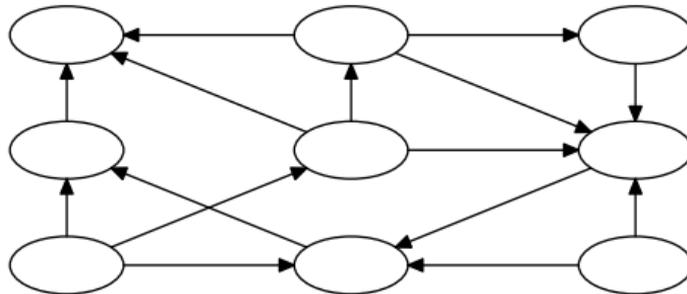


# Spezialfall DAG

Können wir kürzeste Pfade in DAGs schneller finden?

## Theorem

*Kürzeste Pfade von einem Knoten  $s$  in einem DAG können in linearer Zeit gefunden werden.*

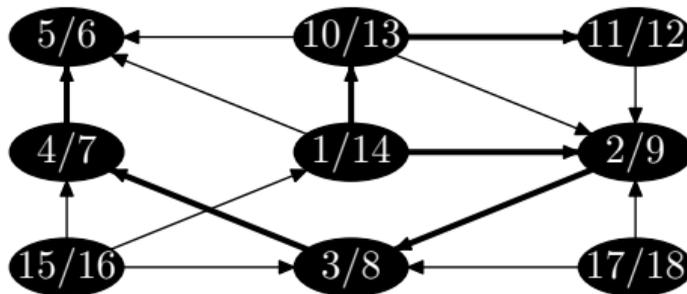


# Spezialfall DAG

Können wir kürzeste Pfade in DAGs schneller finden?

## Theorem

*Kürzeste Pfade von einem Knoten  $s$  in einem DAG können in linearer Zeit gefunden werden.*



# Spezialfall DAG

Können wir kürzeste Pfade in DAGs schneller finden?

## Theorem

*Kürzeste Pfade von einem Knoten  $s$  in einem DAG können in linearer Zeit gefunden werden.*

## Beweis.

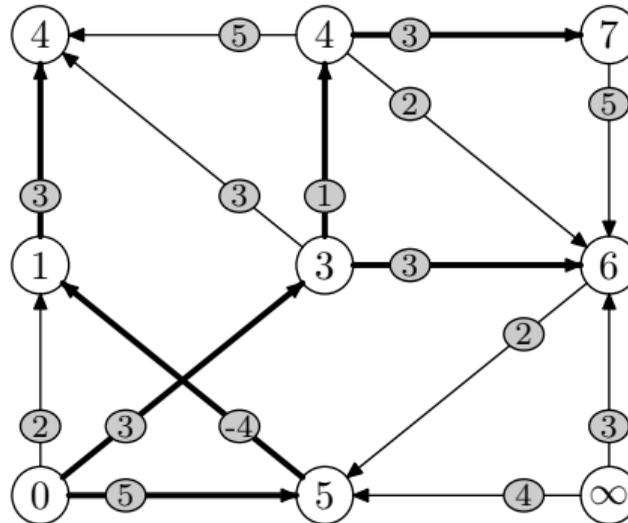
Relaxiere Knoten in topologischer Reihenfolge.

Laufzeit:  $O(|V| + |E|)$ .

Korrektheit: Jeder Knoten, der relaxiert, kennt zu diesem Zeitpunkt seinen echten Abstand. □

Frage: Sind negative Gewichte hier erlaubt?

# Kürzeste Wege mit negativen Kantengewichten



Dijkstra funktioniert nicht mit negativen Kantengewichten.

# Der Algorithmus von Bellman und Ford

Idee:

- Relaxiere alle Kanten
- Wiederhole dies, bis keine Änderung

Warum korrekt?

Induktion über die Länge eines kürzesten Pfads.  
(Also genügen  $n$  Wiederholungen)

Was passiert bei Kreisen mit negativem Gewicht?

→ Keine Terminierung.

# Der Algorithmus von Bellman und Ford

Idee:

- Relaxiere alle Kanten
- Wiederhole dies, bis keine Änderung

Warum korrekt?

Induktion über die Länge eines kürzesten Pfads.  
(Also genügen  $n$  Wiederholungen)

Was passiert bei Kreisen mit negativem Gewicht?

→ Keine Terminierung.

# Der Algorithmus von Bellman und Ford

Idee:

- Relaxiere alle Kanten
- Wiederhole dies, bis keine Änderung

Warum korrekt?

Induktion über die Länge eines kürzesten Pfads.  
(Also genügen  $n$  Wiederholungen)

Was passiert bei Kreisen mit negativem Gewicht?

→ Keine Terminierung.

