ADS: Algorithmen und Datenstrukturen 2

Teil I

Peter F. Stadler & Konstantin Klemm

Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for Bioinformatics, **University of Leipzig**

06. April 2011

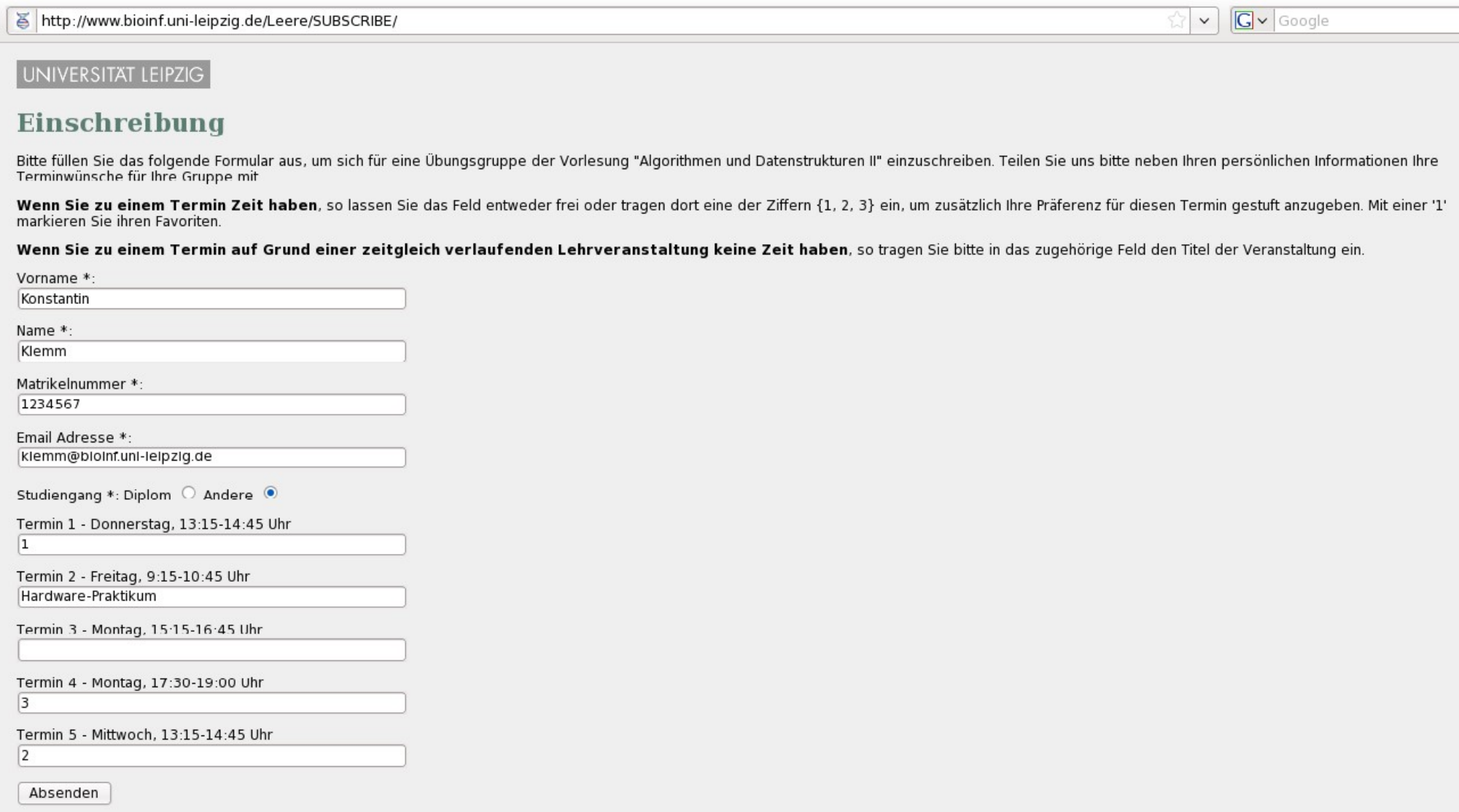
Aktelle Informationen zur Veranstaltung

`http://www.bioinf.uni-leipzig.de/`

$\longrightarrow$ Teaching $\longrightarrow$ Current classes

Übungsaufgaben

- Übungsaufgaben im Netz, Blatt 1 erscheint 20.04.10.
- Abgabe Blatt 1 am 27.04.10, danach alle 14 Tage.
- Abgabe der Lösungen im Hörsaal vor Beginn der Vorlesung.
- Rückgabe der korrigierten Lösungen im Seminar.
- Punkte werden gutgeschrieben für korrekte Lösungen *und* die Fähigkeit, diese zu erläutern.



http://www.bioinf.uni-leipzig.de/Leere/SUBSCRIBE/

Google

UNIVERSITÄT LEIPZIG

Einschreibung

Bitte füllen Sie das folgende Formular aus, um sich für eine Übungsgruppe der Vorlesung "Algorithmen und Datenstrukturen II" einzuschreiben. Teilen Sie uns bitte neben Ihren persönlichen Informationen Ihre Terminwünsche für Ihre Gruppe mit.

Wenn Sie zu einem Termin Zeit haben, so lassen Sie das Feld entweder frei oder tragen dort eine der Ziffern {1, 2, 3} ein, um zusätzlich Ihre Präferenz für diesen Termin gestuft anzugeben. Mit einer '1' markieren Sie ihren Favoriten.

