Kapitel 6: weitere algorithmische Vorgehensweisen

- Wiederholung Graphdefinition und Dijkstra-Algorithmus
- zweitkürzeste Wege in einem Graphen:
  Algorithmen von Pollack und Hoffman & Pavley
  (algorithmusverändernd)
- zweitkürzeste Wege in einem Graphen.
  Algorithmus von Azevedo (graphverändernd)
- Verallgemeinerung auf dritt und
  k-kürzeste Wege
- Ausblick: Wegeverbote

Graphentheoretische Definitionen

Ein (gerichteter) (directed) *Graph* $G := (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge $E \subseteq V \times V \times \mathrm{IN}$. Zwei Kanten (u_1,v_1,n_1) (u_2,v_2,n_2) heissen *Multikanten*, wenn $u_1 = u_2$ und $v_1 = v_2$. Ein Graph heißt *multikantenfrei*, wenn $E \subseteq V \times V$.

Im Folgenden sei G multikantenfrei.

Eine Kante heißt *bidirektional* (ungerichtet), wenn zu $(u,v) \in E$ immer auch $(v,u) \in$ E existiert; sonst heißt sie *unidirektional* (gerichtet).

Eine Kante der Form (u,u) heißt *Schlinge*. Ein n Tupel $(u_i,v_i) \in E$ $i = 1..n$ heißt *Weg* wenn $u_{i+1} = v_i$ für $i = 1$ .. n-1 gilt (Kantenfolge).

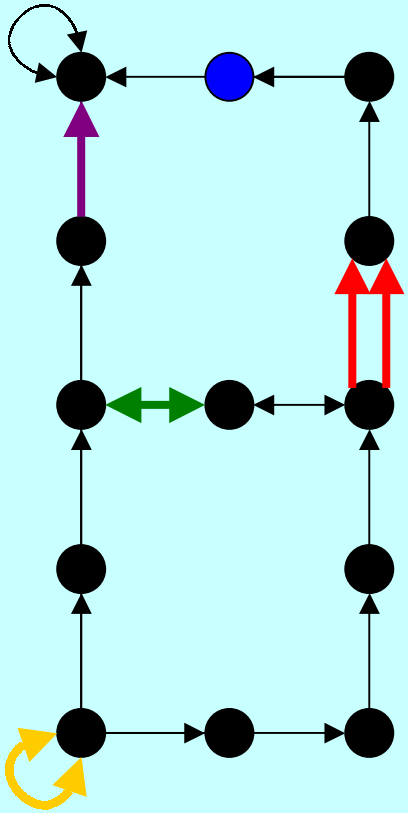
Ein Graph heißt (*kanten*)*gewichtet*, wenn eine Abbildung $\beta : E \rightarrow \mathrm{IR}$ (im Folgenden IR_0^+) definiert ist.

Die Abbildung β kann (intuitiv) auf Wege erweitert werden durch

$$\delta : \{\text{Wege}\} \rightarrow \mathrm{IR} \text{ mit } \delta((u_1,v_1),..,(u_n,v_n)) := \sum_{i=1}^{n} \beta(u_i,v_i).$$

$$d : V \times V \rightarrow \mathrm{IR} \text{ mit } d(u,v) := \min\{\delta((u,v_1),..,(u_n,v))\}.\ d(u,u) := 0$$

Ein Weg heißt *Zyklus* wenn $u_1 = v_n$



Graphentheoretische Definitionen

Ein Graph ist im Folgenden immer multikantenfrei und gerichtet.

Seine Knotenmenge heißt V, die Kantenmenge E.

Seine Kanten sind mit Gewichtsfunktion β gewichtet.

$\beta(u,v) \geq 0$ für alle $(u,v) \in E$

δ(Weg) gibt die Länge des Weges an = Summe der Kantengewichte

$d(u,v)$ = Länge des kürzesten Weges von u nach v.

Der Dijkstra Algorithmus (Greedy - Algorithmus)

Wir konstruieren schrittweise den kürzeste - Wegebaum von einem Startknoten s aus.

0.Schritt : $B_0 := \{s\}$. $d(s,s) := 0$.

Die Menge aller kürzesten Wege von einem Knoten aus hat Baumstruktur.

i : Schritt : Sei B_i der kürzeste - Wege - Baum nach dem i ten Schritt.

Im $i+1$ ten Schritt wird der Knoten $v_{i+1} \notin B_i$ in den B_{i+1} aufgenommen,

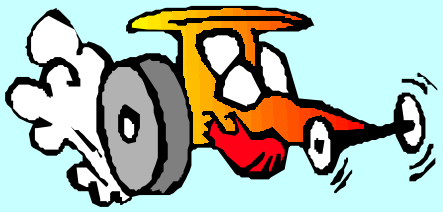
der zu s den $i+1$ kürzesten Abstand hat.

Die Menge $N_i := \{v \in V \mid \exists\ (u,v) \in E \text{ mit } u \in B_i\ ,\ v \notin B_i\}$ heißt *Nachbarschaftsliste.*

Wir suchen also den Knoten v_{i+1} der der folgenden Bedingung genügt :

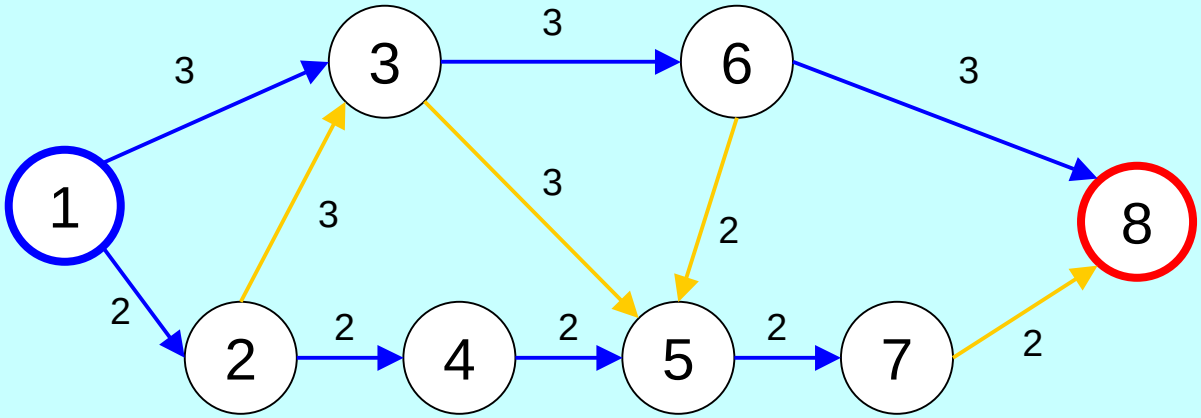
$$d(s,v_{i+1}) := \min\{d(s,u)+\beta(u,v) \mid (u,v) \in E, u \in B_i, v \in N_i\}$$

$$B_{i+1} := B_i \cup \{v_{i+1}\}$$

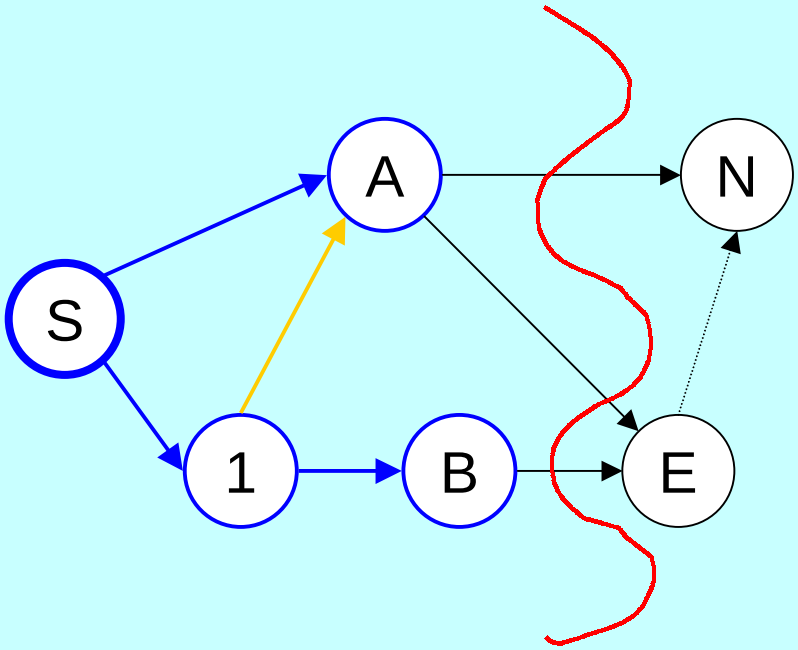


Beweis geht indirekt mit vollständiger Induktion. Der Aufwand des Algorithmus liegt bei $O((|V|+|E|)\cdot \log |V|)$ wenn die Nachbarschaftsliste als Heap organisiert wird.

