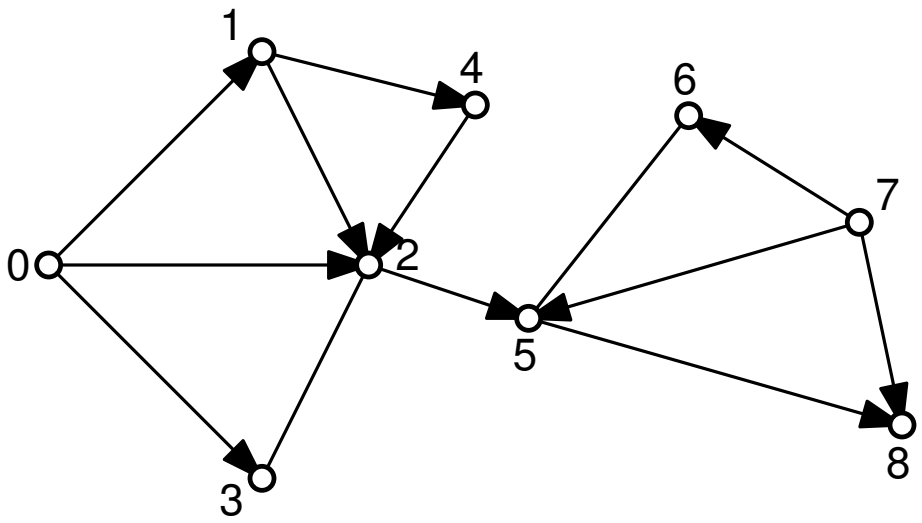
Karlsruher Institut für Technologie

Tutorium Algorithmen 1

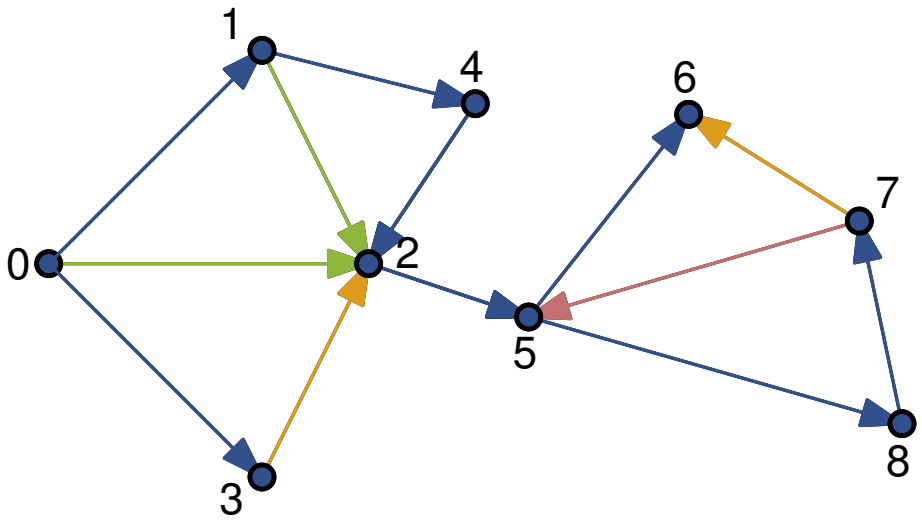
10 · Mehr DFS und Toposort · 1.7.2024
Peter Bohner Tutorium 3

DFS in gerichteten Graphen

- Funktioniert sehr ähnlich wie die DFS auf ungerichteten Graphen
- Mehr Kantentypen im DFS Baum
- Wir definieren die **FIN** Nummer um die Kantentypen unterscheiden zu können
  - Gibt an wann ein Knoten abgearbeitet wurde (Knoten mit kleiner FIN Nummer werden zuerst abgearbeitet)