Wenn Sie zu einem Termin auf Grund einer zeitgleich verlaufenden Lehrveranstaltung keine Zeit haben, so tragen Sie bitte in das zugehörige Feld den Titel der Veranstaltung ein.

Vorname *:
Konstantin

Name *:
Klemm

Matrikelnummer *:
1234567

Email Adresse *:
klemm@bioinf.uni-leipzig.de

Studiengang *: Diplom ○ Andere ◉

Termin 1 - Donnerstag, 13:15-14:45 Uhr
1

Termin 2 - Freitag, 9:15-10:45 Uhr
Hardware-Praktikum

Termin 3 - Montag, 15:15-16:45 Uhr

Termin 4 - Montag, 17:30-19:00 Uhr
3

Termin 5 - Mittwoch, 13:15-14:45 Uhr
2

Absenden

Seminargruppen

Anmeldung bis 08.04. (Freitag) online unter

`http://www.bioinf.uni-leipzig.de/Leere/SUBSCRIBE/`

Termine

Donnerstag	13:15 – 14:45	Raum 109, Härtelstrasse 16-18
Freitag	9:15 – 10:45	Raum 109, Härtelstrasse 16-18
Montag	15:15 – 16:45	Raum 1-10, Seminargebäude
Montag	17:30 – 19:00	Raum 109, Härtelstrasse 16-18
Mittwoch	13:15 – 14:45	Raum 109, Härtelstrasse 16-18

Prüfungs(vor)leistung

- Klausur voraussichtlich am 13.07.2010, Dauer 60 Minuten
- Voraussetzung für die Zulassung: erfolgreiche Übungsbearbeitung, d.h. 50 Prozent der erreichbaren Punkte bei den Übungsaufgaben.

Kriterium für Übungsschein (Diplom-Studiengänge):
60 Prozent der erreichbaren Punkte bei den Übungsaufgaben.

Literatur

- Cormen, Leiserson, Rivest, Stein
  Introduction to Algorithms
  The MIT Press.
- Ottmann, Widmayer
  Algorithmen und Datenstrukturen
  Spektrum Akademischer Verlag.

