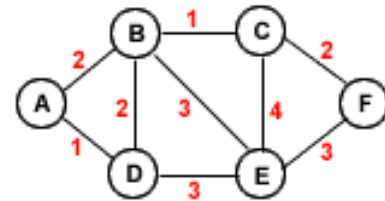




Effiziente Algorithmen

Hartmut Klauck
Universität Frankfurt
SS 06

1.6.





Beweis des Theorems

- **Theorem 12.1:**
 - Sei H^* ein Teilgraph von H , der entsteht, indem jede Kante mit Wahrscheinlichkeit $1-p$ entfernt wird. Sei F ein minimaler Spannwald in H^*
 - Dann ist die erwartete Zahl von T -leichten Kanten in H durch n/p beschränkt
- **Beweis:**
 - Per Annahme sind alle Kantengewichte paarweise verschieden
 - Somit ist der minimale Spannwald eindeutig bestimmt
 - Kanten seien e_1, \dots, e_m mit aufsteigenden Gewichten
 - Für jedes e_i wird eine Münze geworfen, ob e_i in H^* aufgenommen wird [mit Ws. p]
 - Wir „verschieben“ die Münzwürfe, und führen diese während eines Algorithmus zur Spannbaumberechnung aus
 - Verwenden Kruskal als solchen Algorithmus



Beweis des Theorems

- Kruskal beginnt mit leerem Wald F , durchläuft Kanten e_1, \dots, e_m
- e_i wird mit Wahrscheinlichkeit $1-p$ entfernt
- Wenn Kante e_i betrachtet wird, wird e_i durch Kruskal gewählt, wenn e_i zwei Bäume verbindet
- Dann ist e_i zu allen Zeiten $j \geq i$ immer F -leicht
- Verbindet e_i Knoten im gleichen Baum, so ist e_i zu allen Zeiten $j \geq i$ immer F -schwer
- Damit gilt:
 - Ob e_i F -leicht oder F -schwer ist hängt nur von den Kanten e_1, \dots, e_{i-1} ab und den Zufallsentscheidungen für diese
 - Über e_i wird in Schritt i „entschieden“
- Unterteile Berechnung von Kruskal/Zufallswahlen in Phasen:
 - Phase k beginnt, wenn $k-1$ Kanten in F
 - endet, wenn k Kanten in F
 - D.h. während der Phase werden F -leichte Kanten nicht solange wegen der Zufallswahl gewählt bis am Ende eine F -leichte Kante gewählt wird



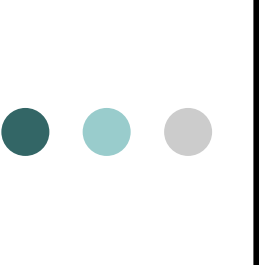
Beweis des Theorems

- Entscheidende Beobachtung:
 - Jede F-leichte Kante wird mit Wahrscheinlichkeit p gewählt
 - Daher ist die erwartete Anzahl verworfener F-leichter Kanten für Phase i genau $1/p$
- Geometrische Verteilung:
 - Eine Münze, bei der Kopf mit Ws. p und Zahl mit Ws. $1-p$ fällt
 - Zufallsvariable X misst, wie lange es braucht, bis Kopf zuerst fällt
 - Erwartungswert $E[X]=1/p$
- Daher ist die erwartete Anzahl betrachteter F-leichter Kanten in den $n-1$ Phasen höchstens $(n-1)/p$
- Da alle Kanten durchlaufen werden ist die Anzahl F-leichter Kanten also erwartet höchstens $(n-1)/p \leq n/p$.



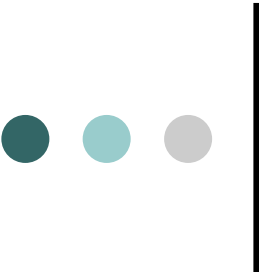
Zusammenfassung

- Wir erhalten einen randomisierten Algorithmus mit erwarteter Laufzeit $O(n+m)$, der minimale Spannäume berechnet.
- Nutzen der Randomisierung:
 - Sampling: Wir erzeugen einen Teilgraphen mit weniger Kanten, hoffen, dass hier ein minimaler Spannbaum eine gute Approximation liefert.
 - Tatsächlich liegen nur wenige Kanten ausserhalb, die noch Kandidaten sein könnten, wir können die meisten Kanten entfernen und nun einen Spannbaum rekursiv berechnen



Verifizierung von Spann­bäumen

- Uns fehlt aber noch ein Linearzeitalgorithmus zur Bestimmung der F-leichten und F-schweren Kanten für einen Graphen G und einen Wald F
- Insbesondere gibt dies einen Algorithmus, der einen gegebenen Spannbaum verifiziert, d.h. entscheidet, ob der Baum tatsächlich minimal ist
- Denn: Wenn alle Kantengewichte paarweise verschieden sind, sind alle Kanten ausserhalb des minimalen Spannbaums T T -schwer
 - T -leichte Kante $\{u,v\}$ wäre leichter als eine Kante auf dem Pfad u nach v in T . Füge $\{u,v\}$ ein und entferne schwerere Kante auf Kreis



Verifizierung von Spannbäumen

- Wenn also ein Spannbaum T gegeben ist, und wir die T -leichte Kanten und die T -schweren Kanten bestimmt haben
- Entscheide „ T ist minimal“ gdw nur die Kanten in T T -leicht sind
 - Wenn T minimal ist, so ist die Bedingung erfüllt
 - Wenn T nicht minimal ist gilt: es gibt Kanten des minimalen Spannbaums, die nicht in T liegen, diese sind T -leicht



