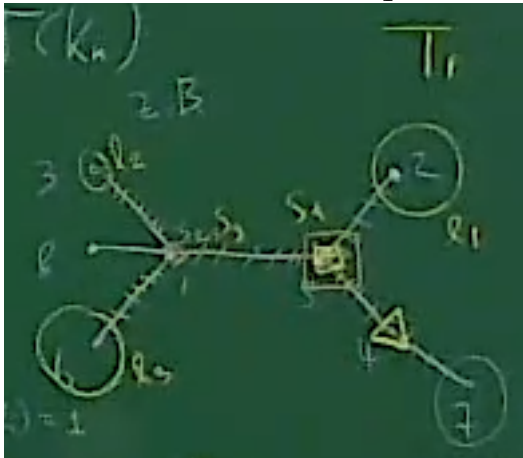


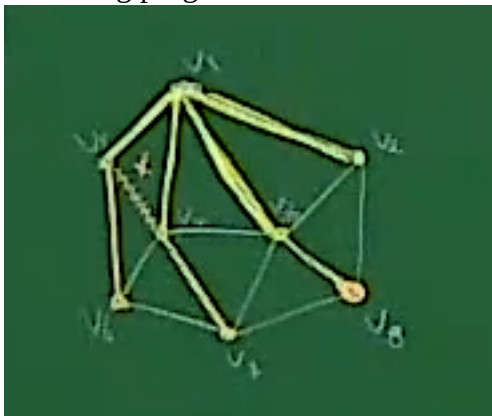
Prüfer-Code

- Bijektion $f: T_n \rightarrow S_{n-2}, T \rightarrow f(T) = s$
 - $S_{n-2} = \{s = (s_1 s_2 \dots s_{n-2}), s_i \in [n]\}$
 - sukzessiv definiert
 - * $T_0 = T$
 - * T_i
 - ♦ nehme kleinste Blatt l_i in $T_i - 1$
 - ♦ entferne l_i und inzidente Kante von $T_i - 1$
 - ♦ definiere i-te Folgenglied s_i als Nachbar von l_i
- Verfahren retourniert Folge S_T



Bread-First-Search

- branching progress



- Input: zusammenhängender Graph G
- Output: Spannbaum T
- Verfahren:
 - wähle Knoten x_0 als Wurzel
 - * Liste $L = (x_0)$
 - Loop bis T alle Knoten enthält $V(T) = V(G)$

- * gegebene Liste $L = (x_0, x_1, \dots)$ und Baum T
- * nimm ersten Knoten x von L
- * falls x keine Nachbarn hat, welche noch nicht im Baum sind
 - ◆ $N(x)/V(T) = \emptyset$
 - ◆ entferne x aus L
- * sonst
 - ◆ füge einen Nachbarn y aus $N(x)/V(T)$ zu T und L hinzu

$$\begin{aligned} V(T) &\leftarrow V(T) \cup \{y\} \\ E(T) &\leftarrow E(T) \cup \{[x, y]\} \\ L &= (x, x', x'', \dots, y) \end{aligned}$$

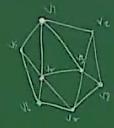
– return T

• Beispiel

Bsp.

Im Verlauf des BFS

- L aktuelle Liste
- T aktueller Baum
- x das erste Element von L
- $N_G(x)$ Nachbarn von x
- $y \in N_G(x) \setminus V(T)$



(I) Zum Beginn wähle einen Knoten, z.B. $v_1 \in V$, als Wurzel
 setze $L = (v_1)$
 $T = (V(T), E(T))$ mit $V(T) = \{v_1\}$, $E(T) = \emptyset$

(II) Nach Schritt (I) von (I) nimmt das erste Element $x = v_1$ aus der Liste $L = (v_1)$
 $N_G(x) = \{v_2, v_3, v_4, v_5\}$
 wegen $N_G(x) \setminus V(T) \neq \emptyset$, nach Schritt (I) nimmt man ein $y \in N_G(x) \setminus V(T)$
 z.B. $y = v_2$
 setze $V(T) = \{v_1, v_2\}$, $E(T) = \{[v_1, v_2]\}$, $L = (v_1, v_2)$

