Der Linearzeit MST-Algorithmus
Der Schnellste Algorithmus Für Das MST/ MSF Problem

Maximilian Springenberg
Proseminar Randomisierte Algorithmen

0 Konventionen

In dieser Ausarbeitung wird sich an den üblichen Konventionen zur Notation von Variablen in Graphen orientiert. So definieren wir einen Graphen als $G = (V, E)$, mit der Anzahl von Knoten $n = |V|$ und Kanten $m = |E|$. Zu G zugehörige Kanten und Knotenmengen werden in dieser Ausarbeitung auch durch V_G, E_G und ihre Mächtigkeiten durch n_G, m_G als solche gekennzeichnet. Für jeden Knoten v nehmen wir an, dass eine Adjazenzliste $adj(v)$ von adjazenten Knoten für ihn vorhanden ist. Ferner betrachten wir ungerichtete gewichtete Graphen und bezeichnen die Gewichtungsfunktion als $w : E \to \mathbb{R}$.
Des weiteren werden auch Wege, bzw. Pfade betrachten. Dazu sei $P(\{v, u\})$ eine Funktion, die in einem Baum den Pfad von v nach u als Knotenfolge angibt. Da es für uns handlicher sein wird gleich mit den Kanten zu arbeiten, definieren wir P_e als die Folge von Kanten auf dem Pfad.

1 Motivation

1.1 MST und MSF

Der minimale Spannbaum, oder auch MST, bezeichnet einen azyklischen zusammenhängenden Teilgraph aus G, der sich dadurch kennzeichnet, dass alle Knoten verbunden sind und die Summe von Kantengewichten $\sum_{e \in E_{MST}} w(e)$ minimal ist.
Ist G selbst nicht zusammenhängend, so werden wir einen minimalen Spannwald, oder auch MSF, als nächst beste Lösung betrachten. Ein minimaler Spannwald besteht aus einer Sammlung von minimalen Spannbäumen für die verbundenen Komponenten in G.

1.2 Vorteile eines nichtdeterministischen Anzatzes

Wir werden in dieser Ausarbeitung zum Kapitel 10.3 aus dem Buch *Randomized Algorithms*, von Motwani, R., Raghavan, P. [1] einen randomisierten Ansatz für einen Algorithmus betrachten, der einen minimalen Spannbaum in erwarteter Linearzeit approximiert.
Das MST Problem ist in P enthalten und es sind bereits deterministische Algorithmen wie die von Prim, Kruskal und Borůvka bekannt, die das Problem mit einer Worst-Case Laufzeitschranke von $O(m \cdot log(n))$

lösen. Zudem existiert der Algorithmus von Bernard Chazelle, für den eine Worst-Case Laufzeitschranke von $O(m \cdot log\beta(m,n))$ bekannt ist, wobei mit $\beta(m,n) = min\{i|log^{(i)}n \leq m/n\}$ die inverse Ackermann Funktion verwendet wird. $log^{(i)}n$ ist hierbei die i-te Anwendung von log auf n. Dementsprechend steigt die Funktion β so schwach, dass die aus ihr resultierenden Faktoren für die Worst-Case Laufzeit hinsichtlich der Größe von Graphen in der Praxis als nahezu konstant angesehen werden kann.
Wozu dient also ein nichtdeterministischer Algorithmus, der im Erwartungswert Linearzeit benötigt? Die Antwort auf diese Frage kann sowohl durch die komplexe Implementierung des Algorithmus von Chazelle, als auch durch die Stabilität, bzw. Güte, des vorgestellten Algorithmus hinsichtlich seiner Laufzeit und Approximation, sowie nicht zuletzt durch das Betrachten des Algorithmus als akademisches Beispiel für kreative Laufzeitverbesserungen begründet werden.

2 F-schwere/-leichte Kanten

F-schwere und -leichte Kanten sind ein wesentlicher Bestandteil des MST-Algorithmus. Mittels der Identifizierung von Kanten in einem Graphen G als F-schwer hinsichtlich eines Waldes F in G kann bereits entschieden werden, dass diese Kante nicht im MST von G enthalten ist. Der Umkehrschluss für F-leichte Kanten gilt jedoch nicht, wie wir sehen werden. Betrachten wir also eine Approximation F eines MST bezüglich G, so können wir neben dem Gewicht des Waldes auch die Anzahl von F-leichten Kanten in G als Gütemaß verwenden.

