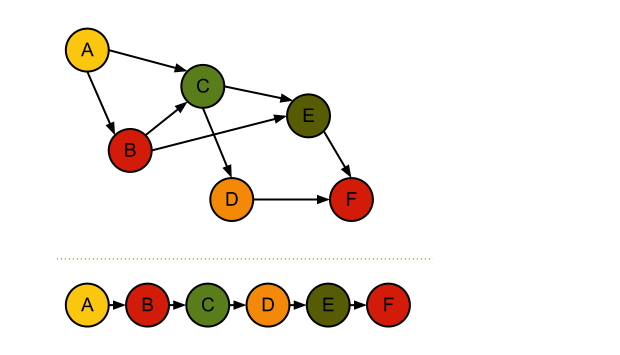
**Algorithmik WS 2019/20**

**D Rot Topologische Sortierung**



Bastian Abt Finn Wehn Jannick Alexander Philipp Zeitel

inf1405 inf1390 inf1252 inf1511

Inhaltsverzeichnis

[Einführung 3](#_Toc23920581)

[Mathematisches Prinzip 3](#_Toc23920582)

[Definition der topologischen Sortierung 3](#_Toc23920583)

[Abstraktes Beispiel 4](#_Toc23920584)

[Pseudo Code 5](#_Toc23920585)

[Programmierung 6](#_Toc23920586)

[Methoden zur Topologischen Sortierung 6](#_Toc23920587)

[Laufzeitverhalten 7](#_Toc23920588)

[Beispiele 8](#_Toc23920589)

[Quellen 8](#_Toc23920590)

# Einführung

Das topologische Sortieren findet in graphbasierten Anwendungen ihren praktischen Einsatz.

Gegeben ist ein azyklischer gerichteter Graph (directed acyclic graph / DAG), dies bedeutet der Graph kann niemals einen Zyklus durchlaufen, wenn man den Kanten in angegebener Richtung folgt. Sind diese Voraussetzungen erfüllt, kann so eine Reihenfolge der Knoten gesucht werden, so dass jeder Knoten nach all seinen Vorgängern kommt. Es gibt also keine "Rückkanten".

Dabei spielt es keine Rolle, dass mehrere Knoten ohne Vorgänger existieren oder mache Knoten mehrere Vorgänger oder Nachfolger haben.

Eine topologische Sortierung muss nicht eindeutig sein! Je nach Graph können mehrere Sortierungen richtig sein.

Interpretationsmöglichkeiten für Sachzusammenhänge sind zum Beispiel kausale oder zeitliche Abhängigkeiten. Ist dies der Fall spricht man auch vom Scheduling-Problem.

Topologische Sortierung bezeichnet in der Mathematik eine Reihenfolge von Dingen, bei der vorgegebene Abhängigkeiten erfüllt sind.

# Mathematisches Prinzip

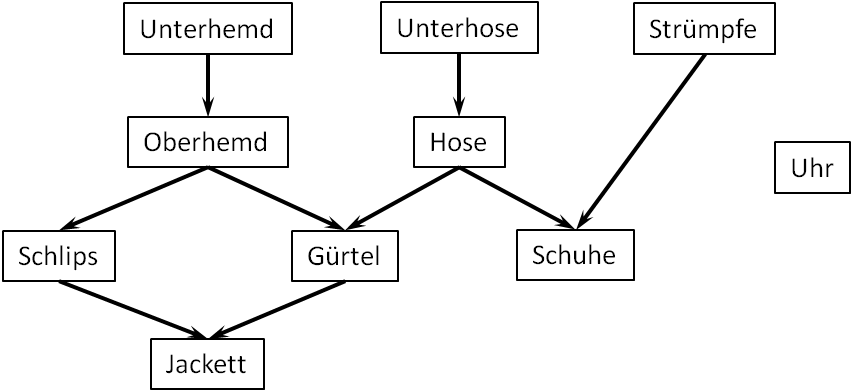
## Definition der topologischen Sortierung

Die Objekte in einer zu sortierenden Menge unterliegen einer Halbordnung. Ist diese irreflexiv (a steht nicht mit a in Relation) und transitiv (wenn a R b und b R c, dann a R c), spricht man von einer topologischen Sortierung der Menge. Diese ist nun eine lineare Ordnung.

