseteq [0, M]$ für ein ebenfalls gegebenes $M \in \mathbb{N}$. Wir suchen eine Teilmenge $S \subseteq X$ mit kleinstmöglicher Kardinalität, so dass die folgenden Eigenschaften gelten:

(i) Es gibt ein $s \in S$ mit $s \leq 20$.

(ii) Es gibt ein $s \in S$ mit $M - s \leq 20$.

(iii) Seien $s_1, \ldots, s_{|S|}$ die Zahlen in S in aufsteigender Sortierung. Dann gilt $s_i - s_{i-1} \leq 20$ für alle $i \in \{2, \ldots, |S|\}$.

D Beweisen oder widerlegen von Sachverhalten

Präsenzübung 4.2

Sei $T = (V, E)$ ein einfacher, ungerichteter Graph mit n Knoten. Zeigt, dass die folgenden Aussagen äquivalent sind.

(i) T ist ein Baum.

(ii) T ist zusammenhängend und kreisfrei.

(iii) T hat $n - 1$ Kanten und ist kreisfrei.

(iv) T hat $n - 1$ Kanten und ist zusammenhängend.

(v) T ist maximal kreisfrei, d.h. T ist kreisfrei und $T \cup \{e\}$ enthält einen Kreis für jede Kante $e \notin E$.

(vi) T enthält einen eindeutig bestimmten Pfad zwischen jeweils zwei seiner Knoten.

Übung 4.2 (7 Punkte)

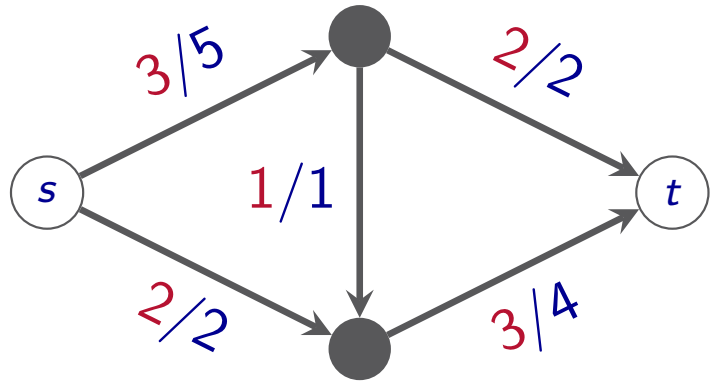
Sei $G = (V, E)$ ein ungerichteter Graph mit Kostenfunktion $c\colon E \to \mathbb{R}$ mit $c(e) \geq 0$ für alle $e \in E$. Seien T_1 und T_2 MSTs von G mit $T_1 \neq T_2$. Zeigt, dass es für jede Kante $e_1 \in T_1 \setminus T_2$ eine Kante $e_2 \in T_2 \setminus T_1$ gibt, so dass $T_3 = (T_1 \setminus \{e_1\}) \cup \{e_2\}$ ein MST von G ist.

Wiederholung Netzwerkflüsse

Wdhl.: s-t-Fluss

Netzwerk

- $G = (V, A, c)$: gewichteter Digraph
- $s \in V$: Quelle
- $t \in V$: Senke
- $c : A \to \mathbb{R}_+$: Kapazität



Ein **zulässiger s-t-Fluss** ist eine Funktion $f : A \to \mathbb{R}_+$, $a \in A$, die folgende Eigenschaften erfüllt:

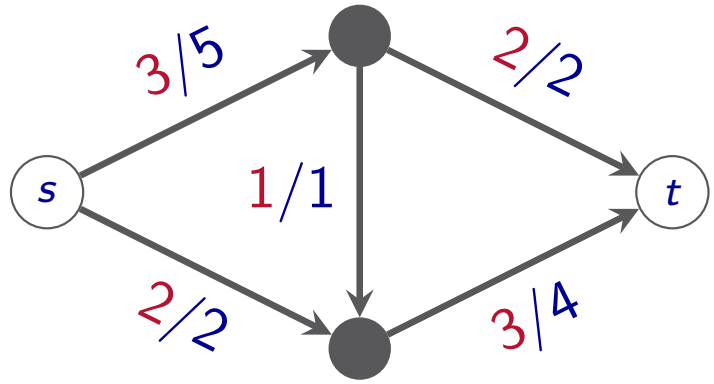
- Kapazitätsbedingung: $0 \leq f(a) \leq c(a), \qquad \forall\, a \in A$
- Flusserhaltungsbedingung:

$$\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a), \qquad \forall v \in V \setminus \{s, t\}$$

