Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Referat - Präsentation

Aufgabe 3 - Flußprobleme

16.12.2019

Adrian Helberg

Prüfer: Prof. Dr. C. Klauck
Vorlesung: Graphentheoretische Konzepte und Algorithmen

Inhalt

Kontext - Aufgabenstellung

Aufgabenstellung

- Der Algorithmus von **Ford und Fulkerson** [a]
- Der Algorithmus von **Edmonds und Karp** [b]
- Algorithmen *ohne* Residualnetzwerk
- Nachvollziehbare Ergebnisse (Erstellen von Log-Dateien, Generierung von SVG-Dateien, etc.)
- *printGFF*
- Nachweise einer erwarteten Komplexität
- Laufzeitmessungen

---

[a] „Ford-Fulkerson“, 1956, L.R.Ford & D.R.Fulkerson
[b] „Edmonds-Karp“, 1970, Yefim Dinitz, 1972, J.Edmonds & R.Karp

Kontext - Algorithmen

1. *(Initialisierung)*
   Weise allen Kanten $f(e_{ij})$ als einen (initialen) Wert zu, der die Nebenbedingungen erfüllt. Markiere q mit $(undefiniert, \infty)$.

2. *(Inpektion und Markierung)*

   (a) *Falls alle markierten Ecken inspiziert wurden, gehe nach 4.*

   (b) *Wähle eine beliebige markierte, aber noch nicht inspizierte Ecke v_i und inspiziere sie wie folg (Berechnung des Inkrements)*

   - *(Vorwärtskante) Für jede Kante $e_{ij} \in O(v_i)$ mit unmarkierter Ecke v_j und $f(e_{ij}) < c(e_{ij})$ markiere v_j mit $(+v_i, \delta_j)$, wobei δ_j die kleinere der beiden Zahlen $c(e_{ij}) - f(e_{ij})$ und δ_i ist.*
   - *(Rückwärtskante) Für jede Kante $e_{ji} \in I(v_i)$ mit unmarkierter Ecke v_j und $f(e_{ji}) > 0$ markiere v_j mit (v_i, δ_j), wobei δ_j die kleinere der beiden Zahlen $f(e_{ji})$ und δ_i ist.*

   (c) *Falls s markiert ist, gehe zu 3., sonst zu 2.(a).*

3. *(Vergrößerung der Flussstärke)*
   Bei s beginnend lässt sich anhand der Markierungen der gefundene verg. Weg bis zur Ecke q rückwärts durchlaufen. Für jede Verwärtskante wird $f(e_{ij})$ um δ_s erhöht, und für jede Rückwärtskante wird $f(e_{ji})$ um δ_s vermindert. Anschließend werden bei allen Ecken mit Ausnahme von q die Markierungen entfernt. Gehe zu 2.

4. *Es gibt keinen verg. Weg. Der jetzige Wert von d ist optimal. Ein Schnitt $A(X, \bar{X})$ mit $c(X, \bar{X}) = d$ wird gebildet von genau denjenigen Kanten, bei denen entweder die Anfangsecke oder die Endecke inspiziert ist.*

Kontext - Algorithmen

Besonderheiten

- **Ford-Fulkerson** und **Edmonds-Karp** unterscheiden sich nur in der Suchreihenfolge nach einem vergr. Weg.
- **Edmonds-Karp** verwendet hierzu eine *Queue*
- Durch den spezifizierten Rückgabewert
  [(*Liste der im letzten Lauf inspizierten Knoten*)]
  kann Schritt 4. der Algorithmen vernachlässigt werden

Kontext - Komplexität

aus 1. *(Initialisierung)*

- Setzen von Werten: $O(1)$
- Iterieren von Kanten: $O(N)$

aus 2. *(Inspektion und Markierung)*

- Markierung/Inspektion prüfen: $O(1)$
- Iterieren von Knoten: $O(N)$

Kontext - Komplexität

aus 2. (b)

- Wählen einer beliebigen Ecke: *O(N)*
- Wählen der nächsten Ecke aus der Queue: *O(1)*
- Inzidente Kanten abfragen: *O(1)*

Erwartete Komplexität

- *O(Anzahl Kanten x Maximale Anzahl an vergr. Wegen)*
- *O(Anzahl Knoten x Anzahl Kanten x Anzahl Kanten)*

Entwurf

Definition

- Alleinige Vorlage zu einer möglichen Implementierung unabhägig zur Programmiersprache
- Mögliche Probleme schnell erkennen, um Aufwand zu minimieren
- Grundlegendes für effiziente Implementierungen und damit gute Softwarelösungen schaffen

