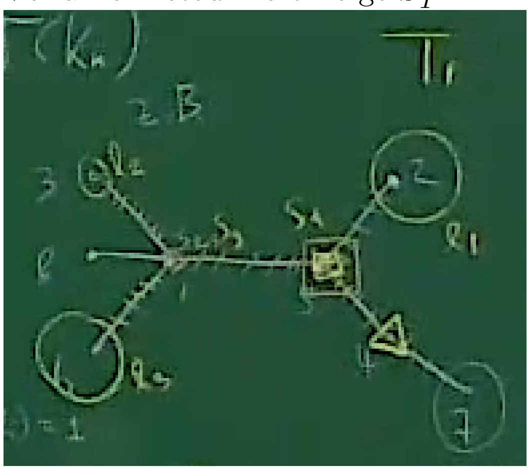
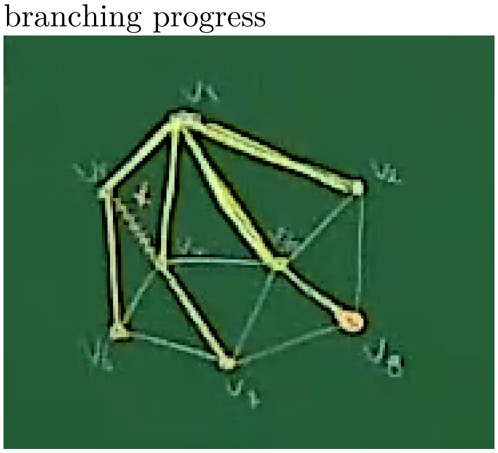
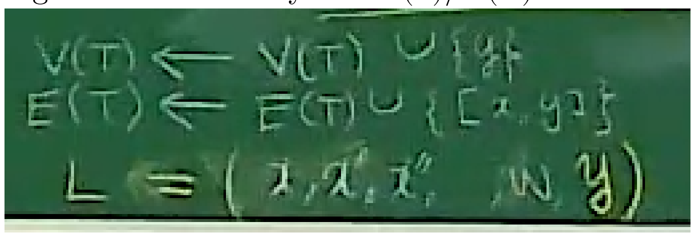
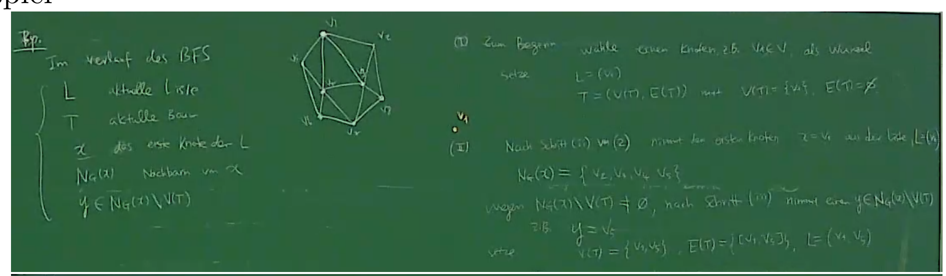
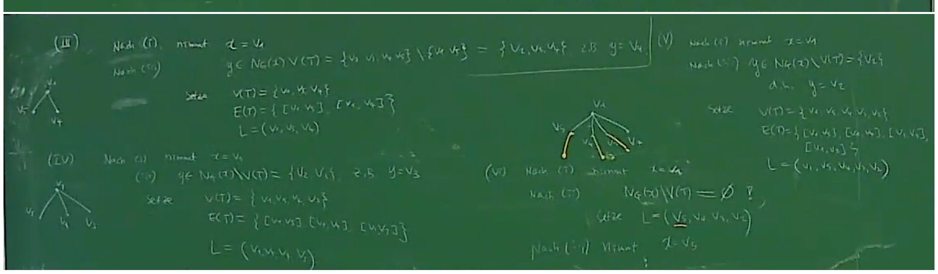
Prüfer-Code

- Bijektion f: $T_n{-}{>}S_{n-2}, T{-}{>}f(T) = s$
  - $S_{n-2} = \{s = (s_1 s2....s_{n-2}), s_i \in [n]\}$
  - 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
- 

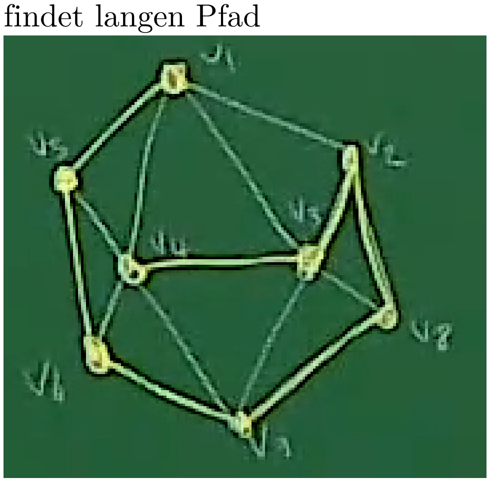
Bread-First-Search

- 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 $V(T) = V(G)$

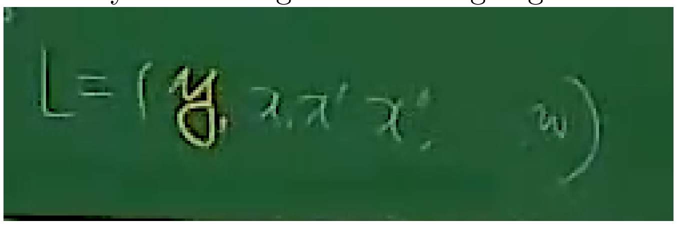
        - gegeben Liste $L = (x_0, x_1, ...)$ und Baum T
        - nimm ersten Knoten x von L
        - falls x keine Nachbarn hat, welche noch nicht im Baum sind
            - $N(x)/V(T) = \emptyset$
            - entfern x aus L
        - sonst
            - füg einen Nachbarn y aus $N(x)/V(T)$ zu T und L hinzu
            - 

    - return T
- Beispiel
    - 
    - 

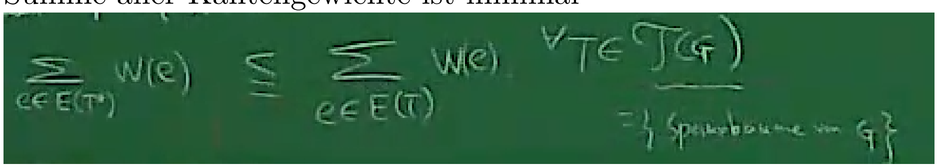
Depth-First-Search

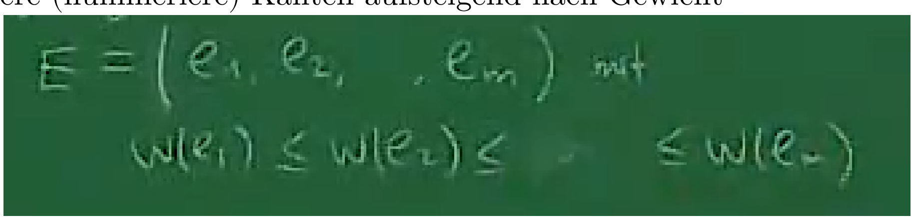
- findet langen Pfad
- 

- Input: zusammenhängender Graph G

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

Algorithmus von Kruskal

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

- Verfahren
  - sortiere (nummeriere) Kanten aufsteigend nach Gewicht
    - 

  - 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]]