Wdhl.: Max s-t-Flussproblem

Der Überschuss (engl. excess) eines Flusses f in $v \in V$ ist

$$ex_f(v) := \sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a).$$



Der Wert eines (ausgehenden) Flusses ist $val(f) = ex_f(s)$.

Wegen der Flusserhaltung gilt $ex_f(s) = -ex_f(t)$, und in jedem anderen Knoten $v \neq s, t$ gilt $ex_f(v) = 0$.

Max s-t-Flussproblem

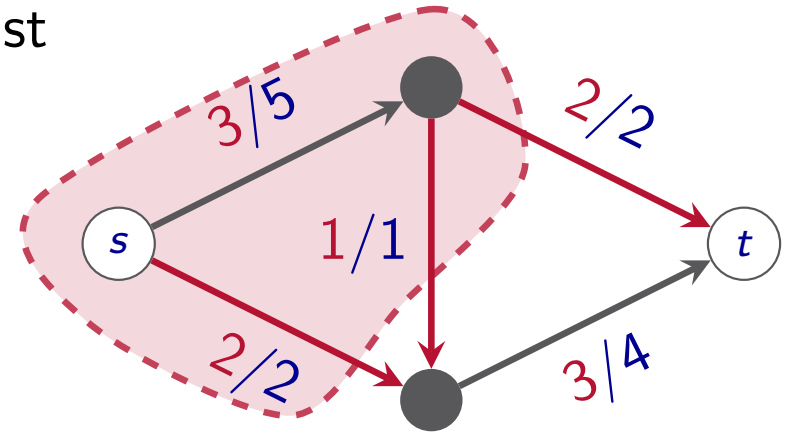
Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$. Finde einen zulässigen s-t-Fluss f mit maximalem Flusswert $val(f)$.

Wdhl.: s-t-Schnitt

Für $U \subseteq V$ sei $\delta^+(U) := \{(u, v) \in A \mid u \in U, v \in V \setminus U\}$.

Sei $U \subseteq V$ mit $s \in U$ und $t \notin U$, dann ist $C := \delta^+(U)$ ein s-t-Schnitt (Cut). Die Schnittkapazität ist

$$\text{cap}(C) := \sum_{a \in C} c(a).$$



Min s-t-Schnittproblem

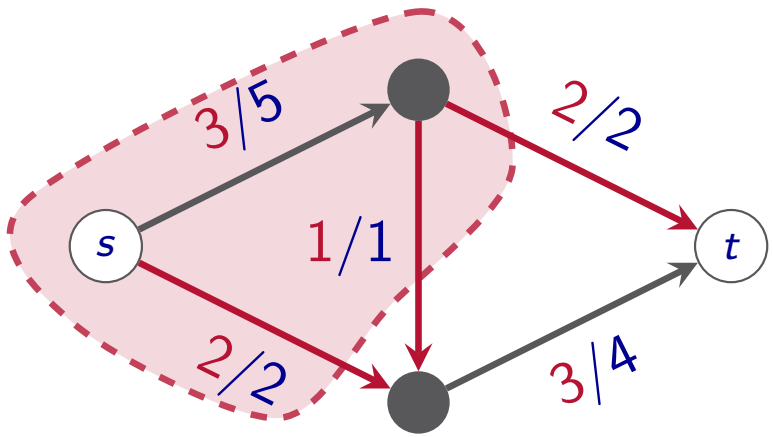
Gegeben sei ein Netzwerk. Finde einen s-t-Schnitt C minimaler Kapazität $\text{cap}(C)$.

Schnittkapazität offensichtlich obere Schranke an den maximalen Fluss.
$\longrightarrow$ formal?!