Skizze des Algorithmus

- Wir skizzieren nur die Grundidee des Verifikationsalgorithmus
- Gegeben sei ein ungerichteter Graph mit paarweise verschiedenen Kantengewichten sowie ein Spannbaum T
- Wir wollen alle Kanten von G als T -schwer oder T -leicht klassifizieren
- Wir betrachte zuerst nur den Fall, dass T voll verzweigend ist, d.h. jeder interne Knoten hat mindestens zwei Kinder, und alle Blätter liegen auf derselben Ebene

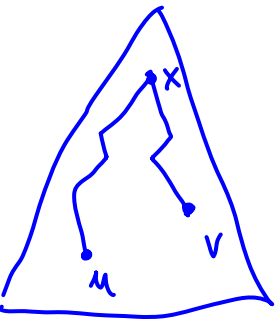


Skizze des Algorithmus

- Wir betrachten zunächst folgendes Modell:
 - Nur Vergleiche zwischen Kantengewichten seien erlaubt (als Operationen auf den Gewichten $W(e)$)
 - Uns interessiert die Anzahl der notwendigen Vergleiche, um T zu testen bzw. alle T -schweren Kanten zu finden.

Bestimmung der schweren Kanten

- Eine Kante $e=\{u,v\}$ ist T-schwer, wenn alle Kanten auf dem Pfad u nach v in T kleineres Gewicht haben als e
- Wir teilen den Pfad u nach v in zwei Teile:
 - Sei x der tiefste gemeinsame Vorgänger (LCA) von u nach v
 - u nach x und x nach v sind die Teilwege
- Wir bestimmen die schwerste Kante des Pfades u nach x für alle u,x (d.h. Teilpfade des Wegs u nach Wurzel)
- Dann:
 - Berechne für alle Kanten $e=\{u,v\}$ den LCA x_e
 - Für alle Kanten e kennen wir die schwerste Kante auf Wegen u nach x_e und v nach x_e . Sind diese kleiner als $W(e)$, so ist e T-schwer





Bestimmung der schweren Kanten

- Gegeben T und m Paare $\{u,v\}$ bzw. $2m$ Paare $\{u,x\}$, so dass x auf dem Weg von u zur Wurzel liegt
- Wir wollen für diese $2m$ Knotenpaare $\{u,x\}$ auf dem Weg u nach x jeweils das grösste Kantengewicht wissen
- **Behauptung:**
 - Wir können diese Gewichte mit $O(n \log ((m+n)/n))$ Vergleichen bestimmen.
- Bemerkung: dies ist immer $O(m+n)$



Bestimmung der schweren Kanten

- Die $2m$ Wege seien in einer Menge A
- Die Wege in A , eingeschränkt auf ihre Teilwege ab Knoten u seien in einer Menge $A(u)$ zusammengefasst
- Wir starten an der Wurzel r , $A(r) = \emptyset$
- Wir durchlaufen T von der Wurzel abwärts
- Sei u ein interner Knoten, mit Vorgänger v
- Die schwersten Kanten für Pfade in $A(v)$ seien bereits bekannt
- Pfade p und q in $A(v)$ haben die Eigenschaft dass p Teilpfad von q oder umgekehrt
- Die schwersten Kantengewichte $s_1, \dots, s_{|A(v)|}$ für Pfade in $A(v)$ sind daher geordnet nach Pfadlänge (Anzahl der Kanten der Pfade): längere Pfade können kein kleineres maximales Kantengewicht besitzen
- Berechnung der schwersten Kantengewichte für $A(u)$:
 - Durch Vergleich von $W(u, v)$ mit s_1, \dots, s_d
 - Anstelle *aller* solcher Vergleiche verwende Binärsuche!
 - Anzahl der Vergleiche so $\log |A(v)|$
- Gesamtkosten: Man kann zeigen, dass $\sum_{v \in T} \log |A(v)| = O(n \log(m/n))$



Implementierung

- Wenn wir nicht nur die Anzahl der notwendigen Vergleiche messen, sondern die Laufzeit, so sind diverse Probleme zu lösen.
- Insbesondere spielen tabellierte Funktionen auf $\log n$ Bits eine Rolle
- Wichtig auch, die Mengen $A(u)$ je nach Grösse unterschiedlich zu behandeln
 - Grosse Mengen werden direkt gespeichert
 - Kleine Mengen durch Pointer in grosse Mengen
- etc.



Allgemeine T

- Wir haben angenommen, dass T an jedem Knoten echt verzweigend ist
- Es gibt eine Reduktion, welche es ermöglicht allgemeine Bäume auf diesen Fall abzubilden



Zusammenfassung

- Es ist möglich mit den obigen Ideen einen Algorithmus anzugeben, der in Linearzeit für einen Baum T und m Knotenpaare (u,v) die schwerste Kante auf dem Weg u nach v in T zu berechnen.
- Damit können in G mit m Kanten zu gegebenem Spannbaum T alle T -schweren Kanten in Zeit $O(m+n)$ berechnet werden
- Wenn alle Kanten ausserhalb von T T -schwer sind, ist T der minimale Spannbaum (paarweise verschiedene Kantengewichte)
- Wir erhalten so einen Linearzeitalgorithmus zur Verifizierung der Minimalität von Spannbäumen
- Sowie einen randomisierten Linearzeitalgorithmus zur Berechnung minimaler Spannäume