Minimierung des Abstandes zur Wurzel



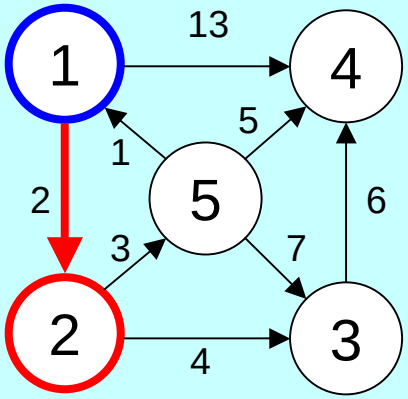
$$6+3=d\,(1,6)+\beta(6,8)<$$
$$d\,(1,7)+\beta(7,8)=8+2$$



$$d(\mathrm{S},\mathrm{B})+\beta(B,E)+\delta(\mathrm{E},\mathrm{N})<d(\mathrm{S},\mathrm{N})\ \text{(Annahme)}$$
$$=d(\mathrm{S},\mathrm{A})+\beta(A,N)$$
$$\le d(\mathrm{S},\mathrm{B})+\beta(B,E)$$
$$\text{(Algorithmus)}$$

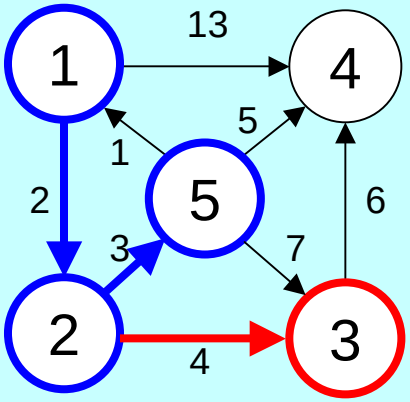
Widerspruch zu $\delta(\mathrm{A},\mathrm{B})\ge 0$

Beispiel für die Anwendung des Dijkstraalgorithmus



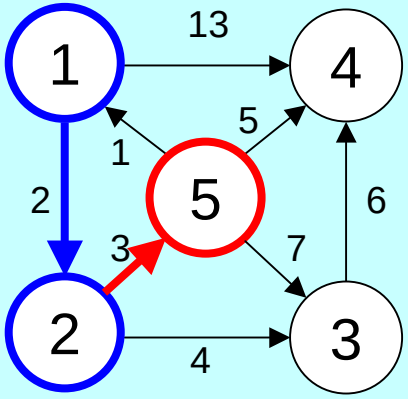
$$\beta(1,2) = 2$$
$$< \beta(1,4) = 13$$

d(1,2):=2



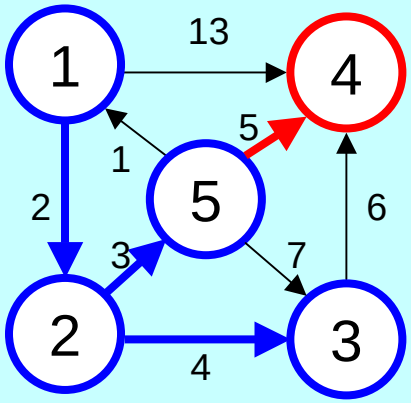
$$d(1,2) + \beta(2,3) = 6$$
$$< d(1,5) + \beta(5,4) = 10$$
$$< d(1,5) + \beta(5,3) = 12$$
$$< \beta(1,4) = 13$$

d(1,3):=6



$$d(1,2) + \beta(2,5) = 5$$
$$< d(1,2) + \beta(2,3) = 6$$
$$< \beta(1,4) = 13$$

d(1,5):=5



$$d(1,5) + \beta(5,4) = 10$$
$$< d(1,3) + \beta(3,4) = 12$$
$$< \beta(1,4) = 13$$

d(1,4):=10

Definition für k - kürzeste Wege

Gegeben sei ein *Graph* $G := (V,E)$ sowie ein Startknoten s und ein Zielknoten z.

Ein Weg w heißt *zyklenfrei* $:\Leftrightarrow$ kein Knoten kommt in w mehrfach vor.

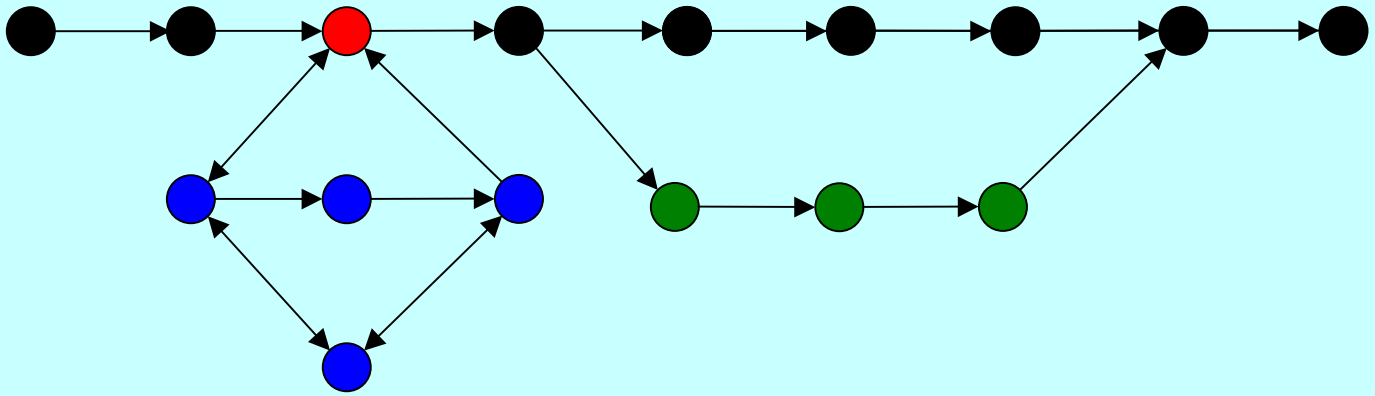
Sei W die Menge aller Wege von s nach z (dies können bis zu $|V-1|!$ zyklenfreie Wege sein).

Ein Weg w_1 heißt *kürzester Weg* von s nach $z :\Leftrightarrow \delta(w) \geq \delta(w_1)$ für alle $w \in W$

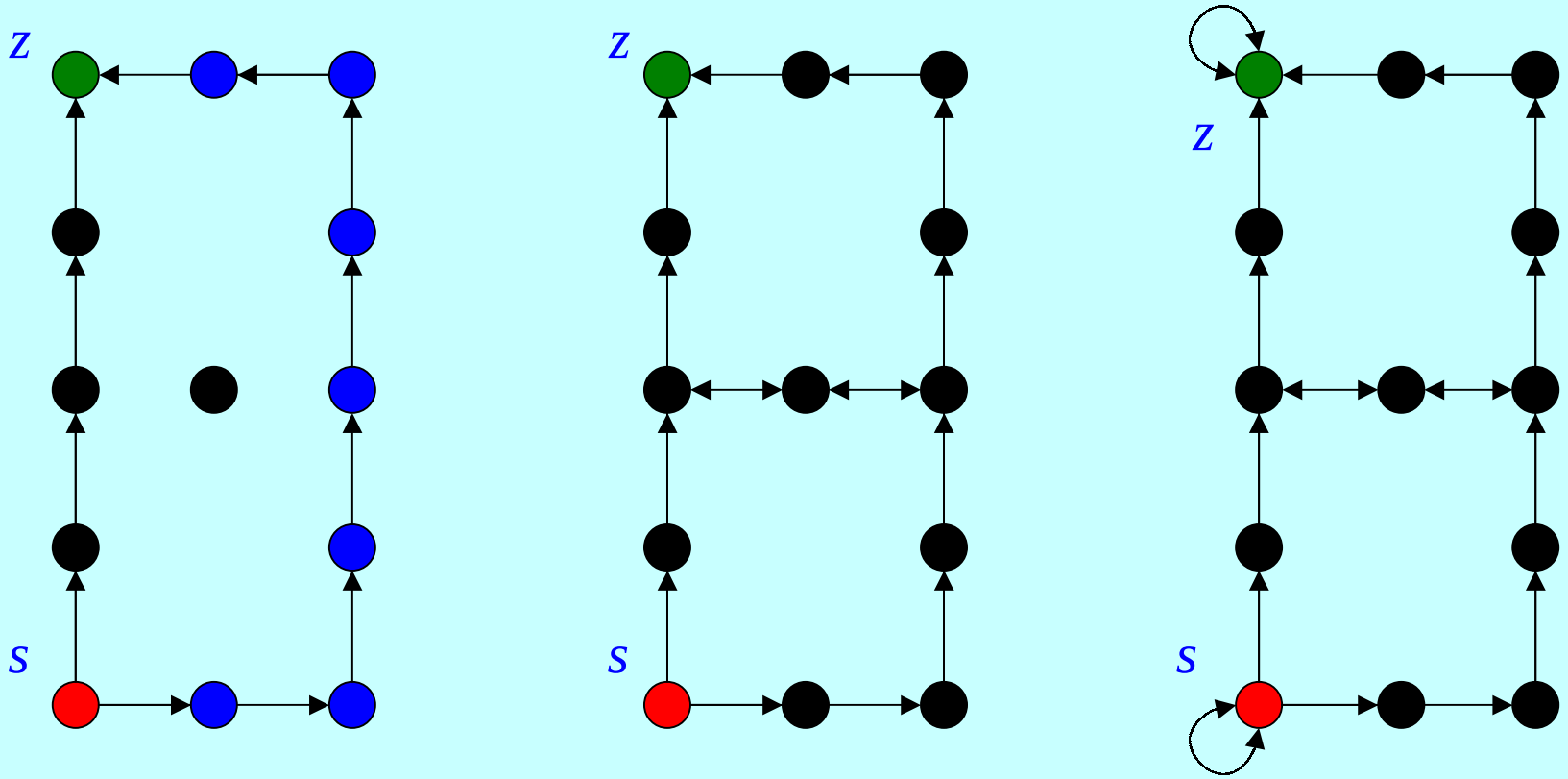
(dieser ist i.a. nicht eindeutig). Sei $w_1 \in$ W fest gewählt,

dann heißt w_2 *zweitkürzester Weg* von s nach $z :\Leftrightarrow \delta(w) \geq \delta(w_2)$ für alle $w \in W \setminus \{w_1\}$.

