Minimum Spanning Trees
- Grundlagen
- Algorithmus
- Beispiel
- Korrektheit
- Komplexität

- Literatur
  - R. G. Gallager, P. A. Humblet, and P. M. Spira: A Distributed Algorithm for Minimum-Weight Spanning Trees. *ACM Transactions on Programming Languages and Systems* 5:1, 66-77, 1983.

Problemstellung

- Voraussetzungen
  - Ungerichteter, zusammenhängender Kommunikationsgraph $G = (V, E)$ ($|V| = n$, $|E| = m$)
  - Kantengewichte $W : E \to \mathbb{R}$, paarweise verschieden
  - Asynchrone Kommunikation
  - Jeder Knoten kennt Kantengewichte inzidenter Kanten
- Problem
  - Bestimme einen aufspannenden Baum minimalen Gewichts (Minimum Spanning Tree, MST)

Motivation

- Fundamentales Problem in verteilten Systemen
- Aufspannende Bäume bilden Grundlage für weitere Algorithmen wie
  - Broadcast
  - Leader Election
  - Zählen der Knoten
  - Berechnen von Funktionen wie Maximum, Summe, ...
- Energieaufwand in Sensor- und Ad Hoc-Netzen
  - Kantengewichte entsprechen Energieaufwand für Kommunikation
  - Minimiere Energieaufwand

Definition 9
Sei T ein Teilbaum („Fragment") eines Graphen $G = (V, E)$, der durch eine Teilmenge der Kanten $E_T \subset E$ induziert wird. Dann heißt eine Kante e ausgehend aus T wenn genau einer der beiden Endknoten von e zum Teilbaum T gehört.

Lemma 10
Sei ein Teilbaum T eines MST induziert durch eine Kantenmenge E_T und e eine aus T ausgehende Kante minimalen Gewichts. Dann ist der durch $E_T \cup e$ induzierte Baum ebenfalls ein Teilbaum eines MST.

Sequentieller Algorithmus von Prim (Wdh.)

1. Wähle einen beliebigen Startknoten und betrachte diesen als „grünen Baum"
2. Wiederhole den folgenden Färbungsschritt $n-1$ mal
3. Wähle eine ungefärbte aus dem grünen Baum ausgehende Kante minimalen Gewichts und färbe sie grün.

- Laufzeit $\mathcal{O}(m \log_{2+m/n} n)$
- Zu sequentiellen MST-Algorithmen siehe auch Kapitel 4 im Skript zur Vorlesung „Entwurf und Analyse von Algorithmen":

http://i11www.ira.uka.de/algo/teaching/scripts/sources/ad.pdf

Lemma 11
Wenn alle Kantengewichte eines Graphen paarweise verschieden sind, so ist der MST eindeutig bestimmt.

Lemma 12
Bei paarweise verschiedenen Kantengewichten ist jeder MST-Algorithmus, der Schritt für Schritt zu (mehreren) bestehenden Teilen des MST die ausgehende Kante minimalen Gewichts hinzufügt, korrekt.

Idee des Algorithmus

- Anfangszustand
  - Alle Knoten sind im Zustand $\mathrm{Sleeping}$
  - Jeder Knoten bildet ein einzelnes Fragment F des MST
  - Das Level L_F jedes Fragments ist am Anfang 0
- Jedes Fragment F sucht die ausgehende Kante e_F minimalen Gewichts und versucht mit dem benachbarten Fragment F' zu verschmelzen
  - Falls $L = L'$ und $e_F = e_{F'}$, so verschmelzen F und F' zu einem Fragment mit Level $L+1$ und core e_F
  - Falls $L < L'$, so wird F von F' absorbiert
  - Ansonsten wartet F bis F' ein höheres Level erhält

Idee des Algorithmus

- Fragment F hat zwei ausgezeichnete Kanten
  - $\min_F$: Die minimale ausgehende Kante
  - core_F: Kante der letzten Verschmelzung oder gleich $\min_F$
- Knoten v speichern
  - Level L
  - Fragment-ID (Gew. der core-Kante der letzten Verschmelzung)
  - Zu inzidenten Kanten ob sie zum MST gehören oder nicht
  - Kante in Richtung $\min_F$
  - Kante in Richtung core_F
- Knotenzustände
  - $\mathrm{Sleeping}$: Anfangszustand
  - Find: Suche nach minimaler ausgehender Kante
  - Found: minimale ausgehende Kante ist bekannt