Beispiel



Schwarz: Knoten

Rot: DFS Nummer

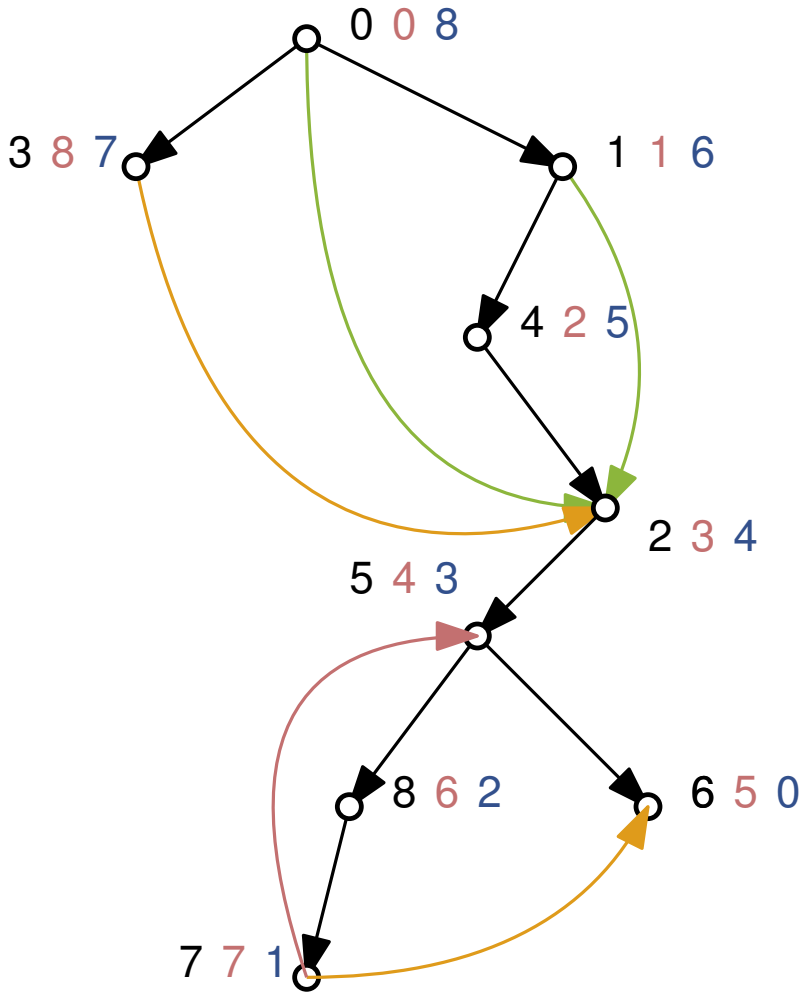
Blau: FIN Nummer

Schwarze Kanten: Baumkanten

Rote Kanten: Rückkanten

Grüne Kanten: Vorkanten

Orange Kanten: Queranten



Nicht-Baumkanten

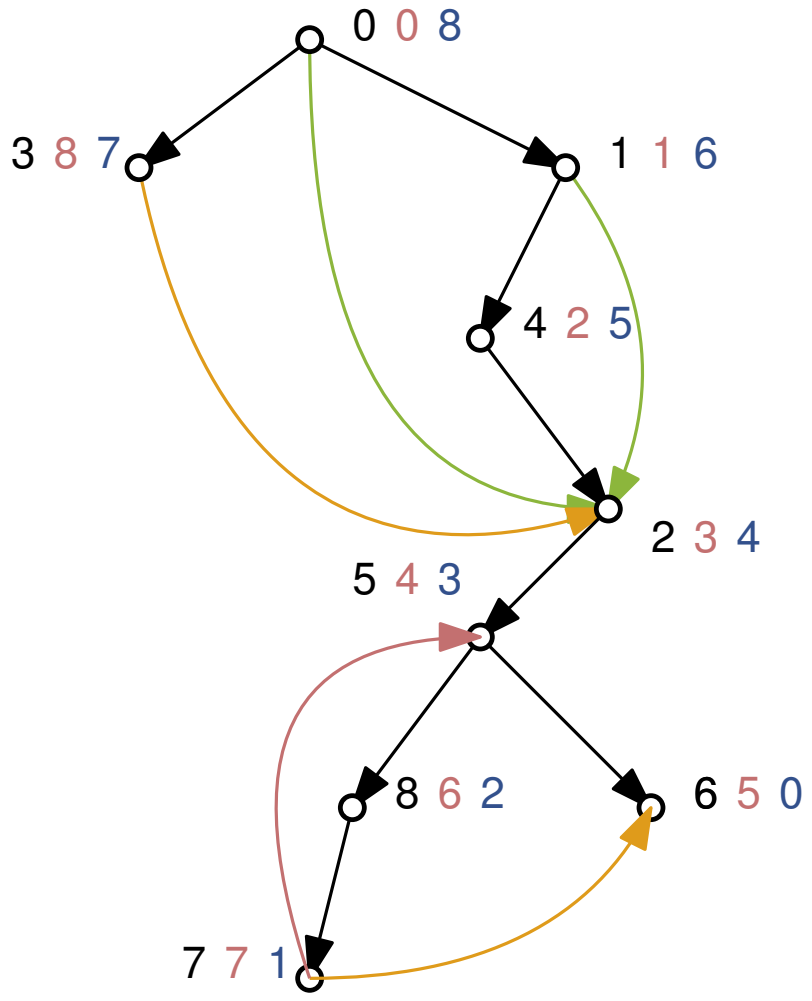
Rückkanten Kennen wir schon
DFS Nummer groß → klein

Vorkanten Kante zu Nachfolger im selben Teilbaum
DFS Nummer klein → groß

Querkanten Kante in einen anderen Teilbaum
DFS Nummer groß → klein