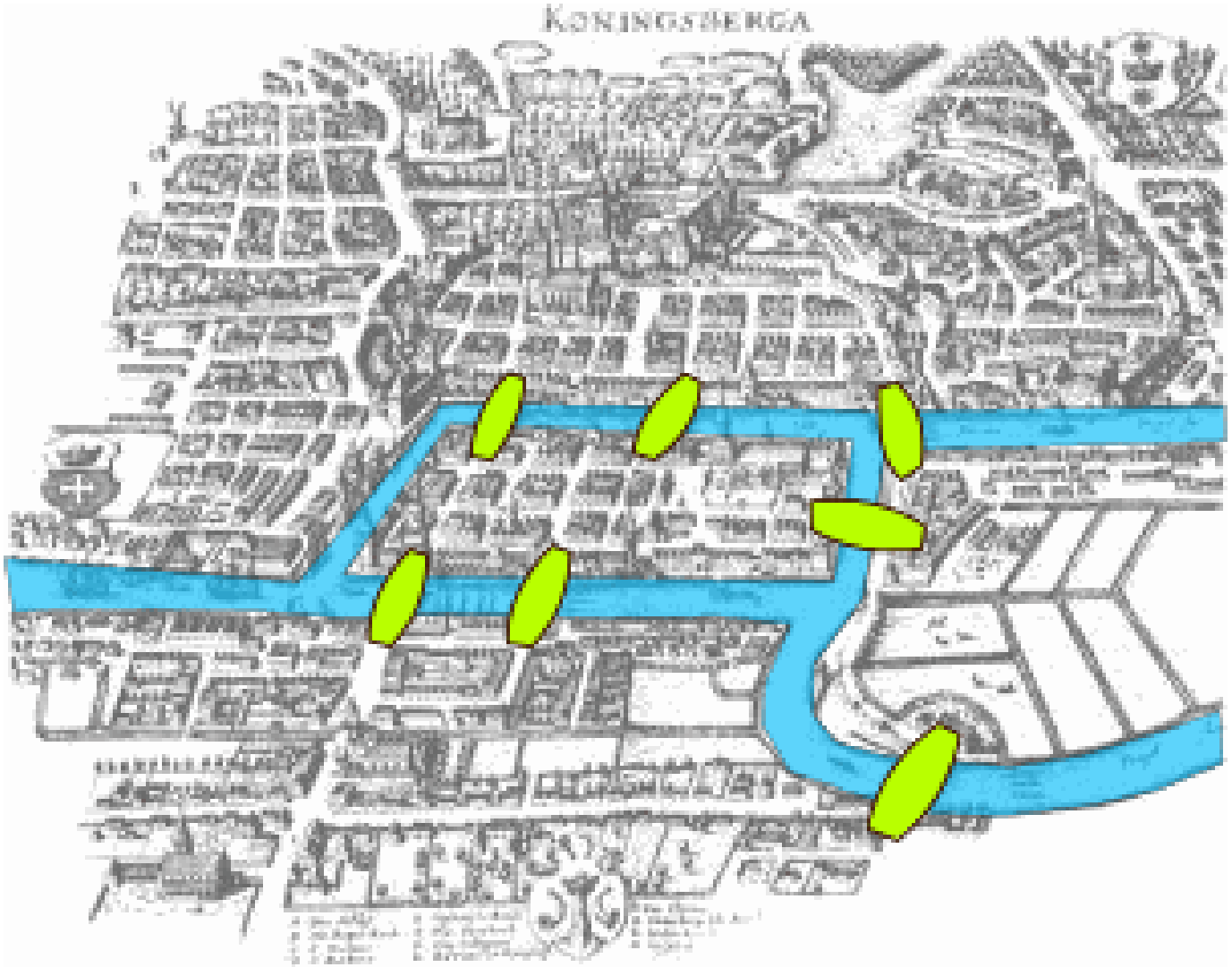
Inhaltsverzeichnis

1. Graphenalgorithmen
2. Verarbeitung von Zeichenketten:
   Suche, Vergleich, Kompression
3. Dynamische Programmierung

Graphen – Themenübersicht

1. Ungerichtete gewichtete Graphen: Grundlegende Definitionen, minimale Spannbäume
2. Gerichtete Graphen: Definitionen, Speicherung, topologische Sortierung, transitive Hülle, starke Zusammenhangskomponenten
3. Gerichtete gewichtete Graphen: Kürzeste Pfade, Flußnetzwerke

Königsberger Brückenproblem (Euler, 1736)

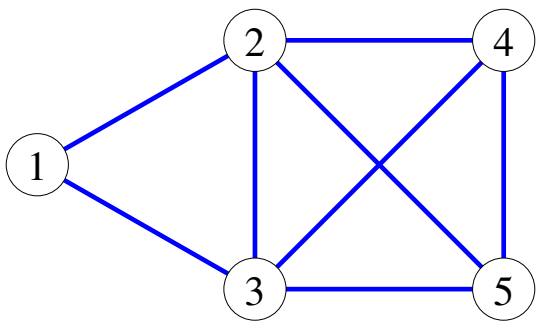


Originalartikel: `http://www.math.dartmouth.edu/~euler/pages/E053.html`

Ungerichteter Graph

Ein Tupel (V, E) heißt *(ungerichteter) Graph*, wenn V eine endliche Menge und E eine Menge ungeordneter Paare von Elementen in V ist.
V heißt *Knotenmenge*, die Elemente von V heißen *Knoten*.
E heißt *Kantenmenge*, die Elemente von E heißen *Kanten*.

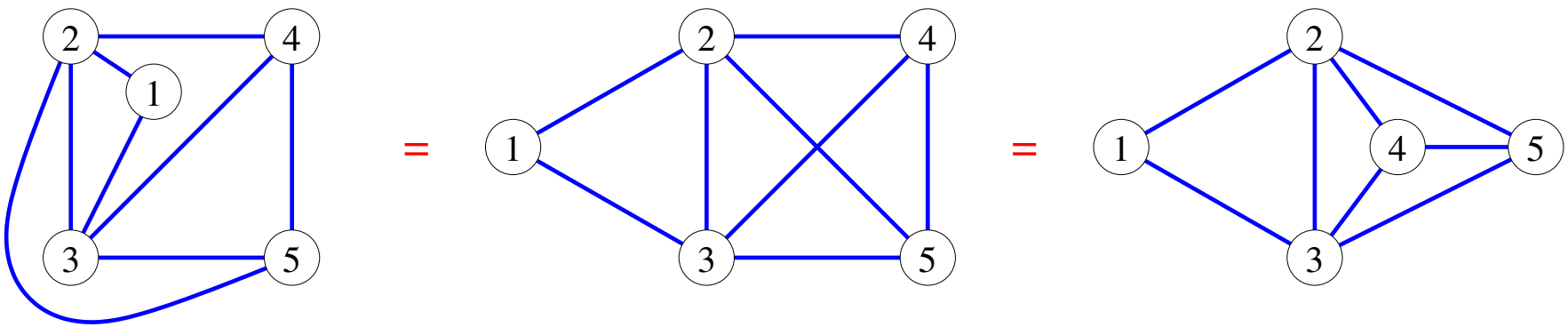
Beispiel: $V = \{1, 2, 3, 4, 5\}$
$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$



Ungerichteter Graph

Ein Tupel (V, E) heißt *(ungerichteter) Graph*, wenn V eine endliche Menge und E eine Menge ungeordneter Paare von Elementen in V ist.
V heißt *Knotenmenge*, die Elemente von V heißen *Knoten*.
E heißt *Kantenmenge*, die Elemente von E heißen *Kanten*.