(III) Nach (II) nimmt $x = v_1$
 Nach (II) $y \in N_G(x) \setminus V(T) = \{v_3, v_4, v_5\}$
 setze $V(T) = \{v_1, v_2, v_3\}$
 $E(T) = \{[v_1, v_2], [v_1, v_3]\}$
 $L = (v_1, v_2, v_3)$

(IV) Nach (III) nimmt $x = v_2$
 Nach (III) $y \in N_G(x) \setminus V(T) = \{v_4\}$
 d.h. $y = v_4$
 setze $V(T) = \{v_1, v_2, v_3, v_4\}$
 $E(T) = \{[v_1, v_2], [v_1, v_3], [v_2, v_4]\}$
 $L = (v_1, v_2, v_3, v_4)$

(V) Nach (IV) nimmt $x = v_3$
 Nach (IV) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_2, v_3, v_4, v_5)$

(VI) Nach (V) nimmt $x = v_2$
 Nach (V) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_3, v_4, v_5, v_6)$

(VII) Nach (VI) nimmt $x = v_3$
 Nach (VI) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_4, v_5, v_6, v_7)$

(VIII) Nach (VII) nimmt $x = v_4$
 Nach (VII) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_5, v_6, v_7, v_8)$

(IX) Nach (VIII) nimmt $x = v_5$
 Nach (VIII) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_6, v_7, v_8)$

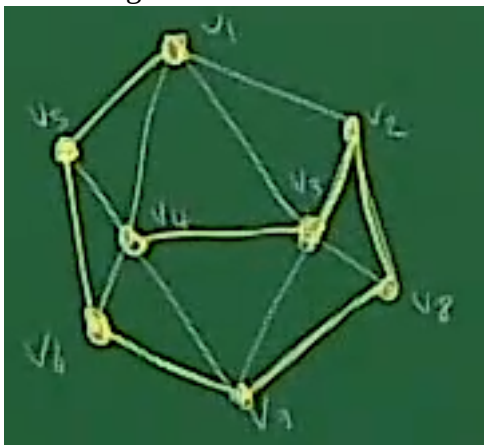
(X) Nach (IX) nimmt $x = v_6$
 Nach (IX) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_7, v_8)$

(XI) Nach (X) nimmt $x = v_7$
 Nach (X) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = (v_8)$

(XII) Nach (XI) nimmt $x = v_8$
 Nach (XI) $N_G(x) \setminus V(T) = \emptyset$
 setze $L = ()$

Depth-First-Search

- findet langen Pfad



- Input: zusammenhängender Graph G

- Output: Spannbaum T
- Verfahren fast ident zu BFS
 - jedoch wird y am Anfang von L hinzugefügt

$$L = (y, x, x', x'', \dots, w)$$

*

Algorithmus von Kruskal

- Input:
 - zus. Graph G mit n Knoten und m Kanten
 - Gewichtsfunktion w
 - * $E \rightarrow$
 - * $e \rightarrow w(e)$
- Output:
 - Spannbaum T von G mit minimalem Gewicht
 - Summe aller Kantengewichte ist minimal

$$\sum_{e \in E(T^*)} w(e) \leq \sum_{e \in E(T)} w(e) \quad \forall T \in \mathcal{T}(G)$$

$\mathcal{T}(G) = \{ \text{Spannbäume von } G \}$

- Verfahren
 - sortiere (nummeriere) Kanten aufsteigend nach Gewicht

$$E = (e_1, e_2, \dots, e_m) \text{ mit } w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$$

*

- setze $E(T) =$
- Loop bis $|E(T)| = n - 1$
 - * nimm die kleinste Kante e_n
 - * füge e_n zu T hinzu, wenn das keinen Kreis erzeugt
 - ♦ sonst wird e_n aus der Liste entfernt
- return T

[[Bäume & Spannbäume]]