2.1 Definition

Wir betrachten neben der Gewichtungsfunktion w nun die Funktion w_F, mit

$$w_F(\{u,v\}) = \begin{cases} \infty, P_e(\{u,v\}) = \emptyset \\ max\{w(P_e(\{u,v\}))\}, \text{ sonst} \end{cases}$$

, wobei $w(P_e(\{u,v\}))$ bedeutet, dass w auf alle Kanten des Pfades angewandt wurde. Hierbei und im Folgenden bezieht sich P_e auf Kantenfolgen von Pfaden in dem betrachteten Wald, bzw. Baum.
w_F, gibt also das Kantengewicht der Kante mit maximalem Gewicht auf dem Pfad von u nach v in F aus. Sollte dieser Pfad nicht existieren, so nehmen wir an, dass diese Kante unendlich schwer ist.
Ist das Gewicht $w(e)$ einer Kante e echt größer als das maximale Gewicht auf dem Pfad $P_e(e)$ in F, bzw. $w(e) > w_F(e)$, so bezeichnen wir sie als F-schwer. Sonst ist sie F-leicht.

2.2 Informationsgewinn durch F-schwere Kanten

Wir haben bereits erwähnt, dass F-schwere Kanten nicht in einem MSF, bzw. MST enthalten sein können. Dies werden wir im Folgenden beweisen und aufbauend darauf dann einen Verifikationsalgorithmus definieren.

2.2.1 Beweis

Sei F ein MSF in G. Existiert eine F-schwere Kante $e = \{u,v\}$ in G, so gelten folgende Eigenschaften für F:
(i) Es existiert ein Pfad zwischen u und v in F, mit $e \notin P_e(e)$, da $w(e) > w_F(e)$
(ii) $\forall e' \in P_e(e) : w(e') < w(e)$

Wäre e in einem MSF von G enthalten, so würde das Tauschen einer Kante aus F durch e das Gewicht des MSF F nicht vergrößern. Nehmen wir also an, das e in einem MSF von G enthalten sei.
Aus (i) folgt, dass durch Hinzunahme von e ein Zyklus entsteht. Insbesondere bedeutet das, dass wir für e eine Kante auf dem Pfad zwischen u und v austauschen müssten. Aus (ii) folgt jedoch, dass dies F verschlechtern würde. Dies erzeugt einen Widerspruch zur Annahme. Somit kann e nicht in einem MSF und damit auch nicht in einem MST von G enthalten sein.
Der Umkehrschluss, dass alle F-leichten Kanten im MSF vorkommen gilt jedoch nicht. Nehmen wir an, dass jede F-leichte Kante eines Graphen auch in einem MSF enthalten sein muss. Betrachten wir nun einen vollständigen Graphen G_{w_1} mit der Gewichtungsfunktion $w(e) = 1, \forall e \in E_{G_{w_1}}$ und $n_{G_{w_1}} > 2$, so stellen wir fest, dass man aus jedem Pfad P in G, der alle Knoten verbindet einen MST F von G mit $F = (P, P_e)$ bilden kann. Zudem ist jede Kante F-leicht, da alle Kanten gleich gewichtet werden. Wäre jede F-leichte Kante im MST enthalten, so wäre der MST gleich G. Da G vollständig und $n_{G_{w_1}} > 2$ gilt, wäre aber mindestens ein Zyklus im MST enthalten und damit eine Baum-, bzw. Wald-Eigenschaft verletzt.
Dies erzeugt einen Widerspruch zur Annahme. Somit gilt der Umkehrschluss nicht.
Wir können also sämtliche F-schwere Kanten für das Erfassen des MST ignorieren, bzw. sogar aus G eliminieren. Diese Erkenntnis nimmt sich auch der MST-Algorithmus zum Nutzen.