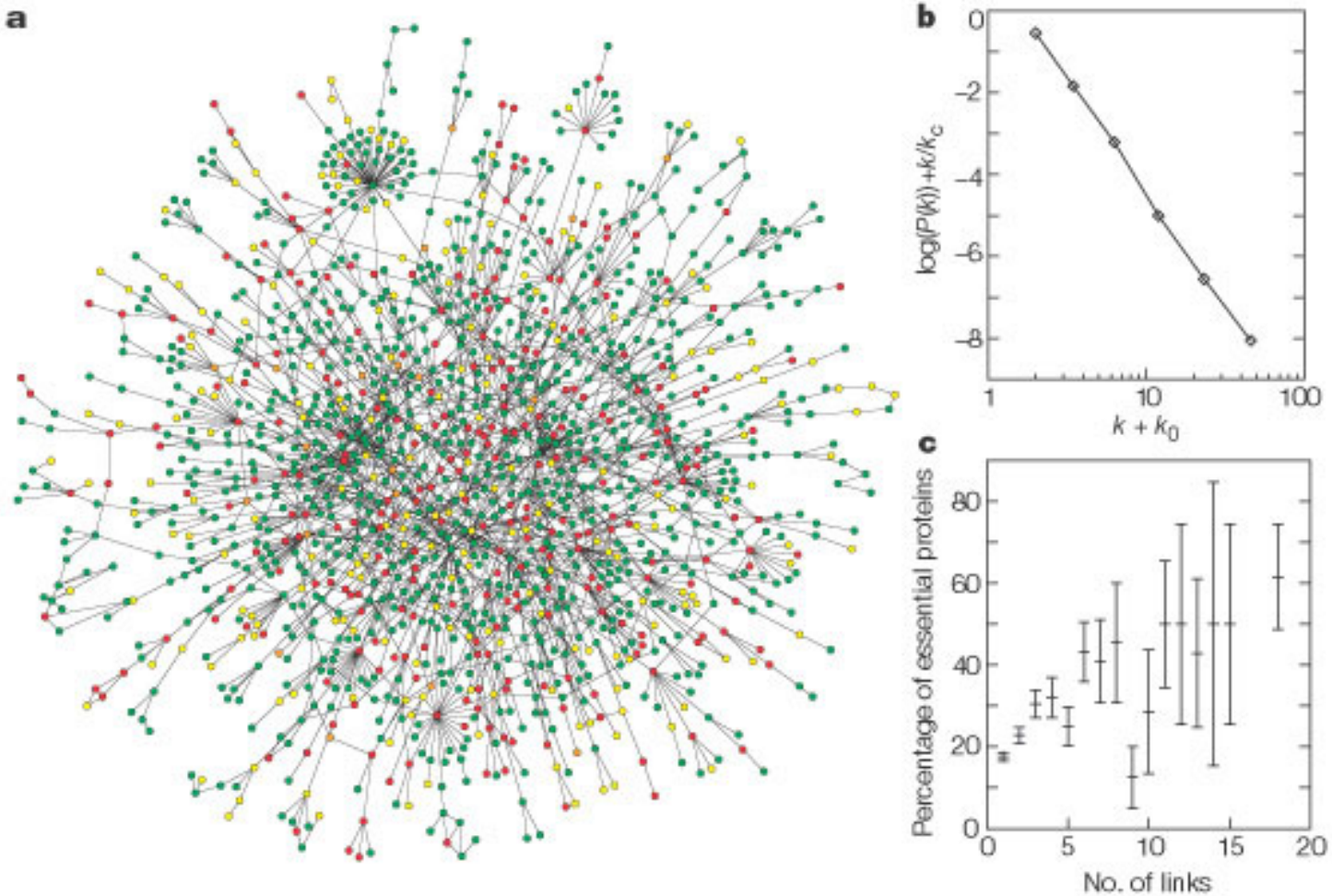
Beispiel: $V = \{1, 2, 3, 4, 5\}$
$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$



Graphen: Beispiele realer Systeme

System	Knoten	Kanten
Internet	Router	Datenleitungen
WWW	Webseiten/-dokumente	Hyperlinks
Gesellschaft	Personen	soziale Kontakte
Biotop	Spezies	trophische Bez., “Fressen”
Molekül	Atome	chem. Bindungen

Graph von Protein-Wechselwirkungen (Hefe)



Jeong, Mason, Barabási & Oltvai, *Nature* 2001.

Teilgraphen

Seien $G = (V, E)$ und $H = (W, F)$ Graphen.

- H heißt *Teilgraph* von G, wenn $W \subseteq V$ und $F \subseteq E$ ist.
- H heißt *spannender Teilgraph* von G, wenn H Teilgraph von G mit $W = V$ ist (H enthält dieselben Knoten wie G).
- H heißt *induzierter Teilgraph* von G, wenn H Teilgraph von G ist und für alle $e \in E$ gilt: $e \subseteq W \Rightarrow e \in F$. (Alle Kanten aus G, deren zwei Knoten in H liegen, sind auch in H enthalten.)

Pfade, Zyklen, Zusammenhang

Sei $G = (V, E)$ ein Graph, $l \in \mathbb{N}$ und $k = (v_0, v_1, v_l) \in V^{l+1}$.

- k heißt *Weg* der Länge l, wenn für alle $i \in \{1, \ldots, l\}$ gilt: $\{v_{i-1}, v_i\} \in E$
- k heißt *Pfad* der Länge l, wenn k ein Weg ist und für alle $i, j \in \{0, \ldots, l\}$ mit $i \neq j$ gilt: $v_i \neq v_j$.
- k heißt *Zyklus* (oder *Kreis*) der Länge l, wenn $(v_1, \ldots, v_l)$ ein Pfad der Länge $l - 1$ ist und $v_0 = v_l$.
- G heißt *zusammenhängend*, wenn für alle $x, y \in V$ ein Pfad zwischen x und y existiert.