Aufwachen: Erster Schritt im Zustand $\mathrm{Sleeping}$

- Knoten im Zustand $\mathrm{Sleeping}$
  - Markiere minimale inzidente Kante als branch
  - Sende $\langle\mathrm{connect}\rangle(0)$ über diese Kante
  - Gehe in Zustand Found
  - Bei Empfang von $\langle\mathrm{connect}\rangle$ oder $\langle\mathrm{test}\rangle$ weiter wie später beschrieben

Verschmelzen zweier Fragmente bei e

- Für F_1 und F_2 ist e minimale ausgehende Kante min und core
- Beide Endknoten haben $\langle\mathrm{connect}\rangle(L)$ zum anderen Endknoten geschickt
- Broadcast von $\langle\mathrm{init}\rangle(L+1, W_e, \mathrm{Find})$ an beide Fragmente über branch Kanten
  - Die Knoten updaten den Level und die Fragment-ID (W_e), senden die $\langle\mathrm{init}\rangle$ Nachricht weiter und suchen dann ihre minimal ausgehende Kante
- Von Blättern aus Antwort von $\langle\mathrm{report}\rangle(W)$: Gewicht der neuen minimalen ausgehenden Kante („Convergecast")
- Neue core-Kante des Fragments wird minimal ausgehende Kante

Knoten sucht minimalen ausgehenden Kante I

- Bei Erhalt einer Nachricht $\langle\mathrm{init}\rangle(L, F, \mathrm{Find})$
- Welche inzidenten Kanten sind ausgehend?
- Klassifiziere Kanten als
  - branch: gehört zum MST
  - $\mathrm{rejected}$: gehört nicht zum MST
  - basic: steht noch nicht fest
- Sende $\langle\mathrm{test}\rangle(F, L)$ an minimale Kante mit Status basic
- Antwort $\langle\mathrm{reject}\rangle()$: Nachbar gehört selbst zu F; markiere Kante als $\mathrm{rejected}$, sende $\langle\mathrm{test}\rangle(F, L)$ an nächste Kante
- Antwort $\langle\mathrm{accept}\rangle()$: Kante gefunden!

Knoten sucht minimalen ausgehenden Kante II

- Antwort auf $\langle\mathrm{test}\rangle(F', L')$
  - Falls $F' = F$, so sende $\langle\mathrm{reject}\rangle()$ zurück
  - Ansonsten
    - Falls $L \geq L'$, so sende $\langle\mathrm{accept}\rangle()$ zurück
    - Falls $L < L'$, so verzögere Antwort bis $L \geq L'$

Antwort auf $\langle \text{init} \rangle$: Convergecast des min. Kantengew.

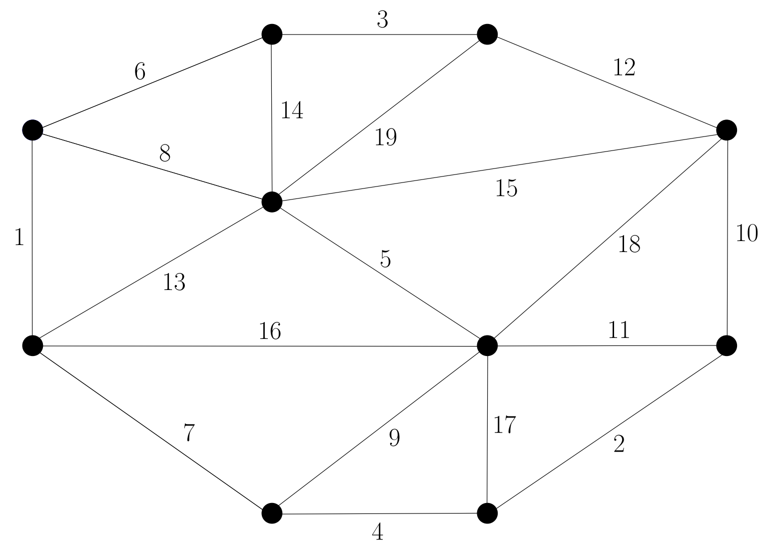
- Blatt eines Fragments
  - W_e sei Gewicht der minimalen ausgehenden Kante e (oder ∞)
  - Sende $\langle \text{report} \rangle (W_e)$ in Richtung core
  - Gehe in Zustand Found
- Innere Knoten
  - Nach Erhalt der $\langle \text{report} \rangle (W_{e_i})$ Nachrichten von allen ausgehenden branch-Kanten e_i
  - Nach Bestimmung der eigenen ausgehenden Kante minimalen Gewichts W_e
  - Bilde das Minimum W über die Gewichte W_{e_i} und W_e
  - Merke Richtung zur minimalen ausgehenden Kante
  - Sende $\langle \text{report} \rangle (W)$ in Richtung core
  - Gehe in Zustand Found