Ein Weg heißt *k - kürzester Weg* von s nach $z :\Leftrightarrow \delta(w) \geq \delta(w_k)$ für alle $w \in W \setminus \{w_1, .., w_{k-1}\}$.



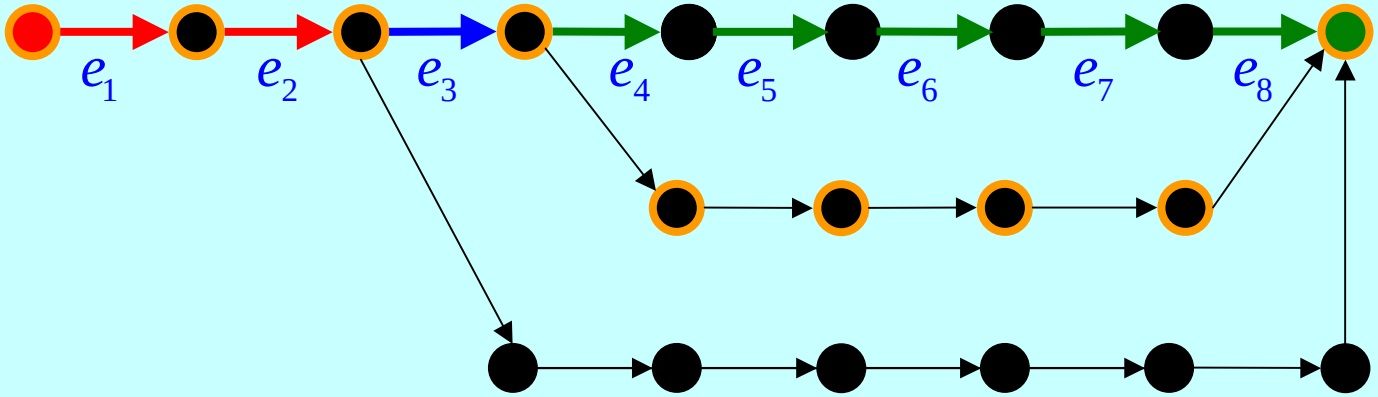
Zweitkürzeste Wege in einem Graphen - (einführende Beispiele)



- Gesucht ist der zweitkürzeste Weg von s nach z
- Dijkstra ist dem Problem zweit kürzester Wege nicht gewachsen
- Algorithmus oder Graphveränderung ist notwendig

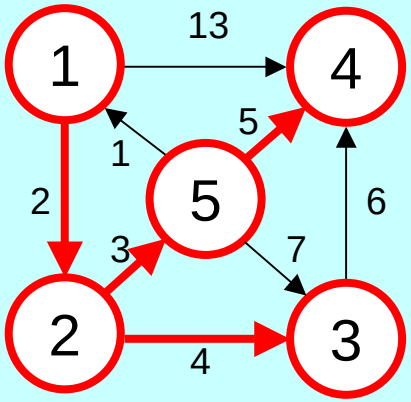
Algorithmus von Pollack

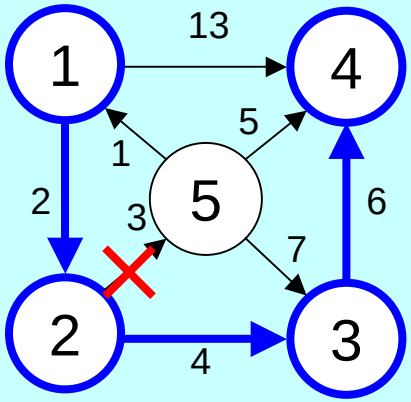
- Dynamisches algorithmusveränderndes Verfahren
- Berechne den kürzesten Weg zwischen Start und Zielknoten (m Kanten)
- Entferne jeweils eine Kante $(e_1,..,e_m)$ des kürzesten Weges und berechne in dem neu entstandenen Graph wieder den kürzesten Weg w_2^i zwischen Start und Zielknoten (m Berechnungen)
- Der kürzeste Weg dieser m Berechnungen ist der zweitkürzeste Weg zwischen Start und Zielknoten



- Kante 1 oder 2 wird weggelassen – kein kürzester Weg wird berechnet
- Kante 3 wird weggelassen – drittkürzester Weg wird berechnet
- Kante 4 – 8 wird weggelassen – zweitkürzester Weg wird berechnet

Der zweitkürzeste Weg von 1 nach 4

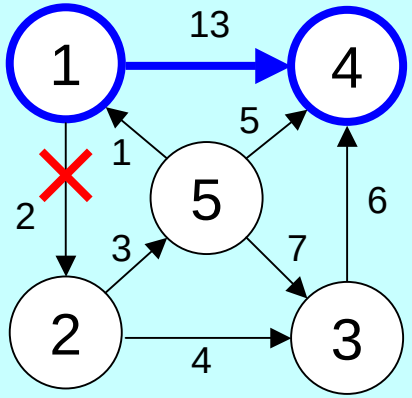




$e_2 = (2,5)$

$w_2^2 = (1,2),(2,3),(3,4)$

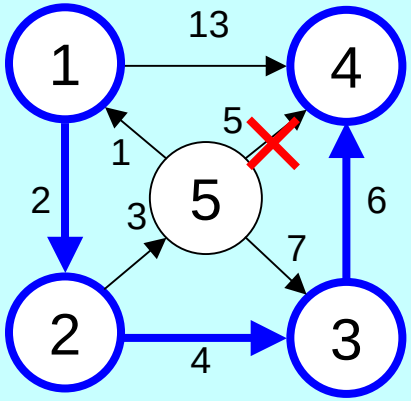
$\delta(w_2^2) = 12$



$e_1 = (1,2)$

$w_2^1 = (1,4)$

$\delta(w_2^1) = 13$



$e_3 = (5,4)$

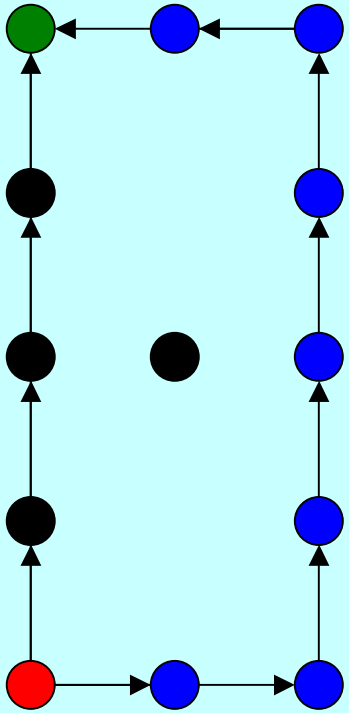
$w_2^3 = (1,2),(2,3),(3,4)$

$\delta(w_2^3) = 12$

Damit ist der zweitkürzeste Weg $w_2 = w_2^2 = w_2^3$

mit $\delta(w_2) = 12 > 10 =$ Länge des kürzesten Weges

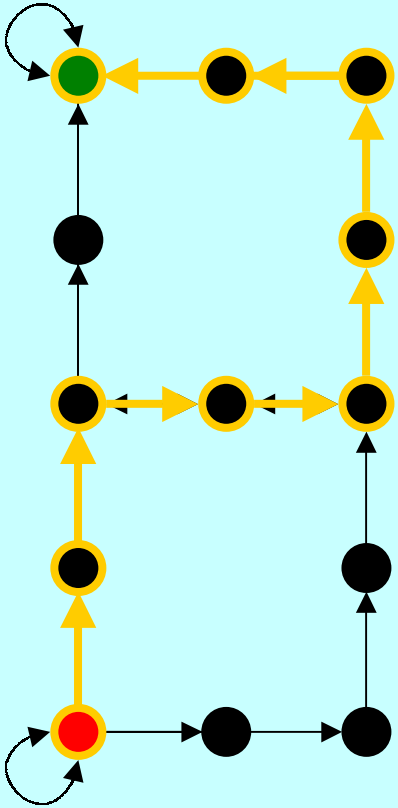
Der Algorithmus von Pollack findet keine Schleifen



$w_2^1 = w_2^2 = w_2^3 = w_2^4$

Der Algorithmus berechnet den orangenen Weg als zweitkürzesten Weg (als w_2^3 und w_2^4). Dies ist aber nur der zweitkürzeste schleifenfreie Weg.