Entwurf - Algorithmen

Wirkungsprinzip

Der Algorithmus beruht auf der Idee, einen Weg von der Quelle zur Senke zu finden, entlang dessen der Fluss weiter vergrößert werden kann, ohne die Kapazitätsbeschränkungen der Kanten zu verletzen. [4] (Wirkungsprinzip)

Informationen an Ecken und Kanten

- Ecken: $(+/-$,Vorgänger,$\delta)$
- Kanten: Fluss, Kapazität

Entwurf - Datenstrukturen

Allgemein

Effiziente Organisation von Daten mit Listen und Tupeln

aus 1. *(Initialisierung)*

- Setzen eines initialen Flusses durch rekursives Durchlaufen aller Kanten

aus 2. *(Inspektion und Markierung)*

- Knoten werden durch die Informationen „Vorzeichen“, „Vorgänger“ und „Delta“ markiert
- Knoten werden inspiziert

Entwurf - Datenstrukturen

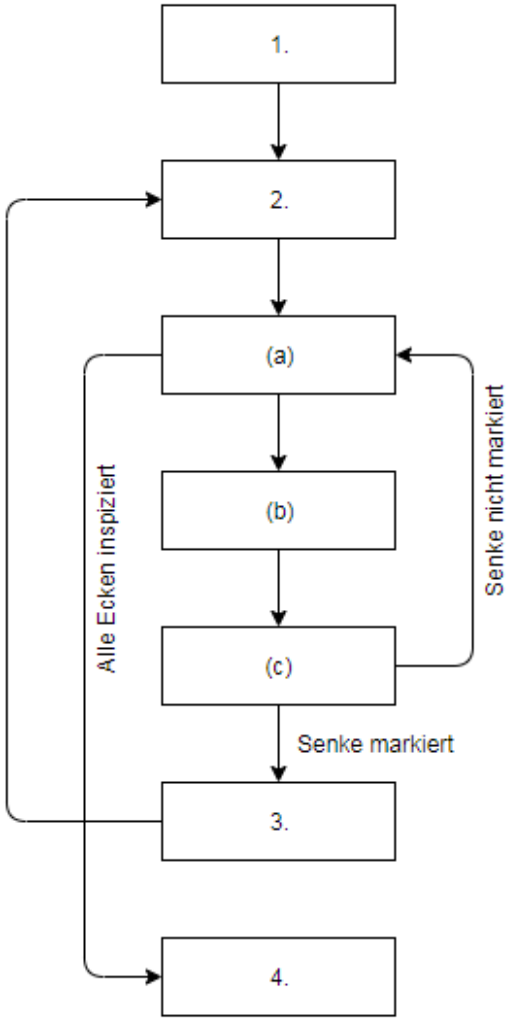
aus 2. (b)

- Ermitteln einer beliebigen Ecke / Holen einer Ecke aus der Queue
- Holen der Inzidenten Kanten der Ecke

aus 2. *(Vergrößerung der Flussstärke)*

- Knoten der Läufe von Vergrößerung sind zu halten (Rückgabewert)

Entwurf - Programmfluss



Laufzeitmessung

Idee

I. Untersuchung bei steigender maximaler Kapazität

II. Untersuchung bei steigender Anzahl an Knoten und Kanten

Zusatz

- Nutzen von **gengraph** zur Erstellung randomisierter Graphen für Laufzeitexperimente

Laufzeitmessung - Versuchsaufbau

Parameter

- Anzahl Ecken
- minimaler Grad
- maximaler Grad
- Mindestgewicht
- Maximalgewicht