Max-Flow = Min-Cut

Satz (Max-Fluss Min-Schnitt Theorem)

Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$ mit Kapazitäten $c(a) \geq 0$, $a \in A$. Dann ist der Wert eines maximalen s-t-Flusses gleich der minimalen s-t-Schnittkapazität.



max Fluss = min Schnitt

Optimalitätskriterium

Satz

Sei $N = (V, A, c, s, t)$ ein Netzwerk und f ein zulässiger s-t-Fluss. Dann gilt:

f maximal $\Leftrightarrow \nexists$ f-augmentierender s-t-Weg im Residualgraphen.

Beweis. (Bild an Tafel)

"$\Rightarrow$": klar, da ein f-augmentierender s-t-Weg den Flusswert erhöhen würde.

"$\Leftarrow$": Sei $U \subset V$ die Menge der Knoten, die im Residualgraphen von s aus über gerichtete Wege erreichbar. $\delta^+(U) = C$ ist ein s-t-Schnitt, denn $s \in U$ und $t \notin U$.

$$\Rightarrow val(f) = ex_f(s) \leq \text{cap}(C). \qquad (*)$$

- Für Kante $a = (x, y) \in \delta^+(U)$ in N gilt $f(a) = c(a)$ (sonst $y \in U$).
- Für Kante $a = (u, v) \in \delta^-(U)$ in N gilt $f(a) = 0$ (sonst $u \in U$).

$$val(f) = ex_f(s) = \sum_{a \in \delta^+(U)} f(a) - \sum_{a \in \delta^-(U)} f(a) = \sum_{a \in \delta^+(U)} c(a) = \text{cap}(C)$$

Mit $(*)$ folgt, dass f maximal ist. □

Berechnung von minimalem s-t-Schnitt

1. Berechne maximalen Fluss f (Ford-Fulkerson)
2. Berechne Residualgraph D_f
3. Berechne die Menge U aller Knoten, die in D_f von s erreichbar sind (Breitensuche)
4. Return $\delta^+(U)$

Satz

Der obige Algorithmus berechnet einen s-t-Schnitt mit minimaler Kapazität in Zeit $\mathcal{O}(m \cdot M)$.

Wiederholung $\mathcal{O}$-Notation

$\mathcal{O}$-Notation

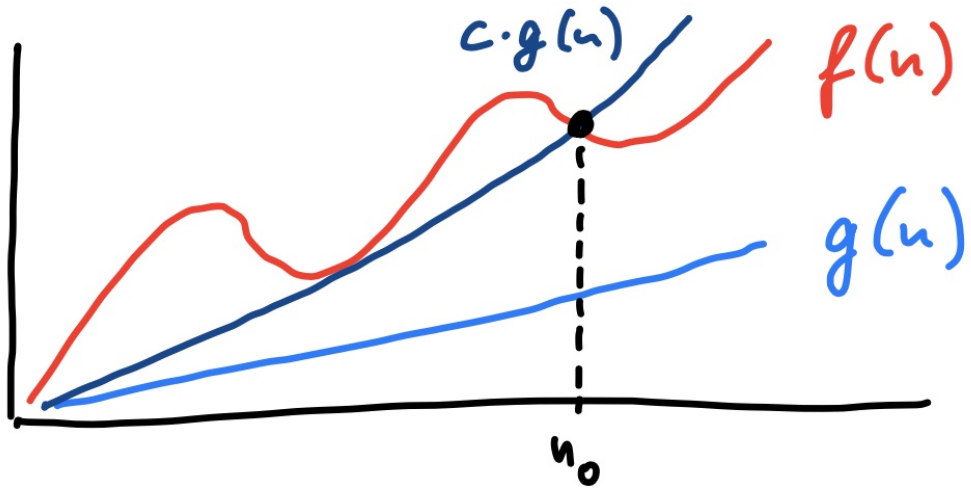
Definition

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion.

- Definiere $\mathcal{O}(g)$ als die Menge aller Funktionen $f : \mathbb{N} \to \mathbb{R}$ mit

$$\exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n).$$

- $\mathcal{O}(g)$ Menge der Funktionen, die nicht schneller wachsen als g



$\mathcal{O}$-Notation

Definition

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion.