Wälder und Bäume

Sei $G = (V, E)$ ein Graph.

- G heißt *Wald* (oder *zyklenfrei*), wenn kein Weg in G ein Zyklus ist.
- G heißt *Baum*, wenn G ein Wald ist und G zusammenhängend ist.

Satz: Ist G ein Baum, so hat G genau $|V| - 1$ Kanten.

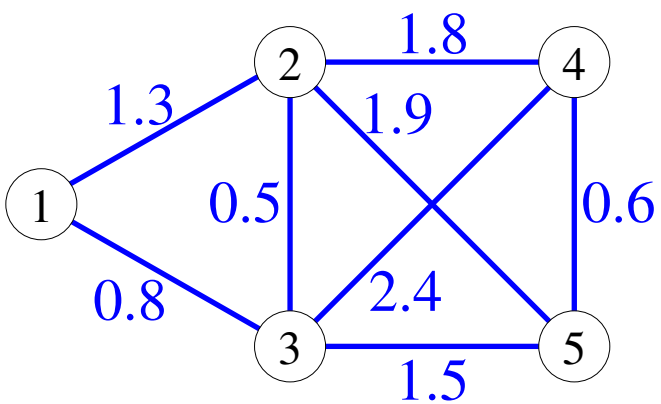
Satz: Ist G zusammenhängend, so hat G einen spannenden Teilgraphen T, so daß T ein Baum ist. T heißt dann *Spannbaum* von G.

Gewichteter Graph

Sei (V, E) ein Graph und $w : E \to \mathbb{R}$.

- Das Tripel $G = (V, E, w)$ heißt *gewichteter* (oder *kantenbewerteter*) Graph.
- $w(e)$ heißt *Gewicht* (oder Länge) der Kante $e \in E$.

Beispiel:



Minimaler Spannbaum

Gegeben: Gewichteter zusammenhängender Graph $G = (V, E, w)$ mit $|V| \geq 2$.

Gesucht: Spannbaum $T = (V, F)$ mit *minimaler Kantensumme*. Wähle also die Kantenmenge $F \subseteq E$ so, daß

$$\sum_{e \in F} w(e)$$

möglichst klein wird.

Kruskal-Algorithmus

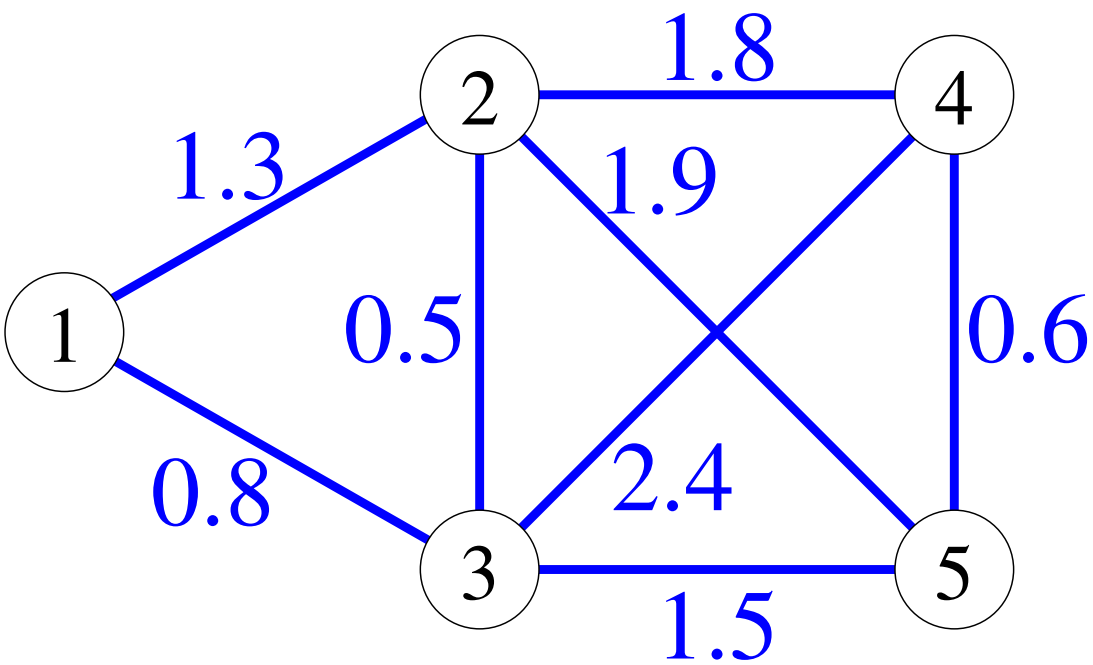
Kruskal-Algorithmus (1956)

1. Erzeuge eine PriorityQueue Q mit der Kantenmenge E sortiert nach Gewicht (kleinstes zuerst). Initialisiere F als leere Menge.
2. Entferne Kante $e = \{u, v\}$ aus Q
3. Falls (V, F) keinen Pfad zwischen u und v enthält:

$$F := F \cup \{e\}$$

   (sonst: tue nichts)
4. Falls Q nicht leer, zurueck zu 2.
5. Ausgabe F.

