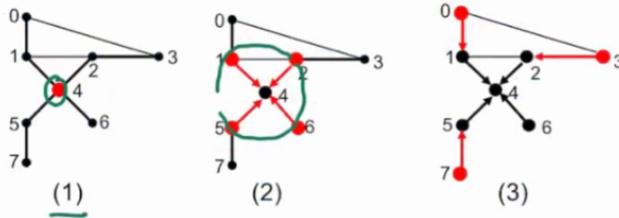


Kapitel 8

Erweiterte Breitensuche

- **Single-source shortest-path:** Gesucht sind alle kürzesten Wege von einem Startknoten s zu allen anderen Knoten.
- Beginne bei s und durchlaufe Graph mit **Breitensuche**.
- Speichere für jeden besuchten Knoten v die Distanz d[v] zu Startknoten s.
- Speichere Vorgängerknoten p[v] im kürzesten Weg nach v.
p stellt ein Baum mit Wurzel s dar.
- Der kürzeste Weg von s nach v ergibt sich in umgekehrter Reihenfolge:
 $v, p[v], p[p[v]], \dots, p[\dots p[p[v]]\dots] = s$

Beispiel: Startknoten s = 4



v	0	1	2	3	4	5	6	7
d[v]	∞	∞	∞	∞	0	∞	∞	∞
p[v]	-	-	-	-	-	-	-	-

v	0	1	2	3	4	5	6	7
d[v]	∞	1	1	∞	0	1	1	∞
p[v]	-	4	4	-	4	4	4	-

v	0	1	2	3	4	5	6	7
d[v]	2	1	1	2	0	1	1	2
p[v]	1	4	4	2	-	4	4	5

- In d[v] Reihe sind Anzahl Kanten zum Knoten 4
 → In p[v] sind Vorgänger vom Knoten. 4 hat kein Vorgänger!

Analyse der Breitensuche

- Für jeden Knoten w kann $d[w]$ höchstens einmal von ∞ auf einen endlichen Wert gesetzt werden. Damit kommen in die Queue q höchstens $|V|$ Knoten. Die while-Schleife läuft damit höchstens $|V|$ -mal.
- Jede Kantenrichtung wird höchstens einmal in einer der for-Schleifen betrachtet. Verwendet man die Adjazenzlistendarstellung, ist der Aufwand für den Durchlauf aller for-Schleifen gleich $O(|E|)$.
- Damit ergibt sich mit Adjazenzlistendarstellung insgesamt:

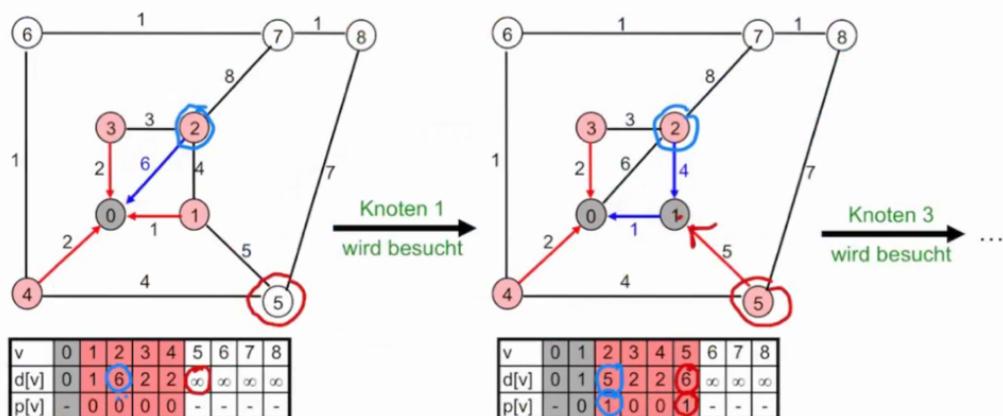
$$T = O(|V| + |E|)$$

Fall Graph dünn : $|E| = O(|V|)$
 $\Rightarrow T = (|V|)$

Fall Graph dicht : $|E| = O(|V|^2)$
 $\Rightarrow T = (|V|^2)$

Algorithmus von Dijkstra – Idee (2)

- Wird ein neuer Knoten v besucht, dann kommen eventuell neue Nachbarknoten zur Kandidatenliste.
- Bei allen Nachbarknoten wird geprüft, ob ein Weg über v einen **kürzeren Weg** ergibt.