- Definiere $\mathcal{O}(g)$ als die Menge aller Funktionen $f : \mathbb{N} \to \mathbb{R}$ mit

$$\exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n).$$

Äquivalente Charakterisierung

Seien $f, g : \mathbb{N} \to \mathbb{R}$ Funktionen. Dann gilt

$$f \in \mathcal{O}(g) \quad \Leftrightarrow \quad \limsup_{n \to \infty} \frac{f(n)}{g(n)} = c$$

für ein $c \in \mathbb{R}_+$.

Groß-$\mathcal{O}$ liefert obere Schranke an die Komplexität einer Funktion.

Ω-Notation

Definition

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion.

- Definiere $\Omega(g)$ als die Menge aller Funktionen $f : \mathbb{N} \to \mathbb{R}$ mit

$$\exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq c \cdot g(n).$$

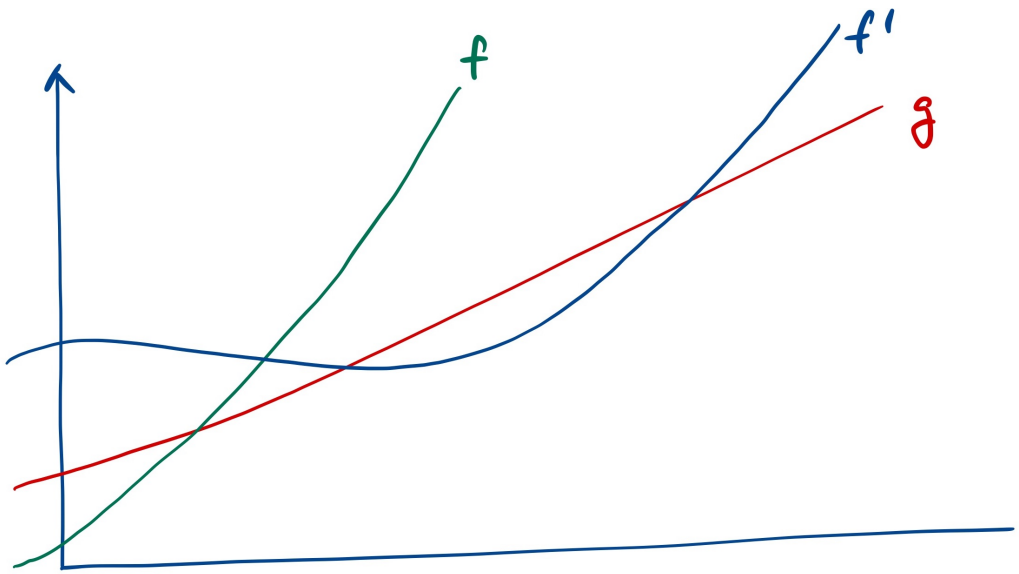
Ω-Notation

Definition

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion.

- Definiere $\Omega(g)$ als die Menge aller Funktionen $f : \mathbb{N} \to \mathbb{R}$ mit

$$\exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq c \cdot g(n).$$



Ω-Notation

Definition

Sei $g : \mathbb{N} \to \mathbb{R}$ eine Funktion.

- Definiere $\Omega(g)$ als die Menge aller Funktionen $f : \mathbb{N} \to \mathbb{R}$ mit

$$\exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq c \cdot g(n).$$

Äquivalente Charakterisierung

Seien $f, g : \mathbb{N} \to \mathbb{R}$ Funktionen. Dann gilt

$$f \in \Omega(g) \quad \Leftrightarrow \quad \liminf_{n \to \infty} \frac{f(n)}{g(n)} > 0.$$

Groß-Ω liefert untere Schranke an die Komplexität einer Funktion.
Große untere Schranken sind nützlich! $\Omega(n^2)$ ist stärker als $\Omega(n)$.

Abschluss

Evaluation: Die Evaluation läuft bis zum 29.02.2024.

Bitte Teilnehmen!

Klausur: 19. Februar, 14:15-15:45 Uhr, HS 1010 und HS 2010

Viel Erfolg!