2.2.2 Verifikation durch F-schwere/-leichte Kanten

Es liegt nahe, dass F kein MSF in G ist, wenn eine Kante $\{u, v\}$ existiert, dessen Gewicht echt kleiner als das des maximalen Gewichts auf dem Pfad $P_e(\{u, v\})$ in F ist. Nähme man an, dass F ein MSF wäre, so könnte man E_F um die Kante $\{u, v\}$ erweitern und den dadurch entstandenen Zyklus mittels Eliminieren der Kante mit maximalem Gewicht auf dem Pfad $P_e(\{u, v\})$ in F lösen. Dadurch hätte man die zusammenhängende Komponente als solche gewahrt und eine schwerere durch eine leichtere Kante substituiert. Folglich hätte man einen Wald mit geringerem Gewicht und F wäre damit kein MSF.
Diese Erkenntnis reicht bereits aus, um einen Verifikationsalgorithmus zum MST und MSF Problem zu konstruieren. Im folgenden wird ein einfacher Algorithmus geschildert, der zeigt, wie w_F für einen Graphen konstruiert werden kann. Man könnte über jeden Pfad in F iterieren und für diesen das maximale Kantengewicht unter dem Start und Endknoten dessen mittels Hashing und einer geeigneten Hash-Funktion in einer Hashmap $h_{w_F} : E \to \mathbb{R}$ abspeichern. Die Hashmap soll ferner ∞ ausgeben, wenn für eine Kante kein Wert gesetzt wurde. Dadurch hätte man w_F mittels h_{w_F} konstruiert. Anschließend würde man über E_G iterieren und sicherstellen, dass gilt $\forall e \in E_G : w(e) \leq w_F(e)$.
Im Worst-Case liegen Alle Knoten aus F auf einem Pfad. In diesem Fall würden $\sum_{i=1}^{n_F} i = \frac{n_F^2 + n_F}{2}$ Pfade existieren. Für dichte Graphen G mit $m_G \to n_G^2 = n_F^2$ wäre dies sogar noch im Rahmen einer linearen Laufzeit, im allgemeinen jedoch nicht. Es gibt allerdings deterministische Algorithmen, wie den von V. King [2] die in Linearzeit auf beliebigen Graphen verifizieren. Eine Aufbereitung dieses Algorithmus würde jedoch den Rahmen dieser Ausarbeitung sprengen.
Man kann den Verifikationsalgorithmus auch so anpassen, dass die F-schweren Kanten ausgegeben werden.

3 Reduzieren des Graphen

Wir werden einen rekursiven Algorithmus konstruieren. Die Rekursion terminiert, wenn nach einer Borůvka Phase der Graph unverändert bleibt oder G leer ist. Folglich müssen wir bei jedem rekursiven Aufruf unseren Graphen reduzieren. Wir werden in diesem Teil betrachten, wie wir durch Verwendung von Borůvka-Phasen die Knoten des Graphen reduzieren können und anschließend durch randomisierte

Stichproben zuzüglich eine Methode kennen lernen, um die Anzahl von Kanten zu reduzieren. Letzteres ermöglicht uns erst eine erwartete lineare Laufzeit, da die Rekursionstiefe durch das alleinige Reduzieren von Knoten für unseren Anspruch zu groß ist.