Weder zweitkürzeste Weg mit Start oder Endschleife noch der Weg mit innerer Schleife werden gefunden.

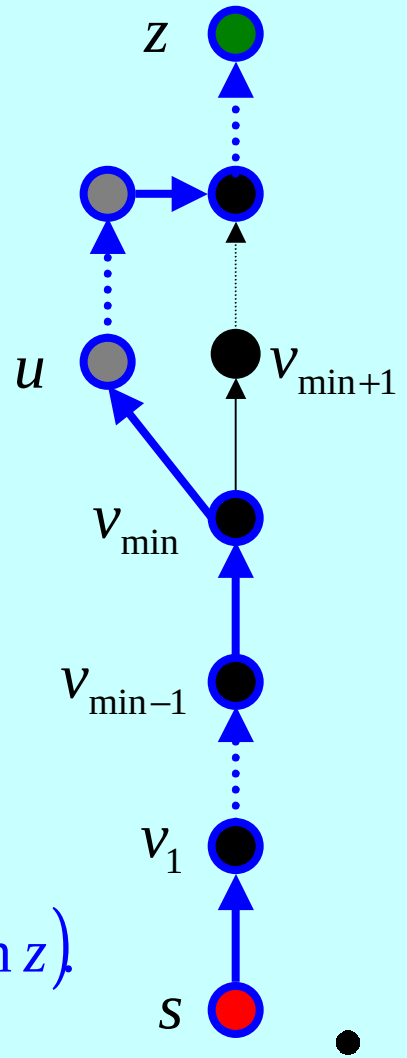


Sei m_1 $(< |V| - 1)$ die Anzahl der Kanten des kürzesten Weges, so liegt der Aufwand für den zweitkürzsten Weg bei $m_1 \cdot \text{Dijkstra} = O(m_1 \cdot ((|V| + |E|) \log |V|)) \leq O(|V| \cdot ((|V| + |E|) \cdot \log |V|))$.

Der Algorithmus von Hoffman & Pavley

Der zweitkürzeste Weg wird als Umweg des kürzesten Weges gesehen.

Berechne alle kürzesten Wege zum Zielknoten hin. Der kürzeste Weg $w := ((s, v_1), (v_1, v_2), ..., (v_n, z))$ von $v_0 := s$ nach $v_{n+1} := z$ wird damit ebenfalls berechnet.



Nimm von jedem Knoten v_i von w jede Kante (v_i, u) mit

(*) $u \neq v_{i+1}$ und berechneden zweitkürzesten Weg als

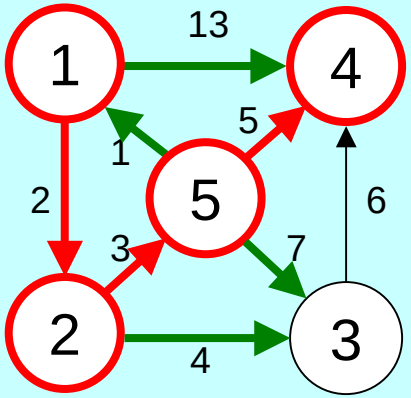
$$w_2 := \min_{i=0}^{n+1} \min_{u} \left\{ d(s, v_i) + \beta((v_i, u)) + d(u, z) \,\middle|\, u \neq v_{i+1} \right\}$$

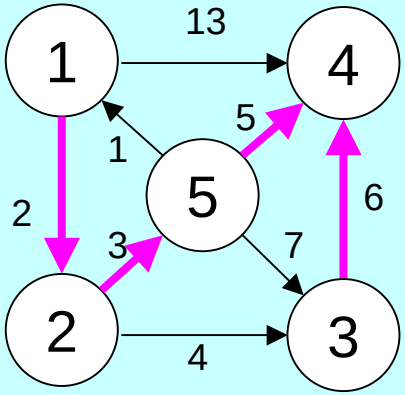
Der zweitkürzeste Weg ist also

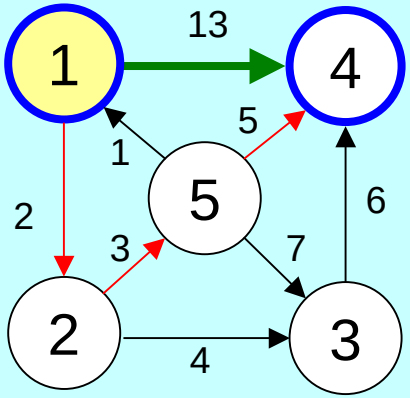
$w_2 := ((s, v_1), ..., (v_{min-1}, v_{min}), ..., (v_{min}, u),$ kürzester Weg von u nach $z)$.