Laufzeitmessung - Versuch 1

Maximale Kapazität

- Schrittweise Erhöhung der maximalen Kapazität
- Konstante Restparameter

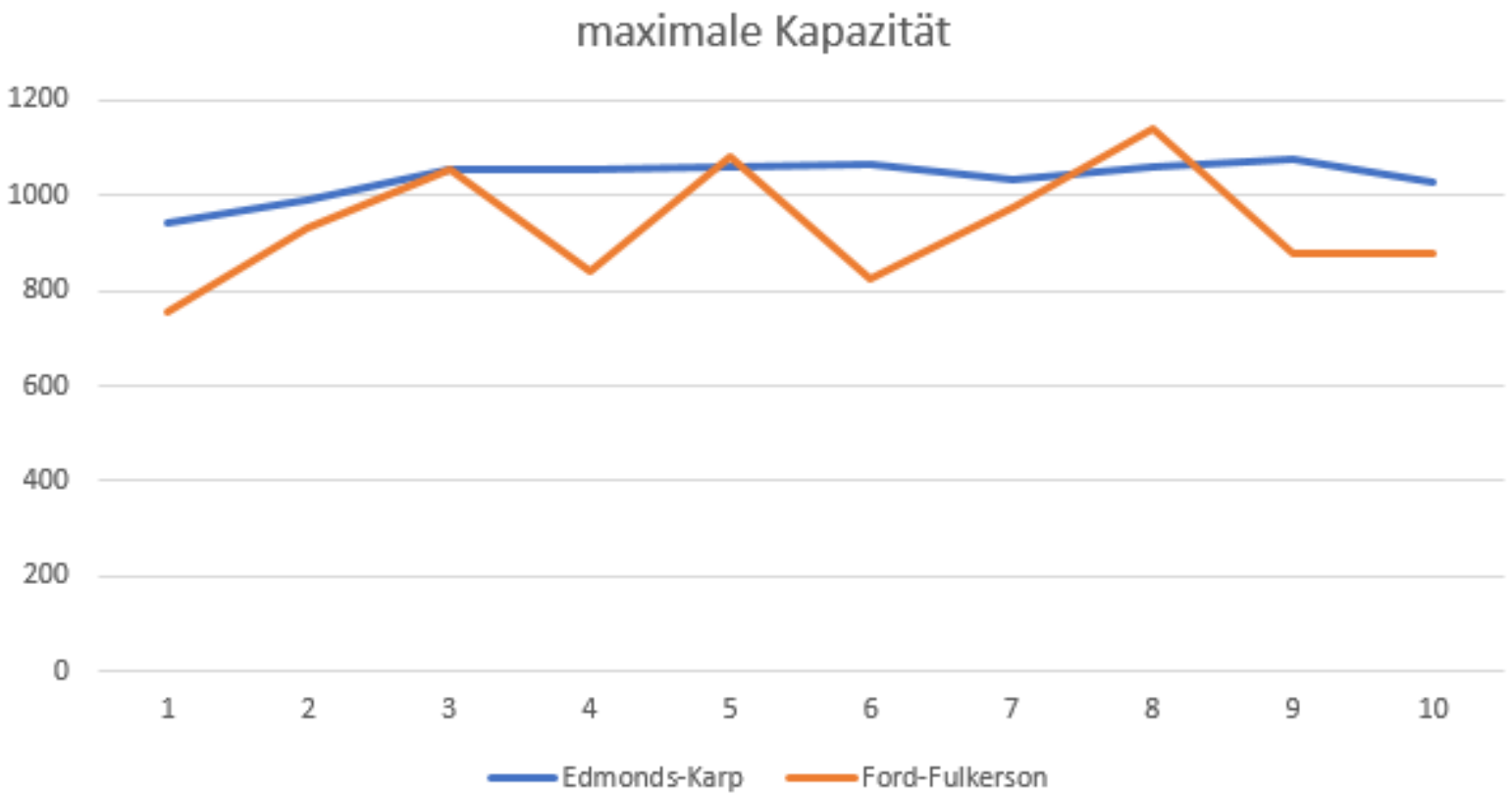
Erwartung

- Ausreißer in der Laufzeit beim **Ford-Fulkerson**
- Konstanter **Edmonds-Karp**

Laufzeitmessung - Versuch 1

Messung	**Edmonds-Karp**	**Ford-Fulkerson**	Maximale Kapazität
1.	942	753	10
2.	989	930	20
3.	1056	1054	30
4.	1054	839	40
5.	1061	1080	50
6.	1064	826	60
7.	1034	974	70
8.	1062	1140	80
9.	1077	878	90
10.	1028	879	100

Laufzeitmessung - Versuch 1



Laufzeitmessung - Versuch 1

Auswertung

- Erwartete Laufzeit

Laufzeitmessung - Versuch 2

Anzahl Ecken und Kanten

- Schrittweise Erhöhung der Anzahl Ecken und Kanten
- Konstante Restparameter
- maximale Kapazität von 1