3.1 Borůvka-Phasen

3.1.1 Idee und Ablauf

Borůvka-Phasen beruhen auf der Erkenntnis, dass in einem beliebigen ungerichteten Graphen G für jeden Knoten $v \in V$ die inzidente Kante mit minimaler Gewichtung $e_{v^{min}} := \{v, u\}, u \in adj(v) : \nexists e' = \{v, u'\}, u' \in adj(v) : w(e') < w(e_{v^{min}})$ im MST von G enthalten ist.
Hierfür können wir einen beliebigen Knoten $v \in V_G$ und die zu v inzidente Kante mit minimalem Gewicht e_{min} in einem zusammenhängenden Graphen G betrachten.
1. Fall $adj(v)$ enthält nur eine Kante oder mehrere inzidenten Kanten sind minimal. Dann ist die minimale Kante zwangsweise im MST enthalten oder ein MST mit gleichem Gewicht kann durch jede der minimalen Kanten konstruiert werden.
2. Fall $|adj(v)| > 1$ und nur eine Kante ist minimal. Nehmen wir an, dass ein MST existiert, der e_{min} nicht enthält. Der MST verbindet alle Knoten. Sei e die zu v inzidente Kante im MST. Nehmen wir e_{min} in den MST auf, so erhalten wir einen Zyklus über v. Entferenen wir e, so ist der Zyklus aufgelöst. Ferner wissen wir $w(e_{min}) < w(e)$, da e_{min} und e zu v inzident sind und e_{min} die einzige minimale, inzidente Kante ist. Folglich haben wir nicht nur den Zyklus wieder aufgelöst und die Baumeigenschaften gewahrt, sondern auch das gewicht des MST reduziert. Damit war der MST nicht minimal und auch kein MST. Aus dem Widerspruch folgt, dass die minimale, inzidente Kante eines jeden Knotens im MST von G enthalten sein muss.
Des weiteren ist für uns interessant, dass durch die markierten, minimalen Kanten kein Zyklus entsteht. Damit ein Zyklus entsteht, müsste ein Pfad geschlossen werden, in dem mindestens eine zusätzliche Kante hinzugenommen wird. Dessen Kantengewicht müsste kleiner als das der bereits inzidenten, minimalen, markierten Kante sein.
Ferner bedeutet das, dass wir durch die zur Kontraktion markierten Kanten E_{min} nach einer Borůvka-Phase einen Wald F in G induzieren. Dies wird für uns insbesondere dann interessant, wenn wir rekursiv einen Wald aufbauen möchten.
Eine Borůvka-Phase hat also den folgenden Ablauf:

1. Markiere inzidente Kanten E_{min} mit minimaler Gewichtung
2. Bestimme die verbundenen Komponenten in $G' = (V, E_{min})$
3. Ersetze jede verbundene Komponente durch einen sie repräsentierenden Knoten in G' und erhalte den Graphen G''
4. Entferne alle Selbstschleifen in G''

3.1.2 Reduktion der Knoten

Es werden die durch E_{min} verbundenen Komponenten auf je einen Knoten reduziert. Wir interessieren uns für den Worst-Case, also die maximale Anzahl an Komponenten. Die kleinste verbundene Komponente wäre theoretisch ein einzelner Knoten ohne Kanten. An dieser Stelle würde unser Algorithmus aber terminieren. Die kleinste verbundene Komponente, die wir in einer Borůvka-Phase betrachten werden, besteht also aus zwei Knoten und einer Kante. Da G zusammenhängend ist, existieren damit maximal $n/2$ verbundene Komponenten.
Wir stellen fest, dass eine Borůvka-Phase die Menge an Knoten in G auf maximal die Hälfte reduziert,

nämlich genau dann, wenn die markierten Kanten ein perfektes Matching beim Bestimmen der verbundenen Komponenten ergeben.

3.2 Randomisierte Stichproben

Wir erfassen randomisierte Stichproben, indem wir für einen Graphen G den Graphen $G(p)$ konstruieren, in dem jede Kante je mit Wahrscheinlichkeit $p \in [0, 1]$ enthalten ist. Den Entscheidungsvorgang, ob eine Kante mit Wahrscheinlichkeit p in $G(p)$ aufgenommen wird bezeichnen wir im Folgenden als Zufallsexperiment. Ferner gilt $V_{G(p)} = V_G$.
Uns sollte bewusst sein, dass im Erwartungswert $|E_{G(p)}| = m \cdot p$ Kanten in $G(p)$ enthalten sind. Dies ist für uns im Kontext des Reduzierens von Kanten zielführend.