Dieser Algorithmus findet auch Wege mit Zyklen

Der zweitkürzeste Weg von 1 nach 4

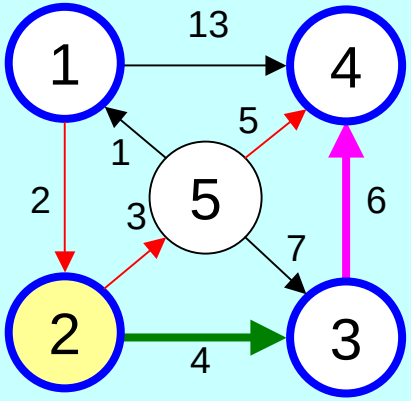


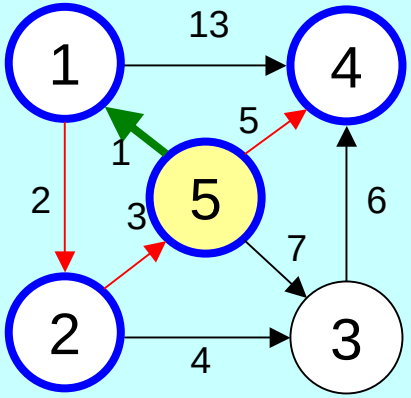


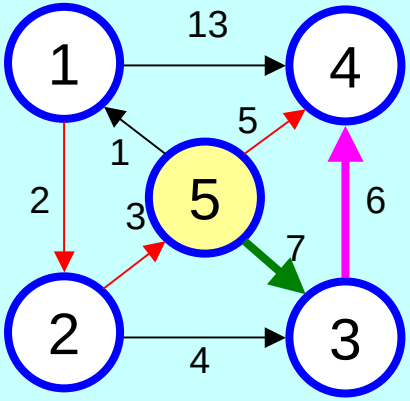


$w_2^1 = 1,4$

$\delta(w_2^1) = 13$







$w_2^4 = 1,2,5,3,4$

$\delta(w_2^4) = 18$

$w_2^2 = 1,2,3,4$

$\delta(w_2^2) = 12$

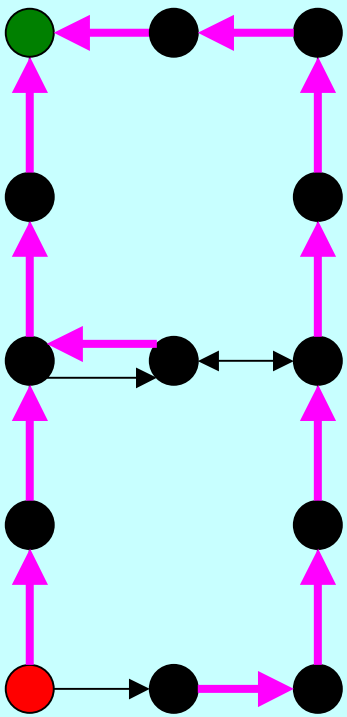
$w_2^2 = 1,2,5,1,2,5,4$

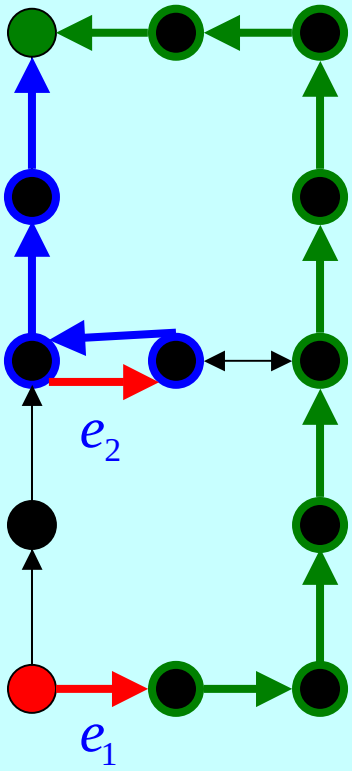
$\delta(w_2^3) = 16$

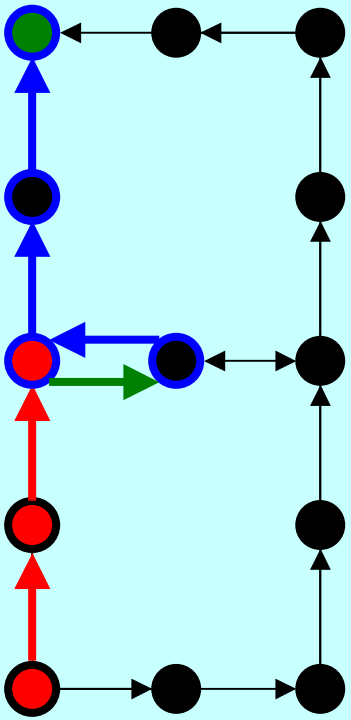
$w_2 = w_2^2 < w_2^1 < w_2^3 < w_2^4$

mit $\delta(w_2) = 12 > 10$

Schleifenbeispiel für den Algorithmus von Hoffman & Pavley

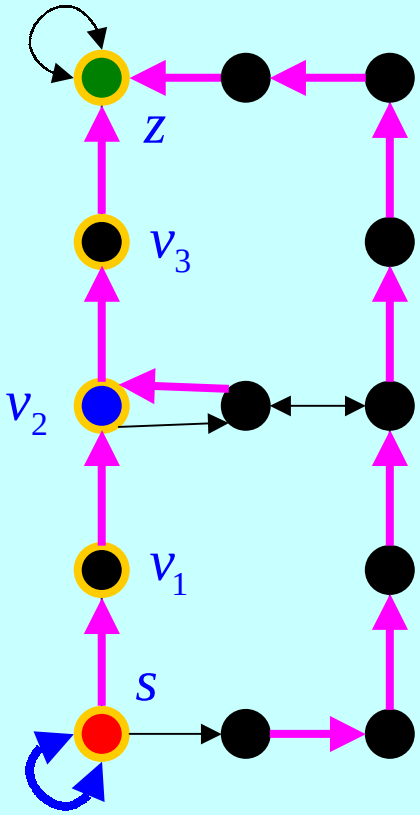






1. Die Kanten des kürzeste Wege Baumes zum Zielknoten hin sind violett
2. Die kürzesten Wege von den Endknoten der Kanten 1 und 2 werden betrachtet
3. Algorithmus erschafft zweitkürzesten Weg mit einem Zyklus

Schlingen am Anfang oder Ende und Aufwand



Gesucht ist wieder der zweitkürzeste Weg zwischen s und z.

Betrachte $s = v_0$. Die Kante (v_0, v_0) erfüllt die Eigenschaft $(*)$ $u \neq v_{i+1}$ nicht, denn $v_0 \neq v_1$. Damit wird der Weg

s, s, v_1, v_2, v_3, z

bei der Minimumbildung berücksichtigt und wird automatisch zweitkürzester Weg. $\delta(w_2) = 5$. Wenn man davon ausgeht, daß v_{n+2} (undefiniert) $\neq v_{n+1}$ ist, so wird auch der Weg

s, v_1, v_2, v_3, z, z

gefunden, der ebenfalls zweitkürzester Weg ist.

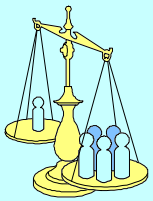
Jeder Knoten kann im kürzesten Weg vorkommen kann und alle Knoten können untereinander verbunden sein. Damit liegt der Aufwand des Verfahrens bei

$O(|V| \cdot |V|) + O(\text{Dijkstra}) = O(|V|^2)$, da $|E| \leq |V|^2$ $\quad O(|V|^2) < O(|V| \cdot ((|V| + |E|) \cdot \log |V|))$ (Pollack)

Verfahrensvergleich

Pollack

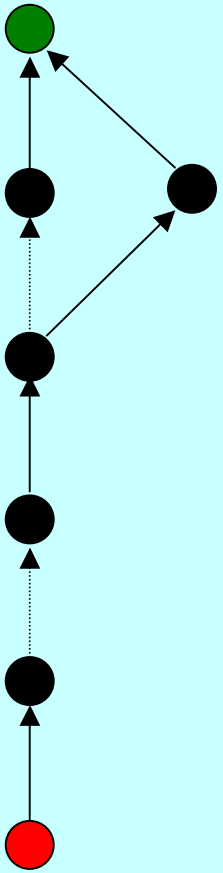
- Dynamisches Verfahren
- Berechne den kürzesten Weg w zwischen s und z
- Lasse eine Kante e von w weg.
- Berechne den kürzesten Weg im Graphen ohne die Kante e
- Minimiere über die so entstandenen kürzesten Wege
- Kürzester Weg ist gesperrt
- Nur schleifenfreie Wege werden gefunden



Hoffman & Pavley

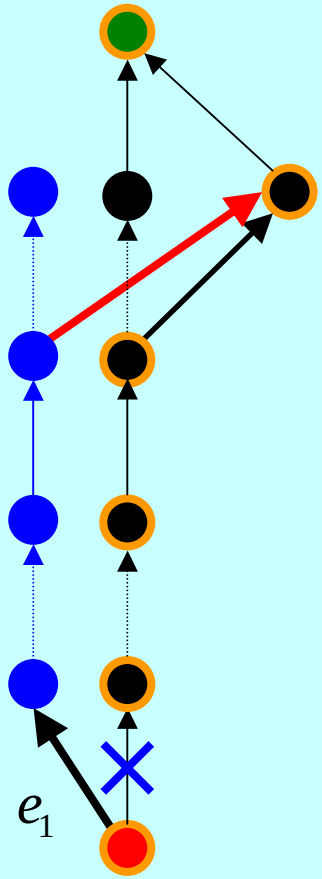
- Dynamisches Verfahren
- Berechne den kürzesten Weg w zwischen s und z
- Nimm jede Kante e deren Anfangsknoten ein Knoten von w ist und selbst keine Kante von w ist
- Berechne den kürzesten Weg vom Endknoten von e nach z und addiere den Weg nach s.
- Minimiere über die so entstandenen kürzesten Wege
- Zweitkürzester Weg wird als Variante des kürzesten Weges gesehen
- Auch Wege mit Schleifen werden gefunden

Algorithmus von Azevedo modifiziert von Schmid

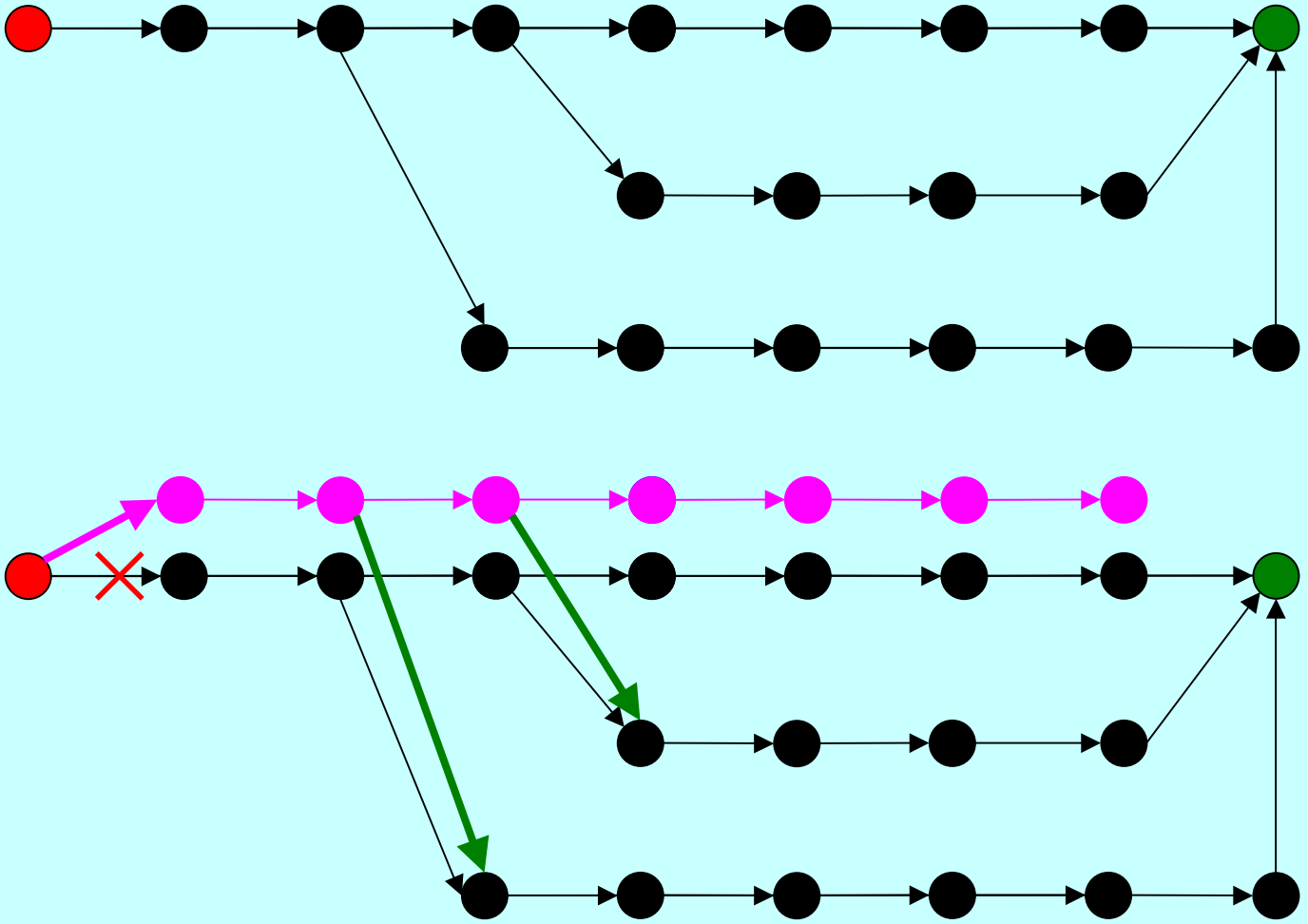


- Statisches Verfahren (verändert nur den Graph – nicht aber den kürzeste Wege- Algorithmus)
- Azevedo verwendet die Methode Pathdeletion (Wegeverbote).