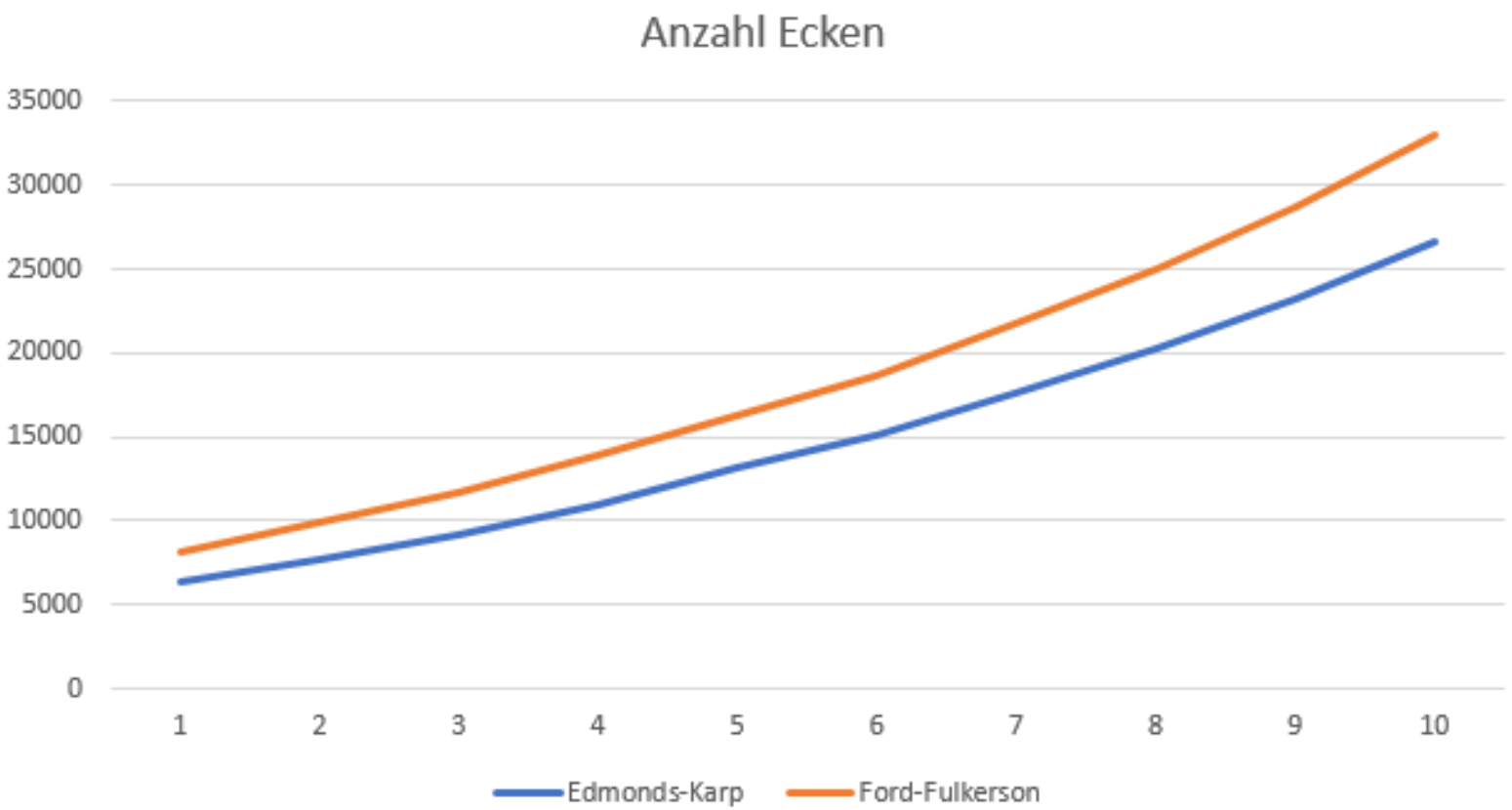
Erwartung

- Laufzeit von *O(E*V)* bei beiden Algorithmen

Laufzeitmessung - Versuch 2

Messung	**Edmonds-Karp**	**Ford-Fulkerson**	Anzahl Ecken (Kanten)
1.	6311	8201	105 (5460)
2.	7744	9850	110 (5995)
3.	9113	11732	115 (6555)
4.	10960	13925	120 (7140)
5.	13108	16321	125 (7750)
6.	15135	18698	130 (8385)
7.	17626	21771	135 (9045)
8.	20338	25076	140 (9730)
9.	23257	28715	145 (10440)
10.	26562	32950	150 (11175)

Laufzeitmessung - Versuch 2



Laufzeitmessung - Versuch 1

Auswertung

- Der *Edmonds-Karp* Algorithmus läuft effizienter
- Nicht die erwartete Laufzeit
- Liegt vermutlich an der Initialisierungzeit der Algorithmen

Ende

Danke für Ihre Aufmerksamkeit!

Quellen

[1] Prof. Dr. C. Klauck. Aufgabe 3: Flußprobleme, 2019.
https://users.informatik.haw-hamburg.de/~klauck/GKA/aufg3.html.

[2] Peter Läuchli. *Algorithmische Graphentheorie.* Birkhäuser, Basel, 9. edition, 1991. ISBN: 978-3-0348-5635-5.

[3] Christoph Klauck & Christoph Maas. *Graphentheorie für Studierende der Informatik.* HAW Hamburg, 6. edition, 2015.

[4] Carsten Milkau. Algorithmus von ford und fulkerson, Juni 2019. Revision 21.06.2019
https://de.wikipedia.org/wiki/Algorithmus_von_Ford_und_Fulkerson.

[5] Marc van Woerkom. Erlang (programmiersprache), November 2019.
https://de.wikipedia.org/wiki/Erlang_(Programmiersprache).