⇒Wir brauchen die FIN Nummer um Quer und Rückkanten zu unterscheiden

	DFS-Nummer	FIN-Nummer
Vorkante	klein → groß	groß → klein
Rückkante	groß → klein	klein → groß
Querkante	groß → klein	groß → klein



DFS

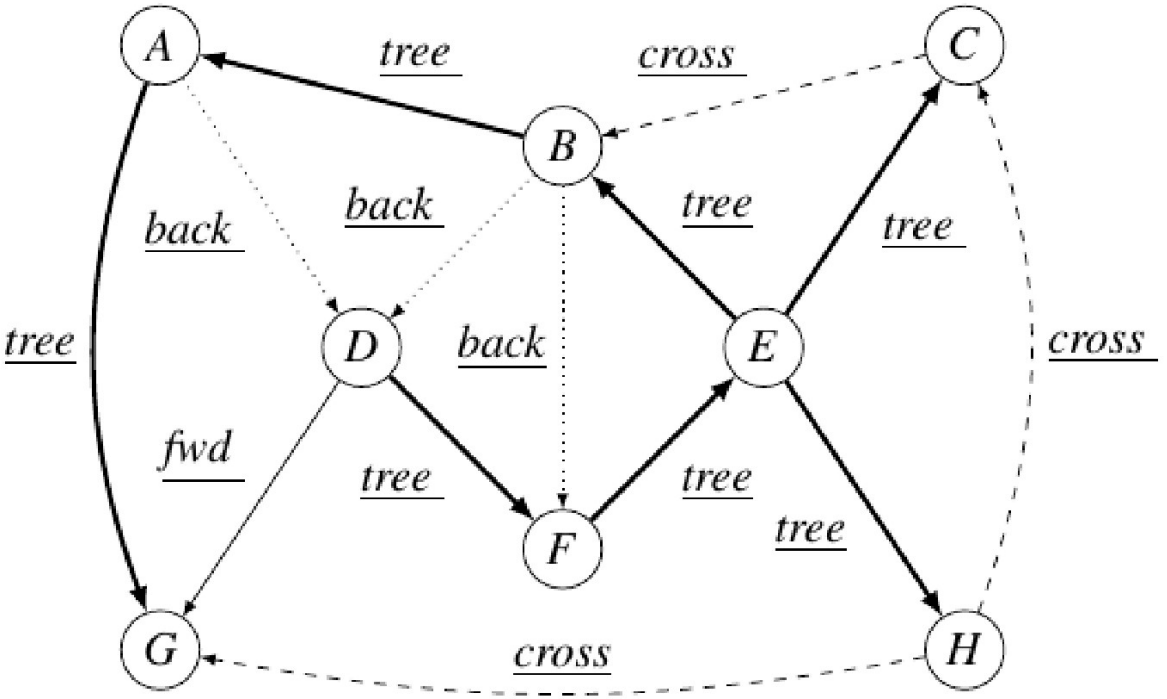
```
DFS:
    NUM, FIN : [ℕ0; n] = [⊥, ..., ⊥]
    num, fin := 0
    DFS(Node u)
        NUM[u] := num
        num++
        for v ∈ N(u) do
            if NUM[v] = ⊥ then
                DFS(v)
        FIN[u] := fin
        fin++
```