3.2.1 Güte einer randomisierten Stichprobe

Wir haben festgestellt, dass randomisierte Stichproben uns die Möglichkeit bieten Kanten elegant zu reduzieren. Jedoch haben wir im Teil *2* zu F-leichten/-schweren Kanten gelernt, dass manche Kanten nicht im MST/MSF von G vorkommen und es wünschenswert ist möglichst wenige F-leichte Kanten in G hinsichtlich einer Approximation F des MST/MSF zu haben.
Im Folgenden werden wir zeigen, dass die Anzahl von F_{MSF}-leichten Kanten in G bezüglich des MSF F_{MSF} von unserer Stichprobe $G(p)$ nach oben durch einen Erwartungswert von n/p beschränkt ist. Diese Annahme folgt aus dem Lemma 10.19 aus dem Buch [1].
Um dies zu beweisen, werden wir zeigen, dass die Annahme von Lemma 10.19 des Buches, nämlich, dass die Zahl F_{MSF}-leichter Kanten in G durch eine Zufallsvariable mit negativer Binomialverteilung und den Parametern n, p stochastisch dominiert wird, korrekt ist. Dazu lehnen wir uns an den Beweis zum Lemma an.
Zunächst möchten wir festlegen, dass im Folgenden alle Kanten aus G nach ihrer Gewichtung aufsteigend sortiert betrachtet werden. Dies wird für uns insbesondere dann handlich sein, wenn wir über die zu betrachtende Kante wissen möchten, ob sie F-leicht ist.
Nun, da wir uns auf eine Iteration der Kanten festgelegt haben, möchten wir $G(p)$ wie üblich durch das Hinzunehmen der betrachteten Kante mit der Wahrscheinlichkeit $p = 0,5$ konstruieren. Während wir $G(p)$ konstruieren, können wir auch simultan, bzw. „online“ den MSF F_{MSF} von $G(p)$ konstruieren. Dies ist beispielsweise durch einen Ansatz, wie den von Kruskal möglich, bei dem wir genau dann eine Kante $\{u, v\}$ in F_{MSF} aufnehmen, wenn u, v in verschiedenen verbundenen Komponenten in F_{MSF} liegen. Wobei wir F_{MSF} mit allen Knoten aus G und einer leeren Kantenmenge initialisieren. Für uns ist dabei interessant, dass dies gleichbedeutend damit ist, dass die Kante $\{u, v\}$ aus G genau dann F_{MSF}-leicht ist, wenn u, v in verschieden verbundenen Komponenten in F_{MSF} liegen. Die Korrektheit von F_{MSF} als MSF von $G(p)$ folgt aus der Vorsortierung der Kanten, bzw. aus der Korrektheit des Algorithmus von Kruskal.
Wir können bereits folgende Eigenschaften zur Konsistenz unseres Verfahrens beobachten:
(i) Ob die zu betrachtende Kante F_{MSF}-leicht oder -schwer ist, ist alleinig von den vorhergehenden Zufallsexperimenten abhängig,
(ii) es werden keine Kanten aus F_{MSF} entfernt,
(iii) eine Kante ist genau dann nach dem i-ten Zufallsexperiment F_{MSF}-leicht, wenn sie auch vor dem i-ten Zufallsexperiment F_{MSF}-leicht war.
Definieren wir nun den Begriff von Phasen in unserem Zufallsexperiment, um auf eine Zufallsvariable zu schließen. Die k-te Phase unseres Verfahrens beginne, sobald $|E_{F_{MSF}}| = k - 1$ Kanten vorhanden sind und ende bei $|E_{F_{MSF}}| = k$. Wir befassen uns in einer Phase also unter anderem mit der Anzahl von Zufallsexperimenten, die durchgeführt werden müssen, bis eine Kante hinzugenommen wird. Insbesondere