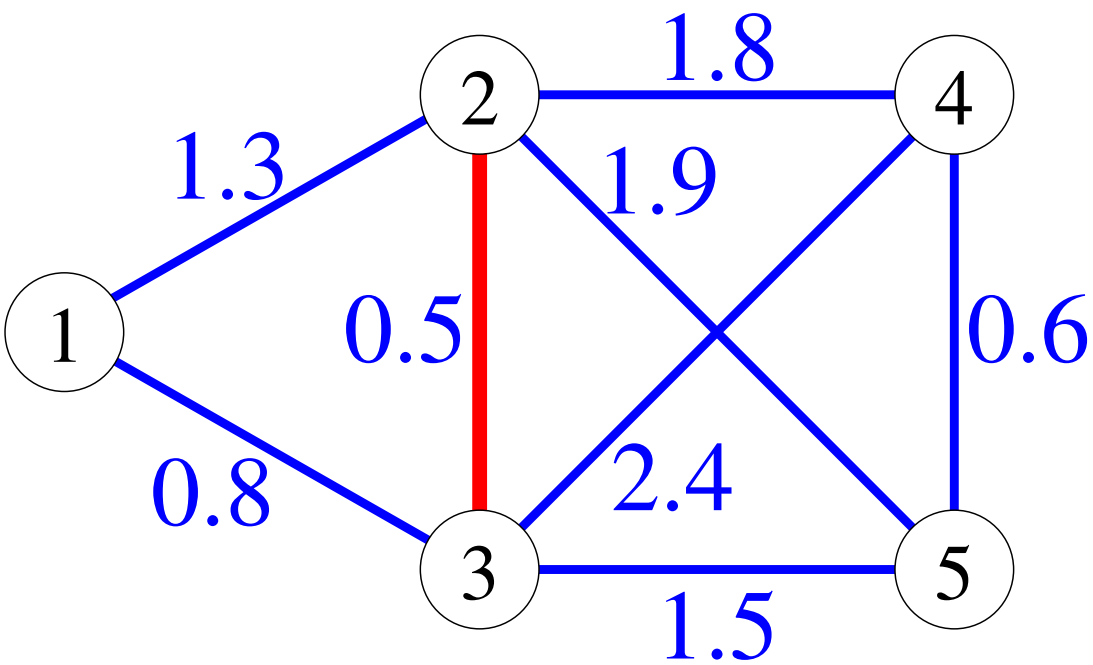
Beispiel-Lauf: Kruskal-Algorithmus



$Q = [\{2,3\},\{4,5\},\{1,3\},\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$T = \{\}$

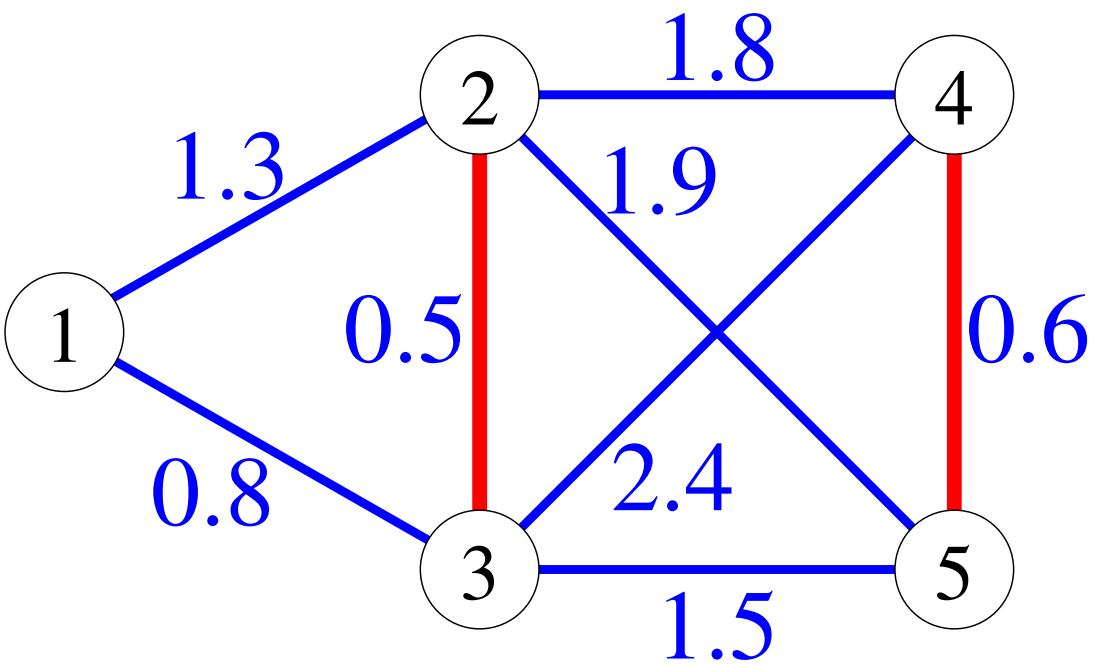
Beispiel-Lauf: Kruskal-Algorithmus



$Q = [\{4,5\},\{1,3\},\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$T = \{\{2,3\}\}$