# Der Algorithmus von Bellman und Ford

## Algorithmus

```
function Bellman – Ford(s) boolean :  
for v in V do d[v] := infty od;  
d[s] := 0;  
for i = 1 to |V| – 1 do  
  forall(v, u) in E do  
    d[u] := min { d[u], d[v] + length(v, u) }  
  od  
od;  
forall(v, u) in E do  
  if d[u] > d[v] + length(v, u) then return false fi  
od;  
return true
```

# Der Algorithmus von Bellman und Ford

## Theorem

*Gegeben sei ein gerichteter Graph  $G = (V, E)$  mit Kantengewichten  $E \rightarrow \mathbf{R}$  und ein Knoten  $s \in V$ .*

*Wir können in  $O(|V| \cdot |E|)$  feststellen, ob ein Kreis mit negativem Gewicht existiert, und falls nicht, die kürzesten Wege von  $s$  zu allen Knoten berechnen.*

## Beweis.

Wir haben die Korrektheit bereits nachgewiesen.

Zur Laufzeit: Jeder Knoten wird  $|V|$  mal relaxiert, also wird auch jede Kante  $|V|$  mal relaxiert (in konstanter Zeit). □

# Der Algorithmus von Bellman und Ford

## Theorem

*Gegeben sei ein gerichteter Graph  $G = (V, E)$  mit Kantengewichten  $E \rightarrow \mathbf{R}$  und ein Knoten  $s \in V$ .*

*Wir können in  $O(|V| \cdot |E|)$  feststellen, ob ein Kreis mit negativem Gewicht existiert, und falls nicht, die kürzesten Wege von  $s$  zu allen Knoten berechnen.*

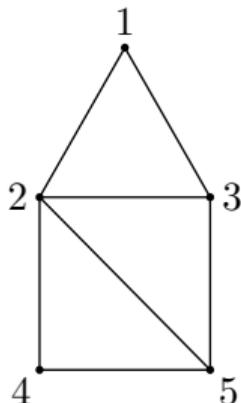
## Beweis.

Wir haben die Korrektheit bereits nachgewiesen.

Zur Laufzeit: Jeder Knoten wird  $|V|$  mal relaxiert, also wird auch jede Kante  $|V|$  mal relaxiert (in konstanter Zeit). □

# Rekapitulation: Darstellung von Graphen

## Adjazenzliste



1		2, 3
2		1, 3, 4, 5
3		1, 2, 5
4		2, 5
5		2, 3, 4

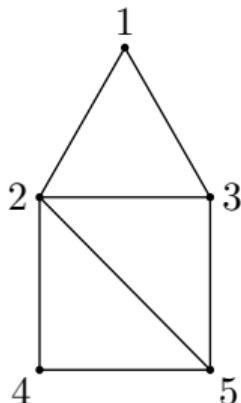
Alle bisherigen Algorithmen:

Adjazenzliste gute Darstellung

Kritische Operation: Alle ausgehenden Kanten besuchen.

# Rekapitulation: Darstellung von Graphen

## Adjazenzliste



1		2, 3
2		1, 3, 4, 5
3		1, 2, 5
4		2, 5
5		2, 3, 4

Alle bisherigen Algorithmen:

**Adjazenzliste gute Darstellung**

Kritische Operation: Alle ausgehenden Kanten besuchen.

# All Pairs Shortest Paths

Eingabe: Gerichteter Graph mit Kantengewichten

Ausgabe: Abstände und kürzeste Wege zwischen allen Knotenpaaren

Laufzeit  $O((|V|^2 + |V| \cdot |E|) \log |V|)$  mit Dijkstra.

→ Wende Dijkstra auf jeden Knoten an.

# All Pairs Shortest Paths

Eingabe: Gerichteter Graph mit Kantengewichten

Ausgabe: Abstände und kürzeste Wege zwischen allen Knotenpaaren

Laufzeit  $O((|V|^2 + |V| \cdot |E|) \log |V|)$  mit Dijkstra.

→ Wende Dijkstra auf jeden Knoten an.

# Algorithmus von Floyd

## Algorithmus

```
procedure Floyd1 :  
for  $i = 1, \dots, n$  do  
  for  $j = 1, \dots, n$  do  $d[i, j, 0] := \text{length}[i, j]$  od  
od;  
for  $k = 1, \dots, n$  do  
  for  $i = 1, \dots, n$  do  
    for  $j = 1, \dots, n$  do  
       $d[i, j, k] := \min \{ d[i, j, k - 1], d[i, k, k - 1] + d[k, j, k - 1] \}$   
    od  
  od  
od
```

Der Abstand von  $i$  nach  $j$  ist in  $d[i, j, n]$  zu finden.

## Theorem

*Die kürzesten Wege zwischen allen Knotenpaaren eines gerichteten Graphen  $G = (V, E)$  können in  $O(|V|^3)$  Schritten gefunden werden.*

## Beweis.

Per Induktion:  $d[i, j, k]$  enthält die Länge des kürzesten Pfades von  $i$  nach  $j$ , wenn nur  $1, \dots, k$  als Zwischenstationen erlaubt sind.

