Beweis:

O.B.d.A. ist $G$ einfach, also $m = O(n^2)$. Betrachte nur den kritischen Schritt (3).

Idee: Datenstruktur, die die jeweils vorhandenen Khythmverwaltet:

Kreis in (3) $\implies$ Kante verbindet zwei Knoten in derselben ZK.

Lösung: Für jede komponente wecken wir uns einen gerichteten Baum mit einer eindeutigen Wurzel, wobei jeder Knoten einen eindeutigen Vorgänger bekommt:

Das liefert eine Struktur $B$ mit $V(B) = V(T)$ und $|E(B)| = |E(T)|$. 

Ger. Baum logarithmische Höhe!
Wenn wir eine Kante $e_i = \{v, w\}$ in

3. überprüfen, finden wir die Wurzeln $v$ und $w$, zu den Komponenten für

$v$ und $w$. Zeitverbrauch hierfür: $O(\log n)$

(Stärke des Baumes!)

⇒ wird noch verifiziert!

Teste $r_v = r_w$?
Falls ja, überprüfe nächsten 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 $h(r_v) > h(r_w)$,
dann füge Kante $(r_v, r_w)$ in $B$ ein,
anzunst $e$.

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

Außerdem gilt sie nach einer Kantenfüllung, wenn \(h(x)\) sich nicht ändert.

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

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

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 \(e = \{x,y\} \in E(G) \setminus E(T)\) ist \(e\) keine Kante auf dem \(x\)-\(y\)-PFd in \(T\), das für alle Kanten \(e \in E(T)\) eine höhere Kosten als \(e\).

(c) Für jedes \(e \in E(T)\) ist \(e\) eine billigste Kante von \(\delta(V(C))\), wobei \(C\) eine Blockkomponente von \(T-e\) ist.
Beweis:

(a) \Rightarrow (b): (also \tau(b) \Rightarrow \tau(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, in Widerspruch zu (a).

(b) \Rightarrow (c) (also \tau(c) \Rightarrow \tau(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 $e \in E(D) \setminus \delta(V(C))$ muss $e$ noch eine weitere 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, in Widerspruch zur Eigenschaft von $T^*$.

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

\[ \text{d.h.} \quad T = T^* \]
Satz 2.10

Kruskals Algorithmus funktioniert korrekt.

Beweis:
Der Algorithmus liefert einen maximalen Kreisfreien Teilgraphen, also nach Satz 2.6 (5) einen aufspannenden Baum. Außerdem ist Bedingung (6) von Satz 2.9 erfüllt, also ist $T$ optimal.

2.3.3 Prim's Algorithmus
Wie verwendet man 2.9(6) ?

Algorithmus 2.11 (Prim)

Eingabe: $G$: Einfacher ungerichteter Graph
$C$: Kantengewichte $E(G) \rightarrow IR$

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 $T := T + e$. 