ist die hinzugenommene Kante per Definition unseres Verfahrens F_{MSF}-leicht. Erinnern wir uns nun an unsere Annahme, so erfassen wir, dass F_{MSF}-schwere Kanten für unsere Zufallsvariable irrelevant sein sollten. Aus diesem Grund ignorieren wir alle F_{MST}-schweren Kanten und befassen uns in einer Phase nun nur noch mit F_{MST}-leichten Kanten und der Anzahl von Zufallsexperimenten, bis eine F_{MST}-leichte Kante zum ersten Mal hinzugenommen wurde.
Nehmen wir nun an, dass wir nach Ablauf unseres Verfahrens F_{MST} mit $|E_{F_{MST}}| = s$ erhalten. Aus (i), (ii), (iii) folgt insbesondere, dass bis zum Ende der s-ten Phase im Erwartungswert pro Phase $1/p$ F_{MSF}-leichte Kanten betrachtet wurden. Dies entspricht der geometrischen Verteilung mit Parameter p. Da es F_{MST}-leichte Kanten geben kann, die in der s-ten Phase noch betrachtet, aber nicht mehr hinzugenommen wurden, müssen wir, um eine Zufallsvariable zu erhalten, die unser Verfahren stochastisch dominiert, noch solange weitere Zufallsexperimente für Pseudokanten durchführen, bis zusätzliche $c = (n-1)-s$ Pseudokanten hinzugenommen wurden. Die Zufallsvariable, die die Anzahl von Zufallsexperimenten unter Berücksichtigung der Pseudokanten beschreibt sei X_{zexp}. Unser Verfahten unter Zunahme der Pseudokanten dominiert also das Verfahren, das während der s-ten Phase endet, Da s maximal gleich $n-1 = s+c$ sein kann. Die maximale erwartete Anzahl von F_{MST}-leichten Kanten in G ist folglich durch den Erwartungswert von X_{zexp} nach oben beschränkt.
X_{zexp} entspricht der negativen Binomialverteilung, da wir eine Anzahl von n Erfolgen je mit Wahrscheinlichkeit p erreichen möchten. Folglich ist die erwartete Anzahl von F_{MST}-leichten Kanten in G nach oben durch einem Erwartungswert von $E[X_{zexp}] = n/p$ beschränkt.

4 Der MST-Algorithmus

MST
Data : Graph G
Result : Approximation eines MST/ MSF in G
1: $G_1, C \leftarrow$ 3 Borůvka-Phasen werden auf G angewandt. Dabei wird der resultierende Graph und ein Teilgraph C mit den zur Kokatenierung markierten Kanten zurück gegeben. **Wenn** G leer ist oder in den Borůvka-Phasen terminiert wird geben wir $F = C$ aus.
2: $G_2 \leftarrow G_1(p = 0,5)$
3: $F_2 \leftarrow MST(G_2)$
4: $G_3 \leftarrow (V_{G_1}, E_{G_1} - E_{F_2-heavy})$
5: $F_3 \leftarrow MST(G_3)$
6: **return** $F = C \cup F_3$

4.1 Aufbau

Nun, da wir Verfahren zum Reduzieren von Knoten und Kanten des Graphen G kennen gelernt haben, können wir anfangen einen Algorithmus zu konstruieren, dessen Ausgabe ein Wald F in G ist.
Die erste Entscheidung, die wir treffen müssen ist, ob wir zuerst Knoten oder Kanten reduzieren.
Da wir in *3.1.1* gelernt haben, dass Borůvka-Phasen nur Kanten markieren, die im MST/MSF enthalten sind, fangen wir mit Borůvka-Phasen an und erhalten den Graphen G_1. Dadurch verringern wir zunächst noch deterministisch die Anzahl F-leichter Kanten in G. Sollten wir in in den Borůvka-Phasen terminieren, geben wir den Teilgraph mit allen markierten Kanten C aus. Die Borůvka-Phasen werden also auch das Abbruchkriterium unserer Rekursion sein.
Anschließend konstruieren wir $G_2 = G_1(p)$ und führen einen rekursiven Aufruf auf diesem durch. Wir wissen also durch *3.2.1*, dass G_2 vor dem ersten rekursiven Aufruf im Erwartungswert $\frac{n/8}{p}$ F_2-leichte Kanten enthält.

Wir haben in *2.2.2* gelernt, dass Verifikationsalgorithmen existieren, die auch F_2-schwere Kanten ausgeben können. Da wir durch *2.2.1* wissen, dass F_2-schwere Kanten nicht im MST/MSF von G enthalten sein können, entfernen wir alle F_2-schwere Kanten $E_{F_2-heavy}$ aus dem Graphen G_1, auf dem lediglich die Borůvka-Phasen angewandt wurden und erhalten G_3.
Führen wir nun noch einmal einen rekursiven Aufruf auf G_3 durch, so erhalten wir einen Wald F_3 mit intuitiv weniger F_3-schweren, als F_2-schweren Kanten in G. Ferner ist insbesondere in dichten Graphen zu erwarten, dass die Stichproben, die in weiteren Rekursionsschritten auf Basis von G_3 gebildet werden nur unwahrscheinlich die zusammenhängenden Komponenten in unserem Wald aufteilen. Dies können wir dadurch begründen, dass dichte Graphen nahe n^2 Kanten enthalten und wir nicht in jedem rekursiven Aufruf gleich n/p Kanten entfernen, da die Graphen mit jedem Aufruf vorher durch die Borůvka-Phasen kleiner werden und damit insegesamt im Erwartungswert $\sum_{i=0}^{k}(n/8^i)/p = n/p \cdot \frac{1-(1/8)^{k+1}}{7/8}$, wobei k die Rekursionstiefe darstellt, Kanten eliminiert werden. Wir verursachen also keinen Schaden durch das Entfernen der F_2-schweren Kanten.
Nachdem wir den Wald F_3 bestimmt, haben geben wir unsere, von den Borůvka-Phasen markierten, Kanten in C vereinigt mit F_3 aus.