Eine topologische Sortierung eines gerichteten Graphen ist also eine Nummerierung seiner Knoten derart, dass aus folgt in der Nummerierung, oder auch u kommt vor v.

## Abstraktes Beispiel

Der folgende Graph ist zyklusfrei und gerichtet, das heißt es existiert mindestens eine topologische Sortierung. Die Knoten *V* des Graphen entsprechen den Elementen und die Kanten beschreiben die Abhängigkeiten.



http://alda.iwr.uni-heidelberg.de/index. 1

Elemente: Unterhemd, Unterhose, Strümpfe, Oberhemd, Hose, Schlips, Gürtel, Schuhe, Jackett, Uhr

Relationen: Unterhemd vor Oberhemd

Oberhemd vor Schlips

Oberhemd vor Gürtel

Schlips vor Jackett

Gürtel vor Jackett

Unterhose vor Hose

Hose vor Gürtel

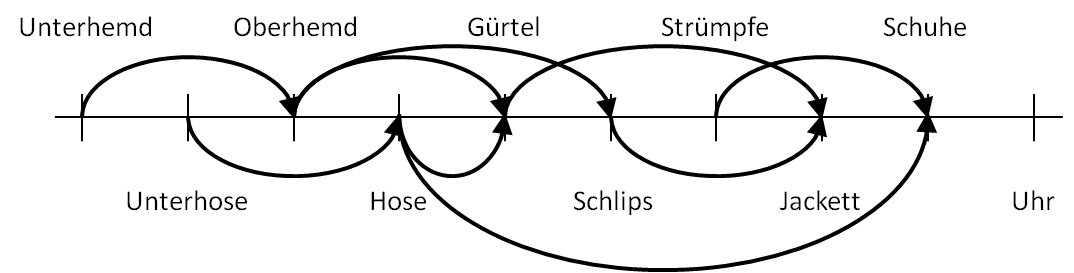
Hose vor Schuhe

Strümpfe vor Schuhe

Uhr

Eine topologische Sortierung beschreibt nun eine sequentielle Folge der Kleidungsstücke, zum Beispiel:

(Unterhemd, Unterhose, Oberhemd, Hose, Gürtel, Schlips. Strümpfe, Jackett, Schuhe, Uhr)



http://alda.iwr.uni-heidelberg.de/index. 2

## Pseudo Code

//Überprüfen ob der Graph azyklisch ist

val g = Graph() Graph erstellen

g.addNode(a,b) mehrere Knoten hinzufügen

while(i=0, i<g.size) gehe sukzessiv zu allen Knoten mit indegree 0

if(g[i].indegree==0)

array.add(g[i]) wenn indegree 0 dann schreibe Knoten in ein array

g[i].neighbors.indegree -= 1 dekrementiere indegree der Nachbarn um jeweils 1

i++

//Wenn der Graph azyklisch ist, enthält das Array am Ende alle Knoten des Graphen

//Vorgehen topologische Sortierung

1) Graph erstellen

2) Knoten mit 0 Vorgängern (this.indegree) bestimmen

3) Einen Beliebigen dieser "Null-Vorgänger-Knoten" auswählen und in Speicher schreiben

-> Reihenfolge irrelevant:

Bei Stack wird zuerst der letzte Knoten genommen

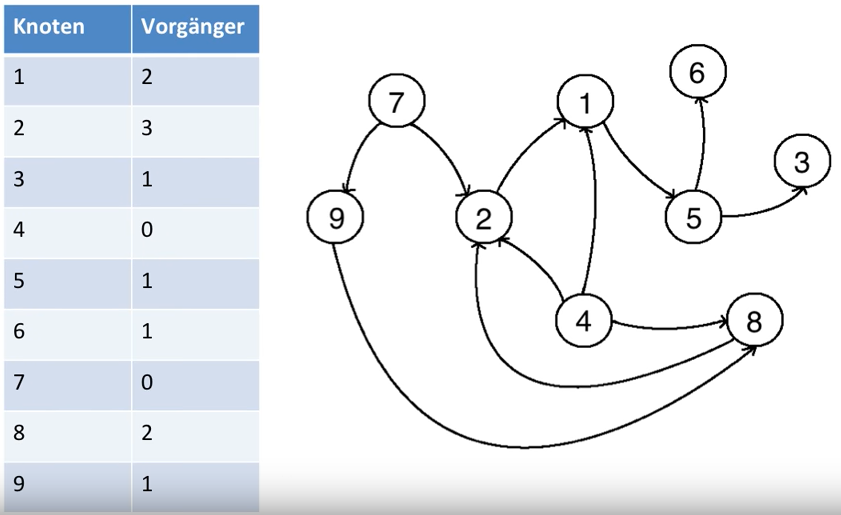
Bei Queue wird zuerst der erste Knoten genommen

4) Diesen Knoten markieren (zB -1 oder visited) und in Output/Array poppen

5) Bei allen Nachfolgern dieses Knotens this.indegree um 1 dekrementieren

-> wenn bei einem dieser Knoten this.indegree 0 wird, in Speicher schreiben (Stack/Queue)

6) Durch alle Schritte iterieren 2-5 bis Output/Array gefüllt ist



# Programmierung

## Methoden zur Topologischen Sortierung

topologicalSort():

override fun topologicalSort(): ArrayList<Node> {

zeroNodes = findRoots().toMutableList()

val out = arrayListOf<Node>()

while (zeroNodes.isNotEmpty()) {

inductiveStep(zeroNodes.first(), out)

}

}

* „zeroNodes“ verweist auf eine MutableList<Node>, in der alle Knoten enthalten sind, welche keinen Vorgänger-Knoten besitzen. D.h. jeder Knoten besizt einen „indegree“-Wert von 0 (mit findRoots(): List ermittelt)
* „out“ verweist auf eine arrayList<Node>, in der später der topologisch sortierte Graph gespeichert wird

inductiveStep():

private fun inductiveStep(nodeToProcess: Node, out: ArrayList<Node>): ArrayList<Node> {

nodeToProcess.successors.forEach {

it.indegree--

if (it.indegree == 0) {

zeroNodes.add(it)

}

it.predecessors.remove(nodeToProcess)

}

zeroNodes.remove(nodeToProcess)

out.add(nodeToProcess)

return out

}

* Nun wird jeder Knoten aus zeroNodes aus der Liste gelöscht und in das Ausgabearray „out“ gespeichert.
* Außerdem werden alle Nachfolger, des jeweiligen Knotens, der „zeroNodes“-Liste hinzugefügt

## Laufzeitverhalten

Sei die Anzahl der Elemente (Knoten) und die Anzahl der Beziehungen (Kanten), dann ist ein Graph bestimmt durch . Benötigte Hilfsdatenstrukturen sind ein Array indegree[v] für die Anzahl der nicht besuchten Vorgänger und ein Speicher Queue/Stack der alle Knoten enthält die als nächstes besucht werden.

Die Initialisierung von indegree[v] hat also eine Laufzeit von O(|E|) und der Stack O(|V|).

Gesamtlaufzeit: O(|V| + |E|)

# Beispiele

Topologisches Sortieren wird unteranderem zur Zeitablaufsplanung benutzt (scheduling jobs). Hierbei kann anhand gegebener Abhängigkeiten bestimmter auszuführender Aufgaben eine zeitliche Sortierung erstellt werden.

Alternative zur Blockchain: DAG-Konstruktionen sind verwandt mit der Blockchain. Der Hauptalgorithmus hierbei wird als topologische Ordnung bezeichnet. Dies bedeutet, dass jede Kante von einer früheren Kante zu einer späteren Kante gerichtet wird.

Ordnerstrukturen und Zugriffsberechtigung: Eine Datei darf erst gelesen werden wenn eine andere Datei vorher dies bestätigt.

Makefiles und Programmaufrufe

# Quellen

Noch zu tun:

* Wege-Suche einfügen
* Vorbedingung einfügen
* Laufzeit kontrollieren
* Ggf. Pseudo Code weiter abstrahieren
* Beispiele mit Graphen suchen
* **Quellen**
* Feinschliff