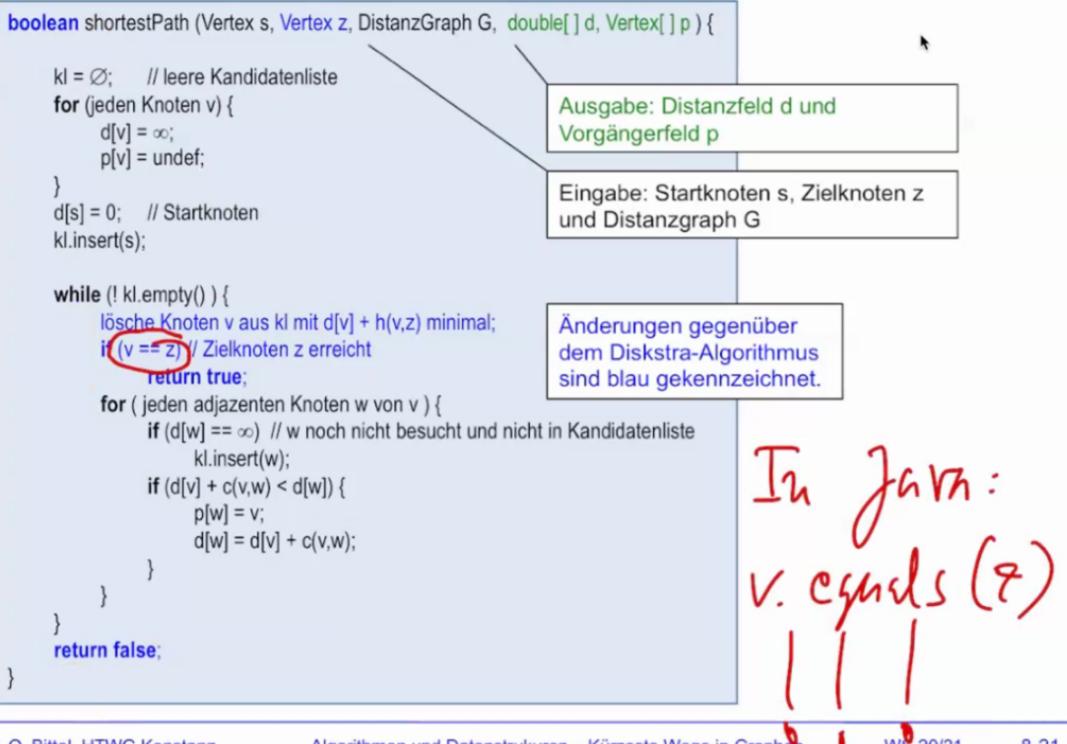


Knoten 5 ist zur Kandidatenliste neu dazu gekommen.

Der Distanzwert von Knoten 2 hat sich von 6 auf 5 verbessert, da der Weg über Knoten 1 kürzer ist als der bisherige.

- Neuer Nachbar, 5
- Kürzerer Weg gefunden zum bereits Nachbar 2

A*-Algorithmus

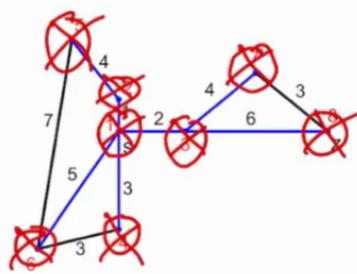


Prof. Dr. O. Bittel, HTWG Konstanz Algorithmen und Datenstrukturen – Kürzeste Wege in Graphen WS 20/21 8-21

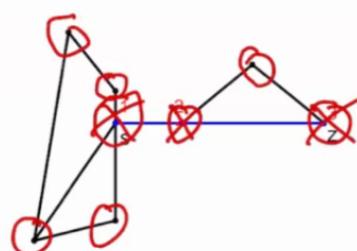
- Rot, kann Stunden dauern um Fehler zu finden. Das ist PSEUDO code, man muss überlegen wie man das umsetzt.

Vergleich A*- und Dijkstra-Algorithmus

Dijkstra-Algorithmus



A*-Algorithmus



O Kandidat
X besucht

- Schätzfunktion h und Kantengewicht c sind als Euklidischer Abstand definiert.
- Im Beispiel wird kürzester Weg von Startknoten s nach Zielknoten z berechnet.
- Reihenfolge der besuchten Knoten in rot.
- Berechnete kürzeste Wege sind blau markiert.