4.2 Laufzeit

$T(n, m)$ sei die erwartete Laufzeit des MST-Algorithmus für einen Graphen G. Zeile 1 benutzt 3 Borůvka-Phasen und läuft damit in $O(n+m)$. $G_2 = G_1(p)$ aus Zeile 2 kann ebenfalls in $O(m+n)$ berechnet werden. G_1 hat nur noch $n/2^3 = n/8$ Knoten und für G_2 hat damit $|V_{G_2}| = n/8$ Knoten und im Erwartungswert $|E_{G_2}| = m/2$ Kanten. Folglich benötigt die Berechnung von F_2 die erwartete Laufzeit $T(n/8, m/2)$. Die Berechnung F_2-schwerer Kanten kann unter einem Ansatz, wie den von V. King [2] durchgeführt werden und erfolgt dann in Linearzeit. Die Konstruktion von G_3 mittels Eleminierung der F_2-schweren Kanten benötigt ebenfalls $O(m + n)$. Nach dem Lemma 10.19 [1] ist die Anzahl von F-leichten Kanten in G durch $n/p = 2n$, mit $p = 0,5$ im Erwartungswert gegeben. Da G_1 aber nur $n/8$ Knoten hat, können wir die Anzahl an Kanten in G_3 durch $2 \cdot n/8 = n/4$ im Erwartungswert abschätzen. Damit beläuft sich die erwartete Laufzeit der Berechnung von F_3 auf $T(n/2, n/4)$. Die Vereinigung von C und F in Zeile 6 benötigt $O(n + m)$.
Aus den Laufzeiten der Teilschritte folgt $T(n, m) \leq T(n/8, m/2)+T(n/8, n/4)+c(n+m), c \in \mathbb{N}$, mit c als Konstante für die Anzahl von Aufrufen, die Linearzeit benötigen. Nach der Literatur kann der rekursive Anteil der Ungleichung durch $2c(n + m)$ abgeschätzt werden. Betrachten wir die Ungleichung

$$
\begin{aligned}
T(n,m) &\leq T(n/8, m/2) + T(n/8, n/4) + c(n+m) \\
&\leq (T(n/8^2, m/2^2) + T(n/8^2, \frac{m/2}{4}) + c(n/8 + m/2)) \\
&\quad + (T(n/8^2, \frac{n/4}{2}) + T(n/8^2, n/4^2) + c(n/8 + n/4)) \\
&\quad + c(n+m)
\end{aligned}
$$

, die wir durch einsetzen erhalten, so können wir diese Abschätzung nachvollziehen, da wir je folgenden Rekursionsschritt nicht mehr als $c \cdot (n/8 + n/8 + n/4 + m/2) = c \cdot (n + m) \cdot 1/2$ aufsummieren und n, m selbst um einen stärkeren Faktor, als die Anzahl rekursiver Aufrufe reduziert werden. Folglich gilt $T(n, m) \leq\ T(n/8, m/2) + T(n/8, n/4) + c(n + m) \leq c \cdot (n + m) \cdot \sum_{i=0}^{\infty}(1/2)^i = 2c(n + m)$.

Literatur

[1] Motwani, R., Raghavan, P. : *Randomized Algorithms.* Cambridge : Cambridge University Press 1995, Kapitel 10.3.

[2] King, V.: *A Simpler Minimum Spanning Tree Verification Algorithm*: Algorithmica June 1997