1. Erschaffe einen neuen Startknoten und einen neuen Zielknoten (später).
2. Verdopple den kürzesten Weg ohne erste und letzte Kante bzw. Knoten. (Wegeverdopplung)
3. Weise der ersten Kante einen neuen Endknoten zu. (Umleitung)
4. Ziehe alle Kanten über die der kürzeste Weg verlassen werden kann. (Verbindung mit dem Ausgangsgraphen)

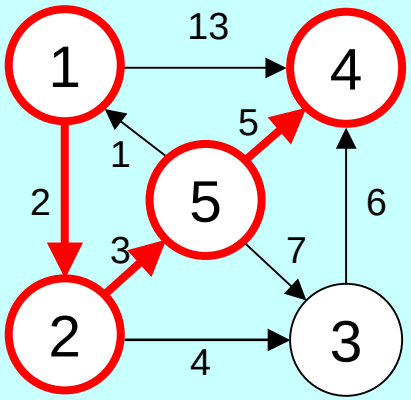


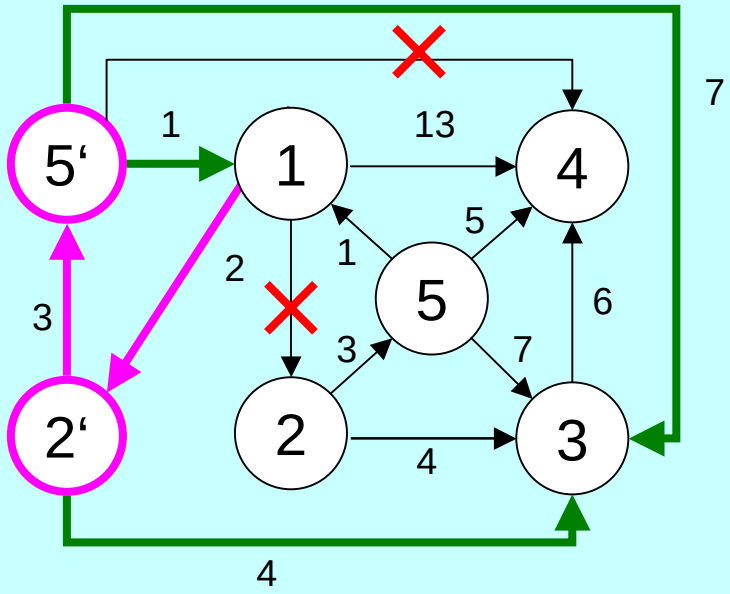
Beispiel für den modifizierten Algorithmus von Azevedo



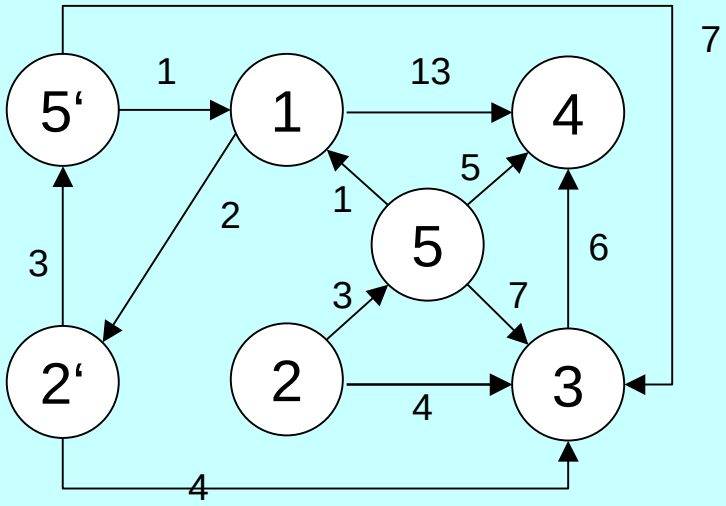
- Der kürzeste Weg ist nicht mehr begehbar. Der zweit und drittkürzeste Weg sind noch möglich.

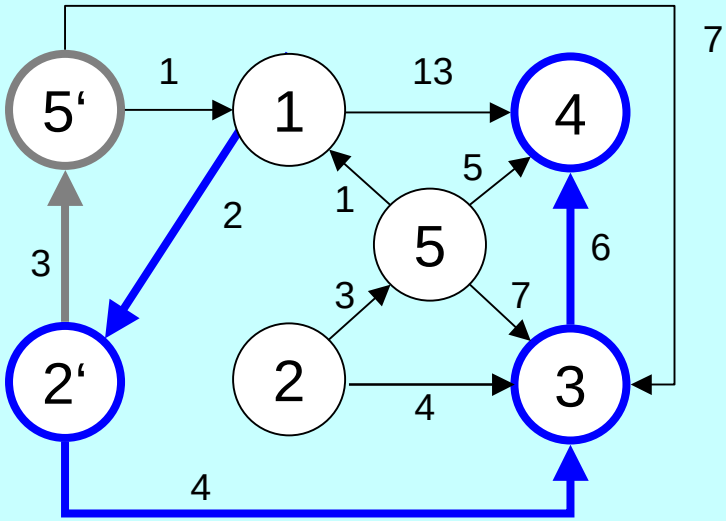
Der zweitkürzeste Weg von 1 nach 4



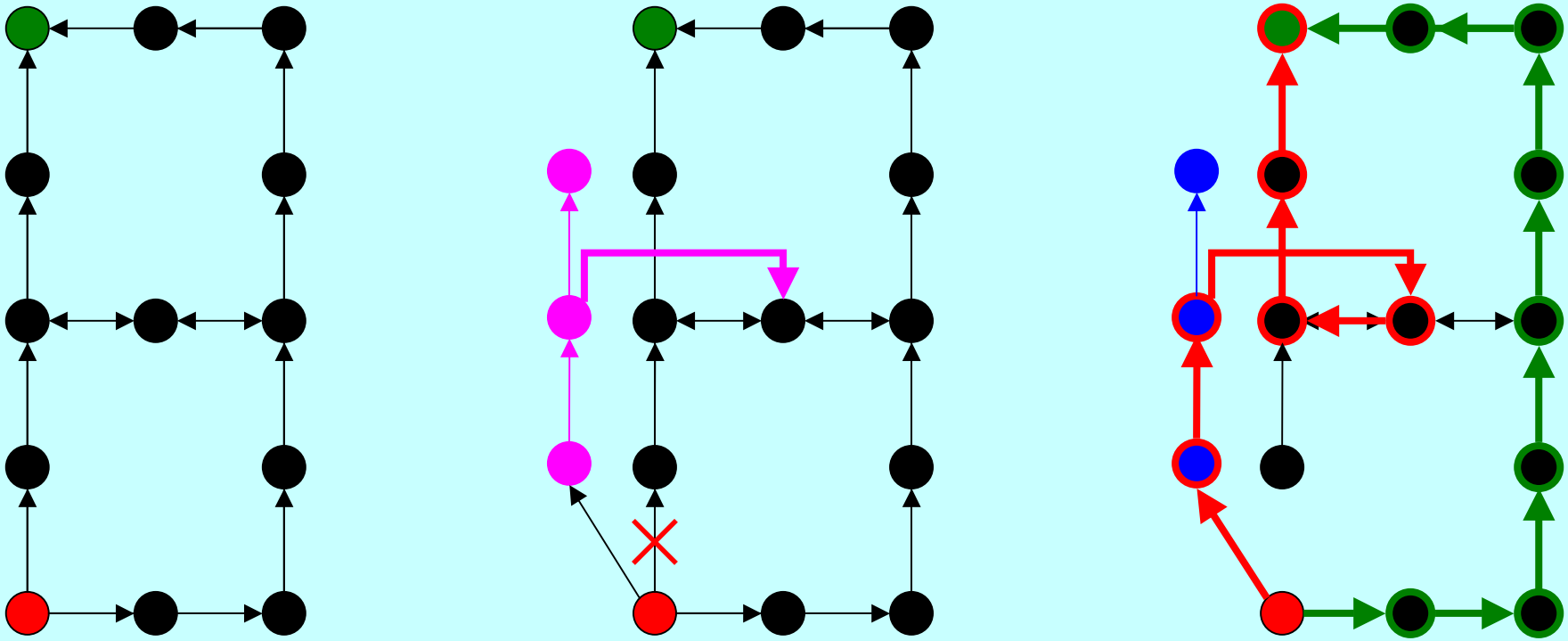


Der verdoppelte Weg ist blau
Die Verbindung mit dem Ausgangsgraphen grün.
Die Kanten (2',5) und (5',4) werden nicht gezogen. Die Kante (1,2) wir zur Kante (1,2').





