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. $\square$

2.3.3 Prims Algorithmus

4.5.11

Wie verwendet man 2.9 (c) ?!

Algorithmus 2.11 (Prim)

Eingabe: G: Zshgdr ungerichteter Graph
c: Kantengewichte $E(G) \to \mathbb{R}$

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
$\rightarrow O(n\,m)$

RICHTIG: Für jeden Knoten $v \in V(T)$ die billigste Kante $e_v \in \underline{E}(V(T), \{v\})$ 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:

$$\left.\begin{array}{l}\text{Yao (1975)}\\ \text{Fredman + Tarjan (1976)}\end{array}\right\} O(m \log \log n)$$

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

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

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

α : Inverses Ackermann-Funktion

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

$A(2,n) > 2n$

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

$A(4,n) > \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\} n$

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

$A(0,m) = m+1$
$A(n+1,0) = A(n,1)$
$A(n+1,m+1) = A(n, A(n+1,m))$

Immer noch offen:

Deterministischer Algorithmus der Laufzeit $O(m)$.

~~Spezialfälle gelöst~~

Bekannt:

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

Mehr siehe Home Page!

KAPITEL 3: Kürzeste Wege

3.1 Problemstellung und Grundstruktur

PROBLEM „KÜRZESTER WEG"

Gegeben: Gerichteter Graph G, Kantengewichte $c: E(G) \to \mathbb{R}$, 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.

→ Taxifahren in Göttingen

→ Taxifahren in NY

Im allgemeinen gar nicht einfach:
Lässt man beliebige Kantengewichte zu, dann hat man es mit einem NP-vollständigen Problem zu tun!

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