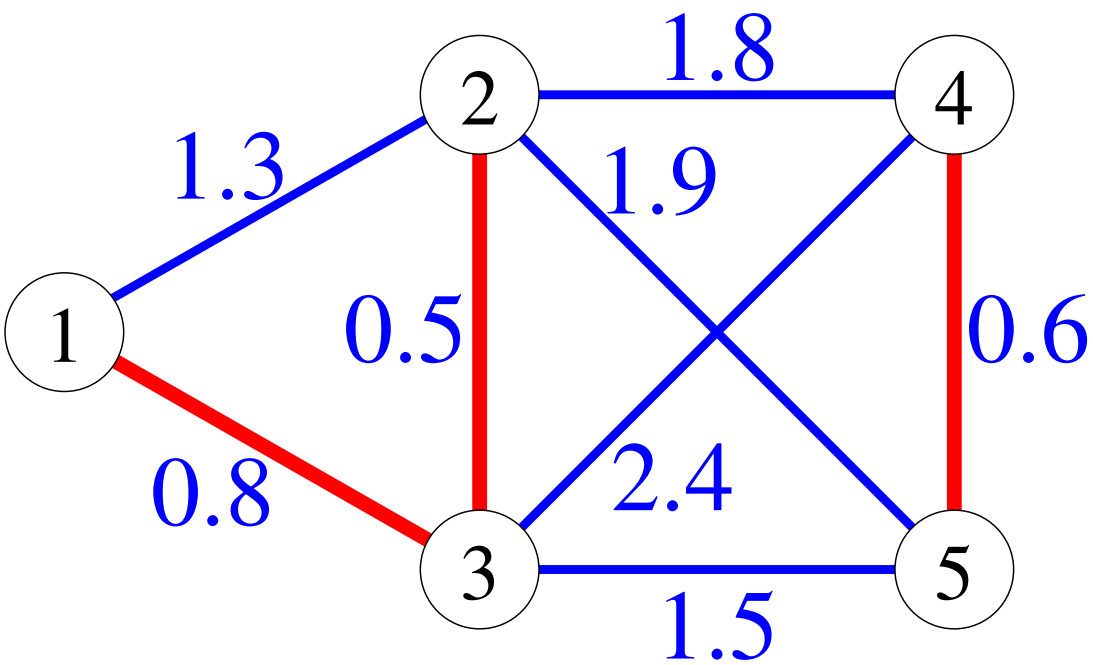
Beispiel-Lauf: Kruskal-Algorithmus



$Q = [\{1,3\},\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$T = \{\{2,3\},\{4,5\}\}$

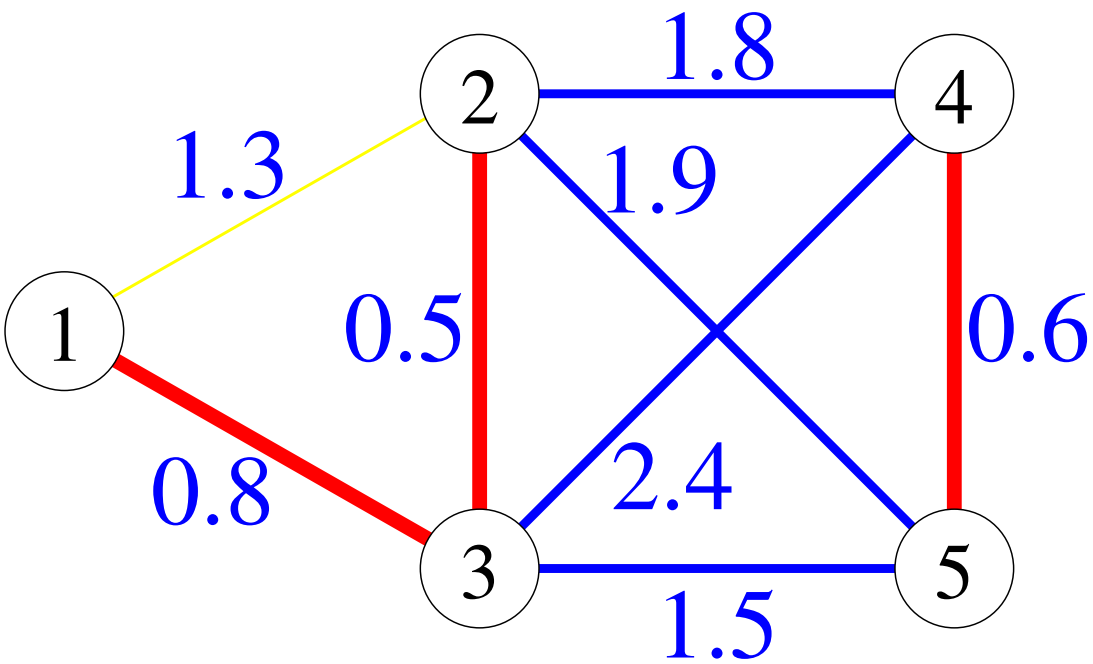
Beispiel-Lauf: Kruskal-Algorithmus



$Q = [\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$T = \{\{2,3\}\},\{4,5\},\{1,3\}\}$