- Für was für gerichtete Graphen + Startknoten ist der DFS-Baum eindeutig?
  - vom Startknoten aus (nur) Baum mit beliebigen Rückkanten erreichbar
- Zeige oder widerlege: Die Anzahl der Baumkanten ist für jeden Tiefensuchbaum auf einem zusammenhängenden, ungerichteten Graphen mit n Knoten gleich.
  - Baumkante zu Knoten genau dann, wenn Knoten erstes Mal gefunden und Knoten nicht der Startknoten
  - Graph zusammenhängend, ungerichtet $\Rightarrow$jeder Knoten wird (genau einmal zum ersten Mal) gefunden
  - daher: Jeder Knoten außer dem Startknoten hat genau eine eingehende Baumkante

Aufgabe

Betrachte die nebenstehende Kantenklassifikation.

- Ist dies ein mögliches Ergebnis einer Tiefensuche? Falls ja, gib eine mögliche Reihenfolge der Knoten an, in der sie zum ersten Mal gefunden wurden
- Ja: Die Reihenfolge muss D-F-E-B-A-G-C-H gewesen sein
- Ist dies ein mögliches Ergebnis einer Breitensuche? Falls ja, gib eine mögliche Reihenfolge der Knoten an, in der sie zum ersten Mal gefunden wurden
- Nein: Als mögliche Startknoten kommen nur F oder E in Frage, dann müsste aber (H,G) eine Baumkante sein



Topo Sort

Definition
Sei $G = (V, E)$ ein gerichteter Graph. Eine **topologische Sortierung** ist eine totale Ordnung der Knoten V, sodass jede Kante von kleinerem zu größerem Knoten zeigt.

- Topologische Sortierungen existieren nur auf kreisfreien Graphen. Warum?

Angenommen es gibt einen Graph G der einen Kreis enthält und eine topologische Sortierung besitzt.

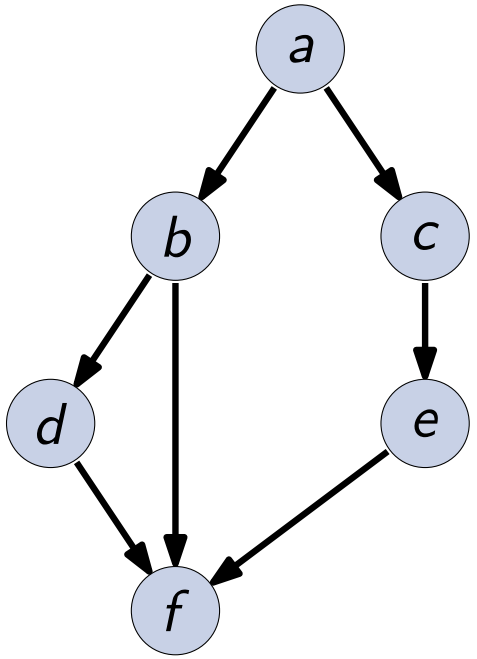
Sei P ein Kreis in G mit mindestens 2 Knoten v_1 und v_2

Da v_2 von v_1 aus erreichbar ist gilt aufgrund der topologischen Sortierung und der Transitivität von $>$, das $v_1 > v_2$