Wechsel der core-Kante und neuer $\langle \text{connect} \rangle$

- Ankunft von $\langle \text{report} \rangle (W)$ am core
  - Falls $W = \infty$ terminiere (Broadcast $\langle \text{terminate} \rangle$)
  - Sende $\langle \text{change-core} \rangle ()$ entlang des Weges zur minimalen ausgehenden Kante
    - Update der Kante in Richtung core
  - Gespeicherte Kanten in Richtung neuer core-Kante bilden jetzt gerichteten Baum
  - Wurzel ist Knoten, der zur core-Kante inzident ist
  - Markiere die Kante minimalen Gewichts als branch
  - Über die neue core-Kante minimalen Gewichts wird $\langle \text{connect} \rangle (L)$ gesendet

Eingliederung Fragmente niedrigeren Levels

- v empfängt Nachricht $\langle \text{connect} \rangle (L')$ mit $L' < L$ von u
  - Eingliederung des benachbarten Fragments
  - Markiere Kante $\{u, v\}$ als branch
  - Sende $\langle \text{init} \rangle (L, W_{\text{core}}, f)$ an u
  - Falls v in Zustand Find, so ist $f = \text{Find}$
    - Die Knoten des benachbarten Fragments nehmen an Suche nach minimaler ausgehender Kante teil
  - Falls v in Zustand Found, so ist $f = \text{Found}$
    - v hat eine ausgehende Kante kleineren Gewichts als die Kante $\{u, v\}$
    - Die Knoten des benachbarten Fragments gehen nach Weiterleitung der $\langle \text{init} \rangle (\ldots)$ Nachricht in den Zustand Found



Korrektheit I

Lemma 13
Der verteilte MST-Algorithmus ist korrekt, d.h. der Algorithmus terminiert und die als branch *klassifizierten Kanten bilden einen MST.*

Beweis.

- Die branch-Kanten bilden einen MST
  - Folgt aus Lemma 12, denn es werden nur minimal ausgehende Kanten aus Fragmenten des MST als branch markiert
- Wenn der Algorithmus terminiert:
  - Für ein Fragment wird keine ausgehende Kante gefunden
  - Dieses Fragment spannt den ganzen Graph auf

Korrektheit II

- Es gibt keine ,,Deadlocks"
  - Betrachte beliebigen Zeitpunkt vor Terminierung
  - Nichtleere Menge von Fragmenten, jeweils mit einer ausgehende Kante minimalen Gewichts
  - Sei F ein Fragment im niedrigsten Level mit kleinster ausgehenden Kante in diesem Level
  - Jede $\langle\mathrm{test}\rangle$ Nachricht von F wird beantwortet oder weckt einen neuen Knoten auf
  - Jede $\langle\mathrm{connect}\rangle$ Nachricht weckt einen neuen Knoten auf oder
    - Geht an Fragment höheren Levels $\Rightarrow$ F wird eingegliedert
    - Geht an Fragment F' gleichen Levels mit gleicher minimaler ausgehenden Kante $\Rightarrow$ F und F' verschmelzen

□

Nachrichtenkomplexität I

Lemma 13
Die Anzahl der Nachrichten im verteilten MST-Algorithmus ist $\mathcal{O}(m + n \log n)$.

Beweis.

- Ein Fragment in Level $L+1$ enthält immer zwei Fragmente in Level L
  - Level-L Fragmente enthalten mindestens 2^L Knoten
  - Der maximale Level ist durch $\log n$ beschränkt
- Eine Kante wird höchstens einmal als $\mathrm{rejected}$ markiert, dabei wird eine $\langle\mathrm{test}\rangle$ und eine $\langle\mathrm{reject}\rangle$ Nachricht verschickt: $2m$ Nachrichten

Nachrichtenkomplexität II

- In jedem Level ausser 0 und dem höchsten Level kann ein Knoten
  - je einmal $\langle\mathrm{accept}\rangle$ und $\langle\mathrm{init}\rangle$ erhalten,
  - je einmal $\langle\mathrm{test}\rangle$ (erfolgreich) und $\langle\mathrm{report}\rangle$ senden, und
  - einmal $\langle\mathrm{change\text{-}core}\rangle$ oder $\langle\mathrm{connect}\rangle$ senden.