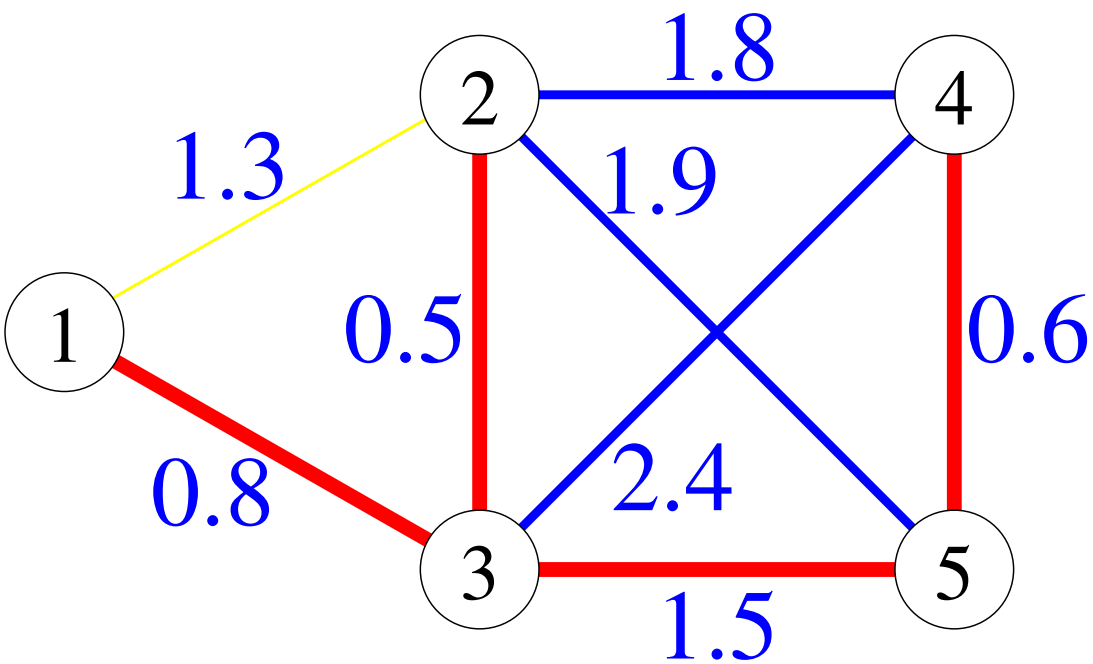
Beispiel-Lauf: Kruskal-Algorithmus



$Q = [\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$T = \{\{2,3\}\},\{4,5\},\{1,3\}\}$

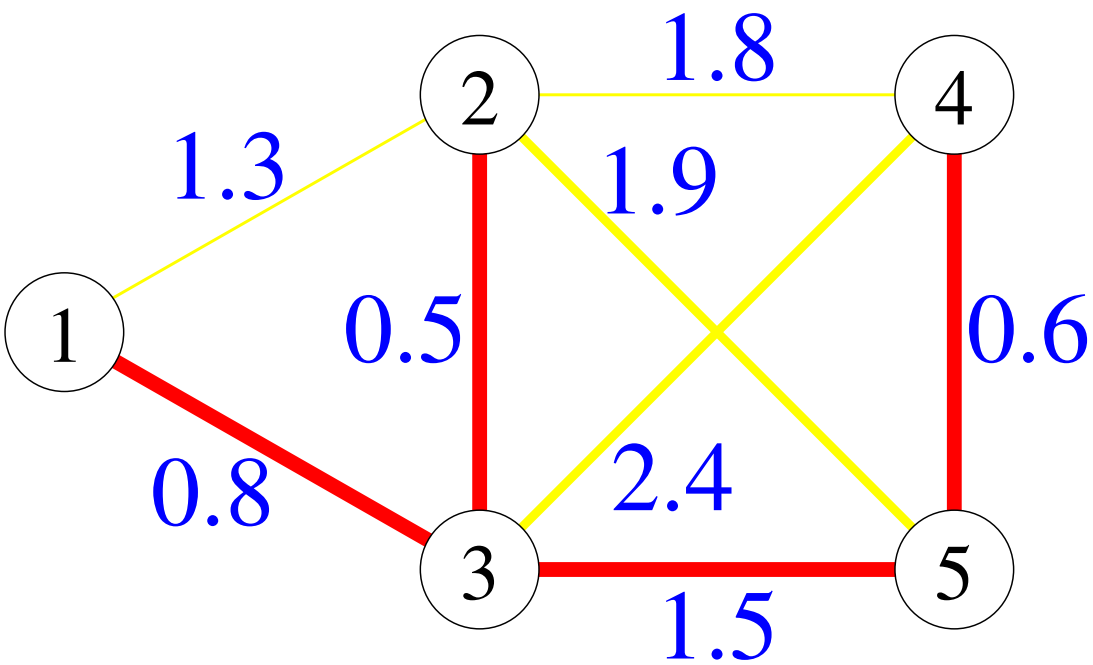
Beispiel-Lauf: Kruskal-Algorithmus



$Q = [\{2,4\},\{2,5\},\{3,4\}]$

$T = \{\{2,3\}\},\{4,5\},\{1,3\},\{3,5\}\}$

Beispiel-Lauf: Kruskal-Algorithmus



$Q = []$

$T = \{\{2,3\},\{4,5\},\{1,3\},\{3,5\}\}$

Anmerkungen zum Kruskal-Algorithmus

- Korrektheit: Findet Kruskal-Algorithmus garantiert einen minmalem Spannbaum? $\rightarrow$ Ja, sehen wir später bei Greedy-Algorithmen.
- Analog: Bestimmung *maximaler* Spannbäume durch den Kruskal-Algorithmus
- Laufzeit-Komplexität $O(|E| \log |V|)$