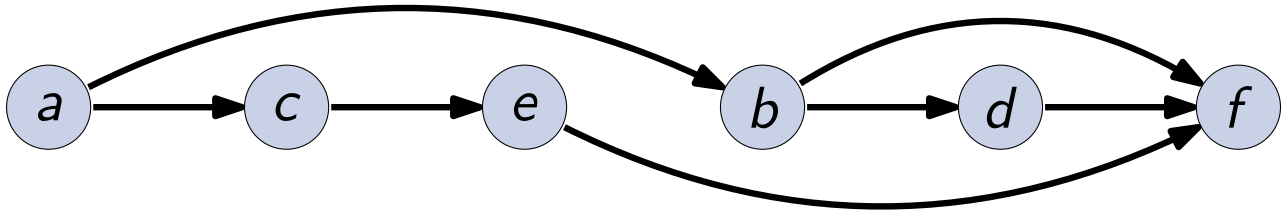
Mit der gleichen Argumentation gilt auch $v_2 > v_1 \Rightarrow$also $v_1 > v_2 \wedge v_2 > v_1$ bzw. $v_1 > v_1$ also insb. $v_1 \neq v_1 \Rightarrow$Widerspruch

Ein gerichteter kreisfreier Graph heißt auch **DAG** (eng. **D**irected **A**cyclic **G**raph)

Beispiel



$\Rightarrow$



Die Sortierung ist nicht Eindeutig
(Hier könnte man z.B. b und e tauschen)

Topo Sort

Die Parameter seien wie bei der DFS initialisiert.

TopoSort(*Graph* $G = (V, E)$):

```
for Node v ∈ V do
    if FIN[v] = ⊥ then
        DFS(G, v)
return V absteigend sortiert nach FIN
```

Wieso macht das sortieren die Laufzeit nicht kaputt?

FIN Nummern sind nicht größer als n $\Rightarrow$wir können Bucketsort oder ähnliches verwenden

- Laufzeit?
  - Wir können jede Kante maximal zweimal betrachten und wir besuchen jeden Knoten genau einmal
    $\Rightarrow \Theta(n + m)$
- Wenn G einen Kreis enthält, so existiert keine topologische Sortierung. Wie stellen wir das fest?
  - Sobald wir eine Rückkante finden, gibt es einen Kreis (siehe DFS)

Aufgabe: Längster Pfad

Finde einen Algorithmus, der in einem (positiv und negativ) gewichteten DAG (gerichteten, kreisfreien Graphen) G die längsten Wege von einem Knoten s zu allen anderen erreichbaren Knoten in Linearzeit findet.

- Wie hilft hier die topologische Sortierung?

Lösung:

- Berechne topologische Sortierung von G
- Relaxiere Kanten in topologischer Reihenfolge wie bei Bellman-Ford

Aufgabe: Deadlocks

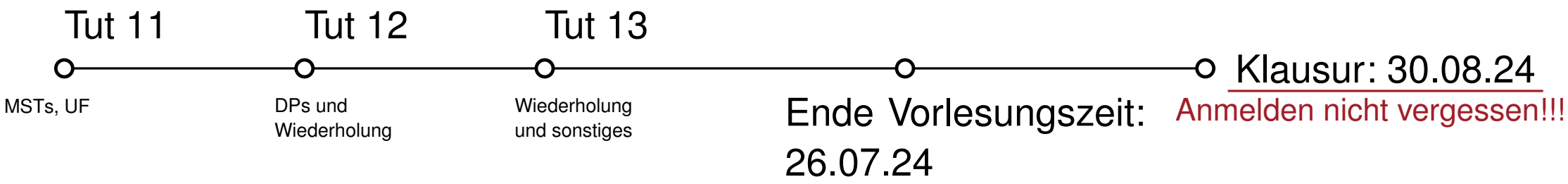
In einem Rechnersystem gibt es n Prozesse $P = \{p_1, \ldots, p_n\}$ sowie k Ressourcen $R = \{r_1, \ldots, r_k\}$, die von diesen benötigt werden. Jeder Prozess p_i benötigt dabei die Teilmenge $R_i \subseteq R$ der Ressourcen und hält zu Beginn bereits die Ressourcen $\tilde{R}_i \subseteq R_i$. Wird ein Prozess ausgeführt, nimmt er sich alle restlichen Ressourcen, die er benötigt, und beendet sich dann erfolgreich, wobei er die genutzten Ressourcen wieder frei gibt.
Finde eine Reihenfolge, in der alle Prozesse nacheinander ausgeführt werden können,
sodass jede Ressource zu jedem Zeitpunkt von höchstens einem Prozess gehalten wird,
oder stelle fest, dass das nicht möglich ist.