A*-Algorithmus

```

boolean shortestPath (Vertex s, Vertex z, DistanzGraph G, double[ ] d, Vertex[ ] p) {
    kl = Ø; // leere Kandidatenliste
    for (jeden Knoten v) {
        d[v] = ∞;
        p[v] = undef;
    }
    d[s] = 0; // Startknoten
    kl.insert(s);
    while (! kl.empty() ) {
        lösche Knoten v aus kl mit d[v] + h(v,z) minimal;
        if (v == z) // Zielknoten z erreicht
            return true;
        for ( jeden adjazenten Knoten w von v ) {
            if (d[w] == ∞) // w noch nicht besucht und nicht in Kandidatenliste
                kl.insert(w);
            if (d[v] + c(v,w) < d[w] ) {
                p[w] = v;
                d[w] = d[v] + c(v,w);
            }
        }
    }
    return false;
}

```

Ausgabe: Distanzfeld d und Vorgängerfeld p

Eingabe: Startknoten s, Zielknoten z und Distanzgraph G

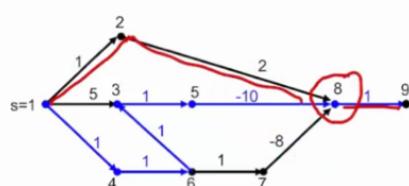
Änderungen gegenüber dem Dijkstra-Algorithmus sind blau gekennzeichnet.

h = 0

Dijkstror
Algo.

In Java:
v. equals (?)
|||

Beispiel zu Moore-Ford-Algorithmus



Nächster besuchter Knoten ist rot gekennzeichnet.

v	1	2	3	4	5	6	7	8	9
d[v]	0	∞	∞	∞	∞	∞	∞	∞	∞
p[v]	-	-	-	-	-	-	-	-	-
	1	5	1						

v	1	2	3	4	5	6	7	8	9
p[v]	-	-	-	-	-	-	-	-	-
	1	1	1						

Kandidatenliste kl									
1									
2, 3, 4									
3, 4, 8									
4, 8, 5									
8, 5, 6									
5, 6, 9									
6, 9, 8									
9, 8, 3, 7									
3, 7, 9									
7, 9, 5									
9, 5, 8									
8									
9									

08_Graphen_KuerzesteWege

Unbenanntes Notizbuch

Beispiel zu Moore-Ford-Algorithmus

Nächster besuchter Knoten ist rot gekennzeichnet.

v	1	2	3	4	5	6	7	8	9
d[v]	0	∞							
p[v]	-	-	-	-	-	-	-	-	-
	1	5	1						
				3					
					6				
						2			
							4		
								2	
									8
									9

v	1	2	3	4	5	6	7	8	9
d[v]	0	∞							
p[v]	-	-	-	-	-	-	-	-	-
	1	1	1						
				3					
					4				
						2			
							8		
								5	
									6
									9

Kandidatenliste kl
1
2, 3, 4
3, 4, 8
4, 8, 5
8, 5, 6
5, 6, 9
6, 9, 8
8, 3, 7
3, 7, 9
7, 9, 5
9, 5, 8
8
9

Prof. Dr. O. Bittel, HTWG Konstanz Algorithmen und Datenstrukturen – Kürzeste Wege in Graphen WS 20/21 8-30

08_Graphen_KuerzesteWege

Unbenanntes Notizbuch

Beispiel zu Moore-Ford-Algorithmus

Nächster besuchter Knoten ist rot gekennzeichnet.

v	1	2	3	4	5	6	7	8	9
d[v]	0	∞							
p[v]	-	-	-	-	-	-	-	-	-
	1	5	1						
				3					
					6				
						2			
							4		
								2	
									8
									9

v	1	2	3	4	5	6	7	8	9
d[v]	0	∞							
p[v]	-	-	-	-	-	-	-	-	-
	1	1	1						
				3					
					4				
						2			
							8		
								5	
									6
									9

Kandidatenliste kl
1
2, 3, 4
3, 4, 8
4, 8, 5
8, 5, 6
5, 6, 9
6, 9, 8
8, 3, 7
3, 7, 9
7, 9, 5
9, 5, 8
8
9

Prof. Dr. O. Bittel, HTWG Konstanz Algorithmen und Datenstrukturen – Kürzeste Wege in Graphen WS 20/21 8-30

Algorithmus von Moore und Ford

