15.05.12

Wenn wir eine Kante $e_i = \{v, w\}$ in (3) überprüfen, finden wir die Wurzeln r_v und r_w zu den Komponenten für v und w. Zeitaufwand hierfür: $O(\log n)$

(Höhe des Baumes!)

↳ wird noch verifiziert!)

Teste $r_v = r_w$?

Falls ja, überprüfe nächst Kante.

Falls nein, füge e_i zu T hinzu;

außerdem wird B um eine Kante ergänzt:

Sei $h(r)$ die maximale Länge eines Pfades von r in B. Falls $h(r_v) \geq h(r_w)$, dann füge Kante (r_v, r_w) in B ein, ansonsten (r_w, r_v).

Wie ändert sich $h(r_v)$?

Falls $h(r_v) = h(r_w)$, dann erhöht sich der Wert um 1, ansonsten bleibt er gleich.

Behauptung:

Ein gerichteter Teilbaum von B mit Wurzel r enthält mindestens $2^{h(r)}$ Knoten.

Beweis durch Induktion:

Anfangs gilt $B := (V(G), \emptyset)$, $h(v) := 0$, also gilt die Behauptung.

Außerdem gilt sie nach einer Kanteneinfügung, wenn $h(r)$ sich nicht ändert.

Ansonsten gilt $h(v) = h(w)$ vor der Einfügung, also hat jeder Teil mindestens $2^{h(v)}$ Knoten. Nach der Einfügung haben wir mindestens $2 \cdot 2^{h(v)} = 2^{h(v)+1}$ Knoten, also gilt die Behauptung.

Also haben wir jeweils logarithmische Höhe, wir erhalten also eine Komplexität von $O(m \log n)$.

$\square$

Damit kennen wir

- den Algorithmus
- seine Laufzeit (dank geeigneter Datenstruktur!)

Es fehlt noch: • die Korrektheit !!!

Dafür brauchen wir etwas Struktur...

2.3.2 Ein Struktursatz

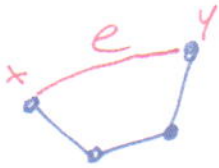
Satz 2.9.

Minimum Spanning Tree

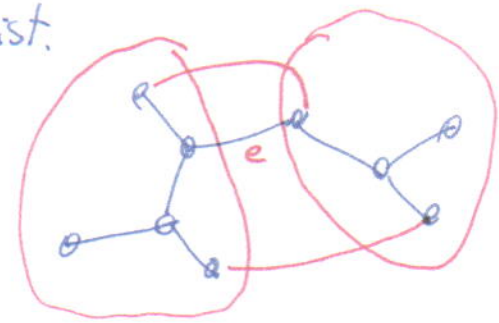
Sei (G,c) eine Instanz von MST, und sei T ein aufspannender Baum in G.
Dann sind die folgenden Eigenschaften äquivalent:

(a) T ist optimal.

(b) Für jede Kante $e = \{x,y\} \in E(G) \setminus E(T)$ hat keine Kante auf dem x-y-Pfad in T höhere Kosten als e.



(c) Für jedes $e \in E(T)$ ist e eine billigste Kante von $\delta(V(C))$, wobei C eine Zshgs.Komp. von $T-e$ ist.



Beweis:

(a) $\Rightarrow$ (b): (also $\neg(b) \Rightarrow \neg(a)$):

Falls (b) verletzt wäre, gäbe es eine Kante $e = \{x,y\} \in E(G) \setminus E(T)$ und eine Kante f auf dem x-y-Pfad in T mit $c(f) > c(e)$.

Dann wäre $(T-f)+e$ ein aufspannender Baum mit geringerem Gesamtgewicht, im Widerspruch zu (a).

(b) $\Rightarrow$ (c) (also $\neg(c) \Rightarrow \neg(b)$):

Sei (c) verletzt, d.h. sei $e \in E(T)$ und sei C eine Zusammenhangskomponente von $T-e$ und $f = \{x,y\} \in \delta(V(C))$ mit $c(f) < c(e)$.

Der x-y-Pfad in T muss eine Kante von $\delta(V(C))$ enthalten; da aber e die einzige derartige Kante ist, ist (b) verletzt.

$(c) \Rightarrow (a)$:

Angenommen, T erfüllt (c).
Sei T^* ein optimaler Baum mit $E(T^*) \cap E(T)$ maximal. Wir zeigen $T = T^*$.

Angenommen wir haben eine Kante $e = \{x, y\} \in E(T) \setminus E(T^*)$.
Sei C eine Zusammenhangskomponente von $T - e$;
$T^* + e$ enthält einen Kreis D. Da $e \in E(D) \cap \delta(V(C))$, muss noch eine weiter Kante $f \neq e$ von D zu $\delta(V(C))$ gehören.

$T^{**} = (T^* + e) - f$ ist ein aufspannender Baum.

Da T^* optimal ist, muss $c(e) \geq c(f)$ gelten.
Da aber (c) für T gilt, muss auch $c(e) \geq c(f)$ gelten; also ist $c(f) = c(e)$.
Damit ist auch $T^{**} = (T^* + e) - f$ ein optimaler aufspannender Baum, der eine Kante mehr mit T gemeinsam hat, im Widerspruch zur Eigenschaft von T^*.

Also gibt es keine Kante $e \in E(T) \setminus E(T^*)$,
d.h. $T = T^*$.

$\square$

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

Wie verwendet man 2.9(c) ?!

Algorithmus 2.11 (Prim)

Eingabe: G: Zshgder 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:

22.05.12

FALSCH: In Schritt (2) jeweils alle Kanten durchsehen
$\rightarrow O(n\,m)$

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

Initialisierung: $O(n)$.

$(n-1)$-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 $O(n^2)$ $\square$