

Satz 2.10

Kruskals Algorithmus funktioniert korrekt.

Beweis:

Der Algorithmus liefert einen maximalen kreisfreien Teilgraphen, also nach Satz 2.6 (f) einen aufspannenden Baum.

Außerdem ist Bedingung (b) von Satz 2.9 erfüllt, also ist T optimal. □

2.3.3 Prim's Algorithmus

wie verwendet man 2.9(c) ?!

07.05.13

Algorithmus 2.11 (Prim)

Eingabe : G : Zshgder ungerichteter Graph
 C : Kantengewichte $E(G) \rightarrow \mathbb{R}$

Ausgabe : Aufspannender Baum T minimalen Gewichts.

- ① wähle $v \in V(G)$. Setze $T := (\{v\}, \emptyset)$
- ② WHILE $(V(T) \neq V(G))$ DO
 wähle eine Kante $e \in \delta_G(V(T))$ minimalen Gewichts.
 Setze $T := T + e$.

Satz 2.12

Prims Algorithmus funktioniert korrekt.
Die Laufzeit ist $O(n^2)$.

Beweis:

Die Korrektheit folgt aus der Tatsache,
das Bedingung (c) von Satz 2.10 erfüllt ist.

Für die Laufzeit:

FALSCH: In Schritt (2) jeweils alle Kanten durchsehen
 $\rightarrow O(nm)$

RICHTIG: Für jeden Knoten $v \in V(T)$ die
billigste Kante $e_v \in E(V(T), \{v\})$
zum existierenden Teilbaum merken. (Kandidatenkanten)

Initialisierung: $O(n)$, also sicher $O(n^2)$

(n-1)-mal { Auswahl der billigsten Kante:
Jeweils $O(n)$, überprüfe die $O(n)$ Kandidaten.
Aktualisierung der billigsten Kandidatenkanten:
Gehe alle Kanten des neu eingefügten
Knotens durch, aktualisiere ggf.
die Kandidatenkanten der Nachbarn
 \rightarrow ebenfalls $O(n)$

\hookrightarrow ebenfalls $O(n^2)$



2.3.4 Andere Algorithmen

$O(n^2)$ ist bestmöglich für dichte Graphen, also $m \in \Omega(n^2)$, denn wir müssen auf jeden Fall jede Kante ansehen.

Wenn wir die Kanten sortieren, dann ist $\Theta(m \log m)$ nicht zu verbessern.

Trotzdem lassen sich bessere Laufzeiten erzielen:

Mit "Fibonacci Heaps" (Datenstruktur)

kommt man auf $O(m + n \log n)$.

(Das ist für $m \in O(n)$ noch keine Verbesserung)

Andere Ansätze:

Yao (1975)
Fredman + Tarjan (1976) } $O(m \log \log n)$

Fredman + Tarjan (1987) $O(m \log^* n)$

mit: $\log^*(n) := \min_i \underbrace{\log \log \dots \log}_i n \leq 1$

Chazelle (2000) $O(m \alpha(m, n))$

α : Inverse Ackermann-Funktion

$A(1, n) > n+1$

$A(2, n) > 2n$

$A(3, n) > 2^n$

$A(4, n) > 2^{2^{\dots^2}} \} n$

$A(5, 4) > 10^{10000}$

Immer noch offen:

Deterministischer Algorithmus der Laufzeit $O(m)$.

~~Spezialfälle~~ gelöst

Bekannt:

- Randomisierter Algorithmus mit erwarteter Laufzeit $O(m)$
- Planare Graphen in $O(m)$.
- Punkte in der Ebene in $O(n \log n)$.

Mehr siehe Home Page!

KAPITEL 3: Kürzeste Wege

30

3.1 Problemstellung und Grundstruktur

PROBLEM „KÜRZESTER WEG“

Gegeben: Gerichteter Graph G , Kantengewichte $c: E(G) \rightarrow \mathbb{R}$,
zwei Knoten $s, t \in V(G)$.

Gesucht: Ein s - t -Pfad minimalen Gesamtgewichts.

Viele praktische Anwendungen, auch bedeutsam
als Teilproblem im Kontext schwierigerer Probleme.

Im allgemeinen gar nicht einfach:

Lässt man beliebige Kantengewichte zu, dann
hat man es mit einem NP-vollständigen Problem
zu tun!

→ Taxifahren
in Göttingen

→ Taxifahren
in NY

(Setzt man in einem Graphen alle Kantengewichte
auf -1 , dann entsprechen die Pfade von
Gewicht $1-n$ gerade den Hamiltonpfaden!)

Definition 3.1

Eine Kantengewichtsfunktion $c: E(G) \rightarrow \mathbb{R}$ ist konservativ,
wenn sie keine Kreise negativen Gesamtgewichts enthält.