  $5n(-1 + \log n)$ Nachrichten
- In Level 0 kann jeder Knoten einmal $\langle\mathrm{init}\rangle$ empfangen und einmale $\langle\mathrm{connect}\rangle$ senden, und im höchsten Level einmal $\langle\mathrm{report}\rangle$ senden: $5n$ Nachrichten
- Insgesamt also höchstens $2m + 5n \log n$ Nachrichten

□

Untere Schranke Nachrichtenkomplexität I

Lemma 13
Die Anzahl der Nachrichten für einen verteilten, uniformen und asynchronen MST-Algorithmus ist $\Omega(m + n \log n)$. D.h. unser MST-Algorithmus ist optimal bzgl. Nachrichtenkomplexität.

Beweis.

- Über jede Kante muss eine Nachricht geschickt werden, denn sonst könnte sich ,,in der Mitte" dieser Kante ein Knoten befinden, der nicht gefunden würde
  - $\Omega(m)$ Nachrichten

Untere Schranke Nachrichtenkomplexität II

- Leader Election in Ringen benötigt $\Omega(n \log n)$ Nachrichten im asynchronen Fall (ohne Beweis; s. Kapitel ,,Verteilte Algorithmen")
  - Algorithmus für MST $\Rightarrow$ Algorithmus für Leader Election mit gleicher Laufzeit
  - $\Omega(n \log n)$ Nachrichten

$\square$

Schlechte Zeitkomplexität

- Es gibt Beispiele, für die $\Omega(n^2)$ Zeiteinheiten notwendig sind
- Problem dabei
  - Nur ein Knoten wacht spontan auf
  - $\Omega(n^2)$ sequentielle Nachrichten
- Verbesserung des Algorithmus
  - Zu Beginn des Algorithmus ,,aufwecken" aller Knoten
  - Schicke $\langle \text{wake-up} \rangle$ über alle Kanten, auf denen noch keine solche Nachricht empfangen wurde
  - $\mathcal{O}(n)$ Zeiteinheiten, $\mathcal{O}(m)$ Nachrichten

Zeitkomplexität I

Lemma 13
Der verbesserte MST-Algorithmus (,,mit Aufwecken") terminiert nach spätestens $\mathcal{O}(n \log n)$ Zeiteinheiten.

Beweis.

- Zeigen per Induktion: Nach $5ln - 3n$ Zeiteinheiten sind alle Knoten in Level l
- Induktionsanfang $l = 1$:
  - Nach n Zeiteinheiten sind alle Prozessoren aufgewacht
  - Jeder Prozessor hat dann eine $\langle \text{connect} \rangle$ Nachricht verschickt
  - Nach $2n$ Zeiteinheiten ist jeder Prozessor in Level 1 (Empfang oder Auslösen von $\langle \text{init} \rangle$)

Zeitkomplexität II

- Induktionsschritt
  - In Level l sendet jeder Knoten max. n $\langle \text{test} \rangle$ Nachrichten, die spätestens nach $2n$ Zeiteinheiten beantwortet sind
  - Weiter $3n$ Zeiteinheiten für
    - $\langle \text{report} \rangle$
    - $\langle \text{change-root} \rangle$ und $\langle \text{connect} \rangle$
    - $\langle \text{init} \rangle$
  - Danach, zum Zeitpunkt $5ln - 3n + 5n = 5(l+1)n - 3n$, sind alle Prozessoren in Level $l + 1$

$\square$

Bessere Zeitkomplexität

- Es gibt Beispiele, für die der verbesserte Algorithmus $\Omega(n \log n)$ Zeiteinheiten benötigt werden
- $\Omega(n)$ ist triviale untere Schranke
- Der beschriebene MST-Algorithmus kann weiter verbessert werden auf optimale Zeitkomplexität von $\mathcal{O}(n)$

Aktuelle Forschung

- Bessere Algorithmen für spezielle Graphklassen
- Durchmesser
$$D(G) = \max_{u \in V} \max_{v \in V} d(u,v)$$
- Radius
$$R(G) = \min_{u \in V} \max_{v \in V} d(u,v)$$
- MST-Radius $\mu(G, W)$
- Zeitkomplexität in Abhängigkeit von D bzw. μ
  - $\mathcal{O}(D(G) + \sqrt{n} \log^* n)$
  - $\mathcal{O}(\mu(G,W) + \sqrt{n})$