Dann enthält  $d[i, j, n]$  den wirklichen Abstand. □

## Theorem

*Die kürzesten Wege zwischen allen Knotenpaaren eines gerichteten Graphen  $G = (V, E)$  können in  $O(|V|^3)$  Schritten gefunden werden.*

## Beweis.

Per Induktion:  $d[i, j, k]$  enthält die Länge des kürzesten Pfades von  $i$  nach  $j$ , wenn nur  $1, \dots, k$  als Zwischenstationen erlaubt sind.

Dann enthält  $d[i, j, n]$  den wirklichen Abstand. □

# Algorithmus von Floyd

Einfachere Version:

## Algorithmus

```
procedure Floyd :  
for i = 1, ..., n do  
  for j = 1, ..., n do d[i, j] := length[i, j] od  
od;  
for k = 1, ..., n do  
  for i = 1, ..., n do  
    for j = 1, ..., n do  
      d[i, j] := min { d[i, j], d[i, k] + d[k, j] }  
    od  
  od  
od
```

## Spezialfall: Transitive Hülle – Algorithmus von Warshall

Frage: Zwischen welchen Knotenpaaren gibt es einen Weg?

### Algorithmus

```
procedure Warshall :  
for  $i = 1, \dots, n$  do  
  for  $j = 1, \dots, n$  do  $D[i, j] := A[i, j]$  od  
od;  
for  $k = 1, \dots, n$  do  
  for  $i = 1, \dots, n$  do  
    for  $j = 1, \dots, n$  do  
       $D[i, j] := D[i, j] \vee D[i, k] \wedge D[k, j]$   
    od  
  od  
od
```

# Transitive Hülle

## Theorem

*Wir können die transitive Hülle eines gerichteten Graphen  $G = (V, E)$ , der  $k$  starke Zusammenhangskomponenten hat, in  $O(|V| + |E'| + k^3)$  Schritten berechnen, wobei  $E'$  die Kanten der transitiven Hülle sind.*

## Beweis.

- 1 Berechne die starken Zusammenhangskomponenten
- 2 Schrumpfe jede SCC  $X$  zu einem Knoten
- 3 Berechne die transitive Hülle  $H$  dieses Graphen
- 4 Für jede Kante  $(X, Y)$  in  $H$  gib alle Kanten  $(x, y)$  mit  $x \in X, y \in Y$  aus.



# Transitive Hülle

## Theorem

*Wir können die transitive Hülle eines gerichteten Graphen  $G = (V, E)$ , der  $k$  starke Zusammenhangskomponenten hat, in  $O(|V| + |E'| + k^3)$  Schritten berechnen, wobei  $E'$  die Kanten der transitiven Hülle sind.*

## Beweis.

- 1 Berechne die starken Zusammenhangskomponenten
- 2 Schrumpfe jede SCC  $X$  zu einem Knoten
- 3 Berechne die transitive Hülle  $H$  dieses Graphen
- 4 Für jede Kante  $(X, Y)$  in  $H$  gib alle Kanten  $(x, y)$  mit  $x \in X, y \in Y$  aus.



# Transitive Hülle

## Theorem

*Wir können die transitive Hülle eines gerichteten Graphen  $G = (V, E)$ , der  $k$  starke Zusammenhangskomponenten hat, in  $O(|V| + |E'| + k^3)$  Schritten berechnen, wobei  $E'$  die Kanten der transitiven Hülle sind.*

## Beweis.

- 1 Berechne die starken Zusammenhangskomponenten
- 2 Schrumpfe jede SCC  $X$  zu einem Knoten
- 3 Berechne die transitive Hülle  $H$  dieses Graphen
- 4 Für jede Kante  $(X, Y)$  in  $H$  gib alle Kanten  $(x, y)$  mit  $x \in X, y \in Y$  aus.



# Transitive Hülle

## Theorem

*Wir können die transitive Hülle eines gerichteten Graphen  $G = (V, E)$ , der  $k$  starke Zusammenhangskomponenten hat, in  $O(|V| + |E'| + k^3)$  Schritten berechnen, wobei  $E'$  die Kanten der transitiven Hülle sind.*

## Beweis.

- 1 Berechne die starken Zusammenhangskomponenten
- 2 Schrumpfe jede SCC  $X$  zu einem Knoten
- 3 Berechne die transitive Hülle  $H$  dieses Graphen
- 4 Für jede Kante  $(X, Y)$  in  $H$  gib alle Kanten  $(x, y)$  mit  $x \in X, y \in Y$  aus.



# Transitive Hülle

## Theorem

*Wir können die transitive Hülle eines gerichteten Graphen  $G = (V, E)$ , der  $k$  starke Zusammenhangskomponenten hat, in  $O(|V| + |E'| + k^3)$  Schritten berechnen, wobei  $E'$  die Kanten der transitiven Hülle sind.*

## Beweis.

- 1 Berechne die starken Zusammenhangskomponenten
- 2 Schrumpfe jede SCC  $X$  zu einem Knoten
- 3 Berechne die transitive Hülle  $H$  dieses Graphen
- 4 Für jede Kante  $(X, Y)$  in  $H$  gib alle Kanten  $(x, y)$  mit  $x \in X, y \in Y$  aus.