- **Hinweis:** Finde zuerst eine Modellierung als Graph.

Lösung: Deadlocks

- Stelle Prozesse und Ressourcen als Knoten dar.
  Hält p_i r_i, so füge Kante (p_i, r_i) ein.
  Benötigt p_i r_i und hält dieses noch nicht, so füge Kante (r_i, p_i) ein.
- Berechne eine topologische Sortierung, die Reihenfolge der Prozesse in dieser ist die gesuchte Ausführungsreihenfolge

Roadmap



Falls ihr spezielle Themen nochmal wiederholen wollt, ruhig Bescheid geben (per Discord, E-Mail, auf einem Übungsblatt...)

Zusammenfassung

Was haben wir gemacht?

- DFS auf gerichteten Graphen
- DAGs
- Toposort

Worauf könnt ihr euch nächste Woche freuen?

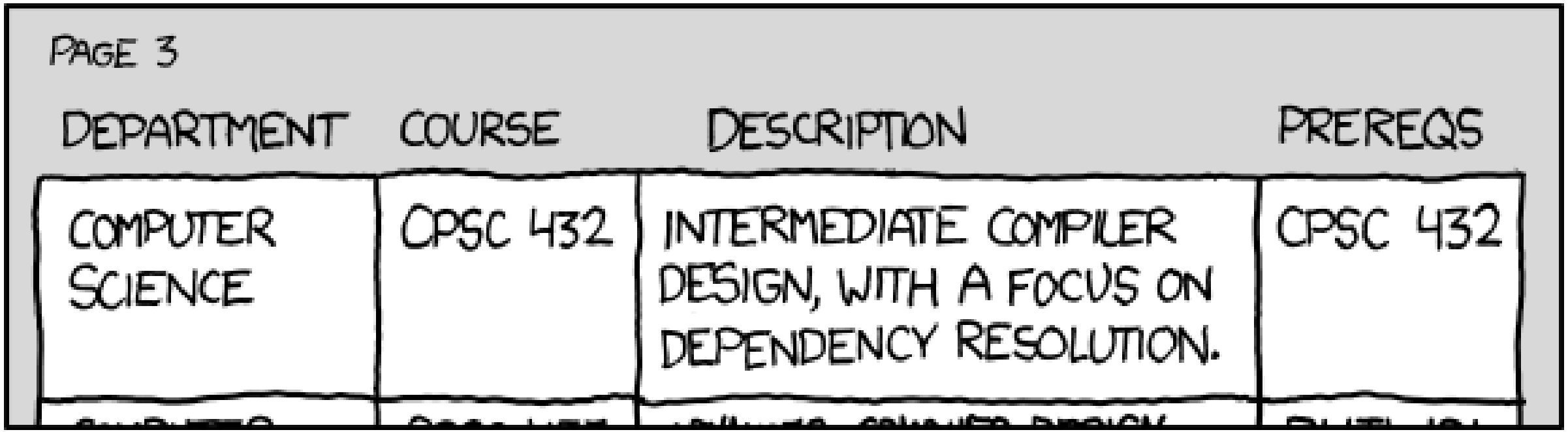
- Spannbäume
- Union Find (die letzte Datenstruktur der Vorlesung)

Fragen?

Karlsruher Institut für Technologie

Fragen!

Ende



https://xkcd.com/754/