Beispiel für zweitkürzesten Weg mit Schlinge

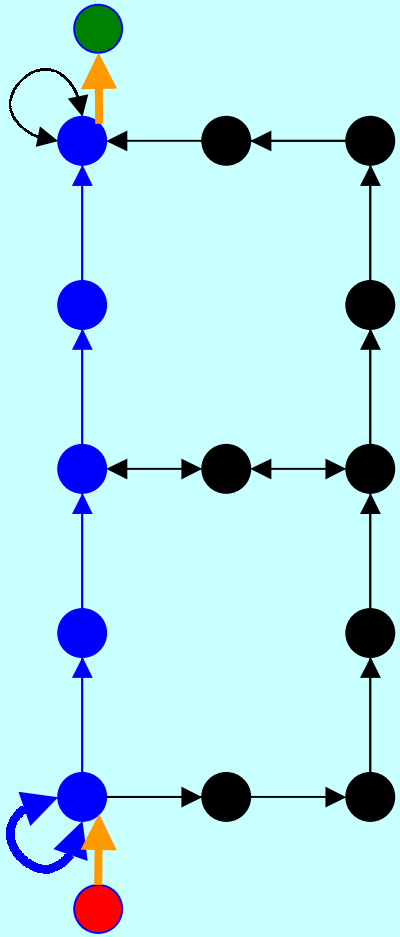


- Der zweitkürzeste Weg ist nicht der vermutete (grüne) Weg, sondern der rote Weg, der sich vom kürzesten Weg nur durch eine Schlinge unterscheidet.
- Länge des roten Weges = 6 < 8 = Länge des grünen Weges

Der erste Schritt

Problem:

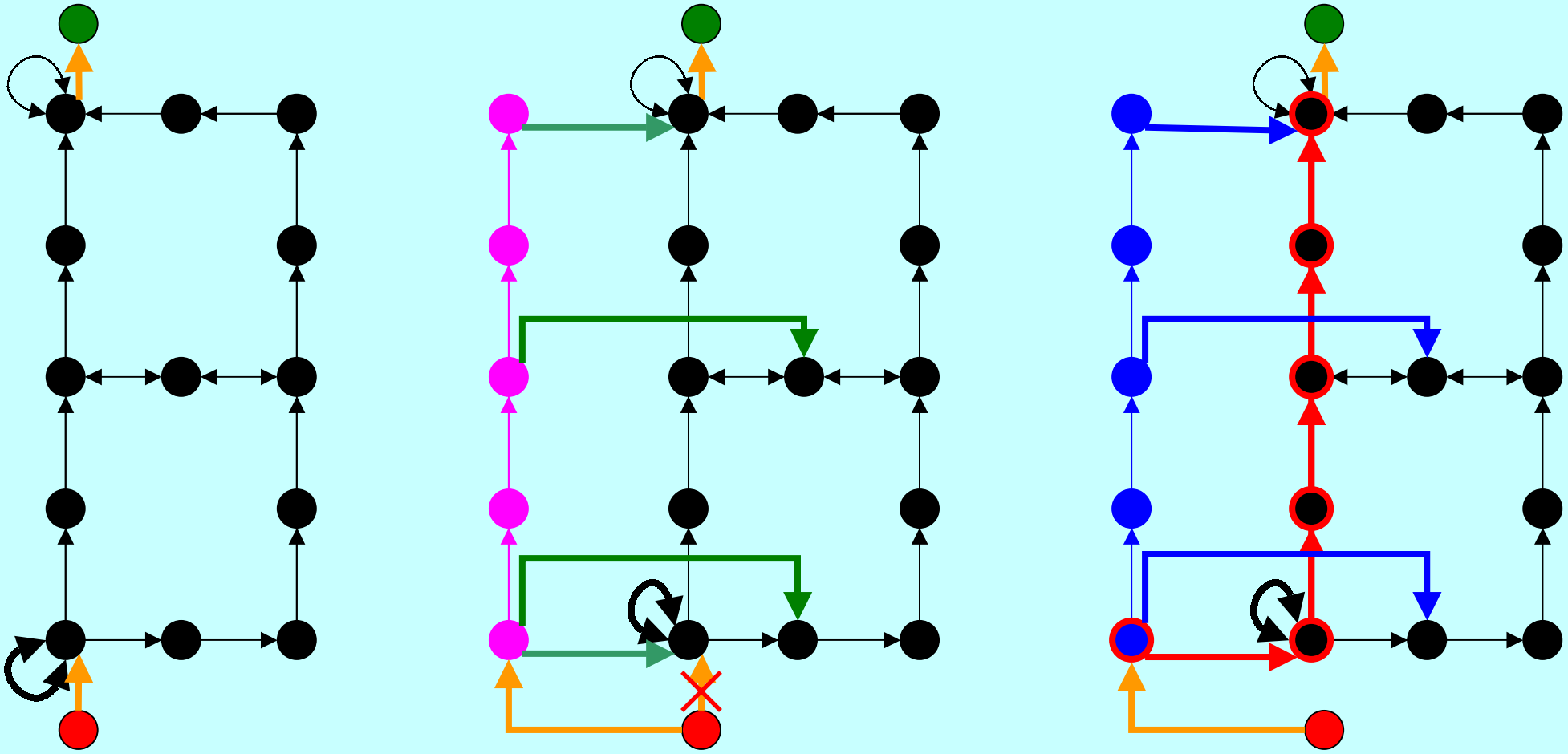
Der Algorithmus von Azevedo kann in dieser Form einen zweitkürzesten Weg der Form
Schlinge - (alter) kürzester Weg oder
(alter) kürzester Weg – Schlinge nicht erkennen.



Lösung:

- Erschaffe einen neuen Startknoten und einen neuen Zielknoten
- Verbinde den neuen Startknoten mit dem alten Startknoten durch eine Kante der Länge 0.
- Verbinde den alten Zielknoten mit dem neuen Zielknoten durch eine Kante der Länge 0.

Beispiel für einen erweiterten Graphen incl. 1 Schritt



Der zweitkürzeste Weg (rot) beinhaltet die Schlinge beim Startknoten

Aufwand des Verfahrens von Azevedo

Der kürzeste Weg zwischen 2 Knoten kann maximal $|V|$ Knoten und $|V|$-1 Kanten beinhalten. Die Anzahl, der vom kürzesten Weg wegweisenden Kanten ist maximal $|E|-2$. Damit ist die Anzahl der neu erschaffenen Elemente maximal $|V|$ Knoten und $|E|-2$ Kanten. Der Platzbedarf kann sich also fast verdoppeln.

Der Aufwand für die Grapherweiterung ist $O(|V|+|E|)$.

Der Aufwand, den zweit-kürzesten Weg zu berechnen, kann also bei Anwendung des Dijkstra-Algorithmus *zwei* mal so lange dauern wie die Berechnung des kürzesten Weges. Dies ist also $O((|V|+|E|)\cdot\log|V|)$ sofern $|E| \le \frac{|V^2|}{\log|V|}$.

Azevedo $= O((|V|+|E|)\cdot\log|V|) \le$ Hoffman $= O(|V|^2)$

$<$ Pollack $= O(|V|\,(|V|+|E|)\cdot\log|V|)$

Drittkürzeste Wege nach Pollack

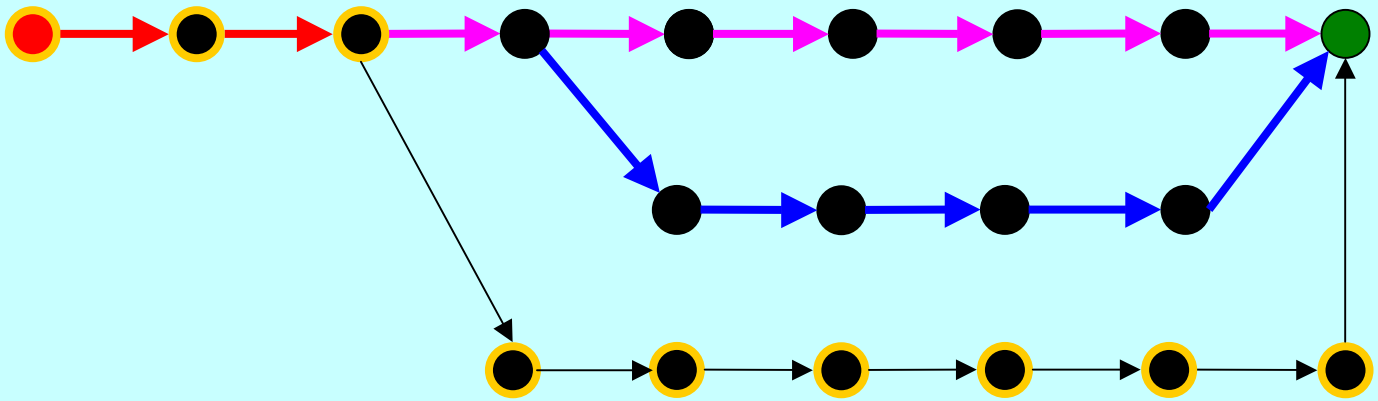
Berechnung des drittkürzesten Weges: Berechnung des drittkürzesten Weges:

Berechne den kürzesten Weg $w_1 = (e_1^1, \ldots e_{m_1}^1)$ und den zweitkürzesten Weg

$w_2 = (e_1^2, \ldots e_{m_2}^2)$ zwischen s und z. Sperre je ein Paar (e_i^1, e_j^2) aus $(w_1 \times w_2)$

berechne den kürzesten Weg $w_3^{i,j}$ und bilde dann das Minimum

$$w_3 = \min_{i,j}\{\delta(w_3^{i,j})\}$$



k kürzeste Wege nach Pollack

Berechnung des $k+1$ kürzesten Weges:

Berechne die k kürzesten Wege $w_1 = (e_1^1, \ldots e_{m_1}^1), \ldots, w_k = (e_1^k, \ldots e_{m_k}^k)$

zwischen s und z. Sperre je ein k Tupel $(e_{i_1}^1, \ldots e_{i_k}^k)$ aus $(w_1 \times .. \times w_k)$.

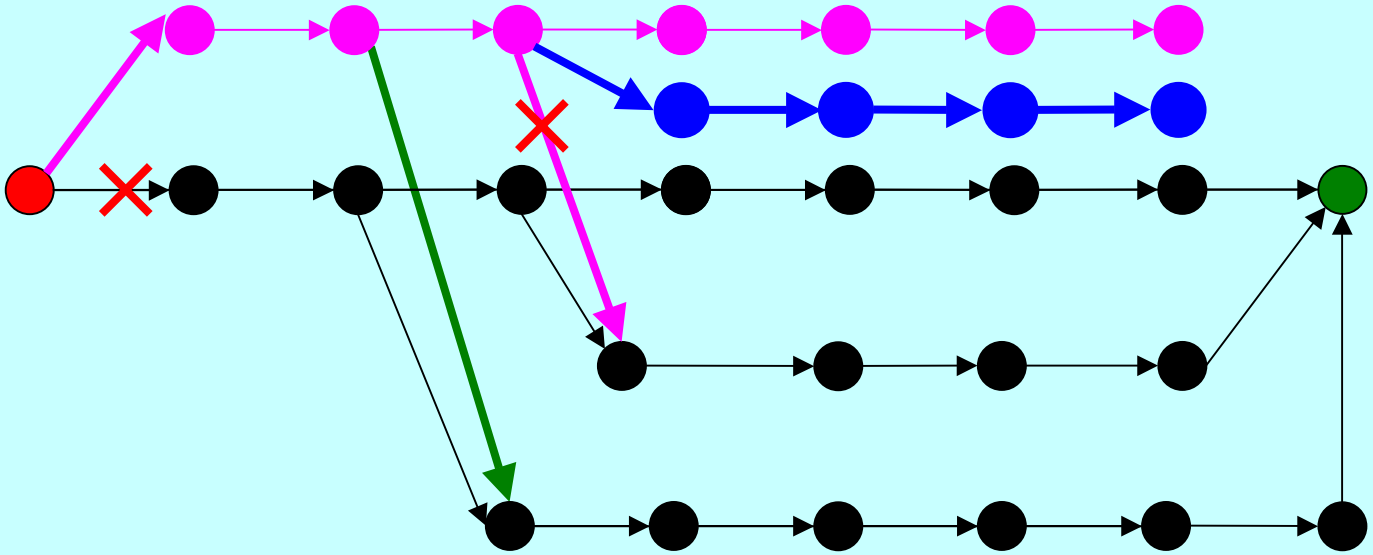
berechne den kürzesten Weg $w_k^{i_1,\ldots,i_k}$ und bilde dann das Minimum

$$w_k = \min_{i_1,..,i_k} \left\{ \delta\left(w_k^{i_1,\ldots,i_k} \right) \right\}$$

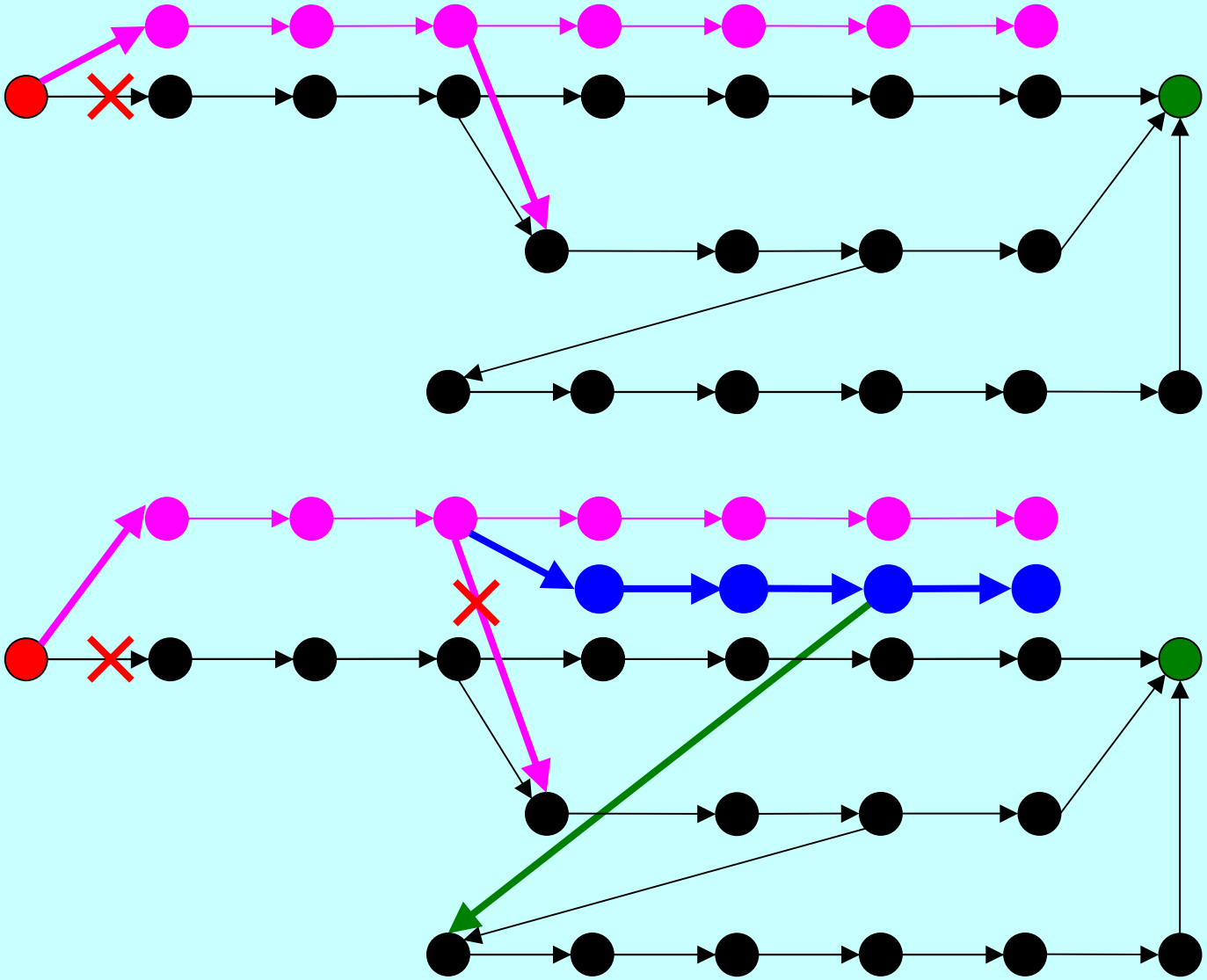
Der Aufwand liegt bei $O(\prod_{i=1}^{k} m_i) \cdot O(\text{Dijkstra}) \leq O(|V|^{k+1})$.

Drittkürzeste Wege nach Azevedo

Erschaffe einen neuen Startknoten und einen neuen Zielknoten. Berechne den kürzesten Weg w_1 und den zweitkürzesten Weg $w_2 = (e_1^2, .. , e_{m_2}^2)$. Der gemeinsame maximale Anfangsweg $(e_1^1 = e_1^2, e_2^1 = e_2^2, ... , e_j^1 = e_j^2)$ muß nicht verdoppelt werden. Es werden nur die Kanten $(e_{j+2}^2, .. , e_{m_2-1}^2)$ und die dazugehörigen Knoten verdoppelt. Die Kante e_{j+1}^2 über die der kürzeste Weg verlassen wird, bekommt einen neuen Endknoten. Ziehe alle Kanten über die w_2 verlassen werden kann.



Drittkürzeste Wege nach Azevedo



k - kürzeste Wege nach Azevedo

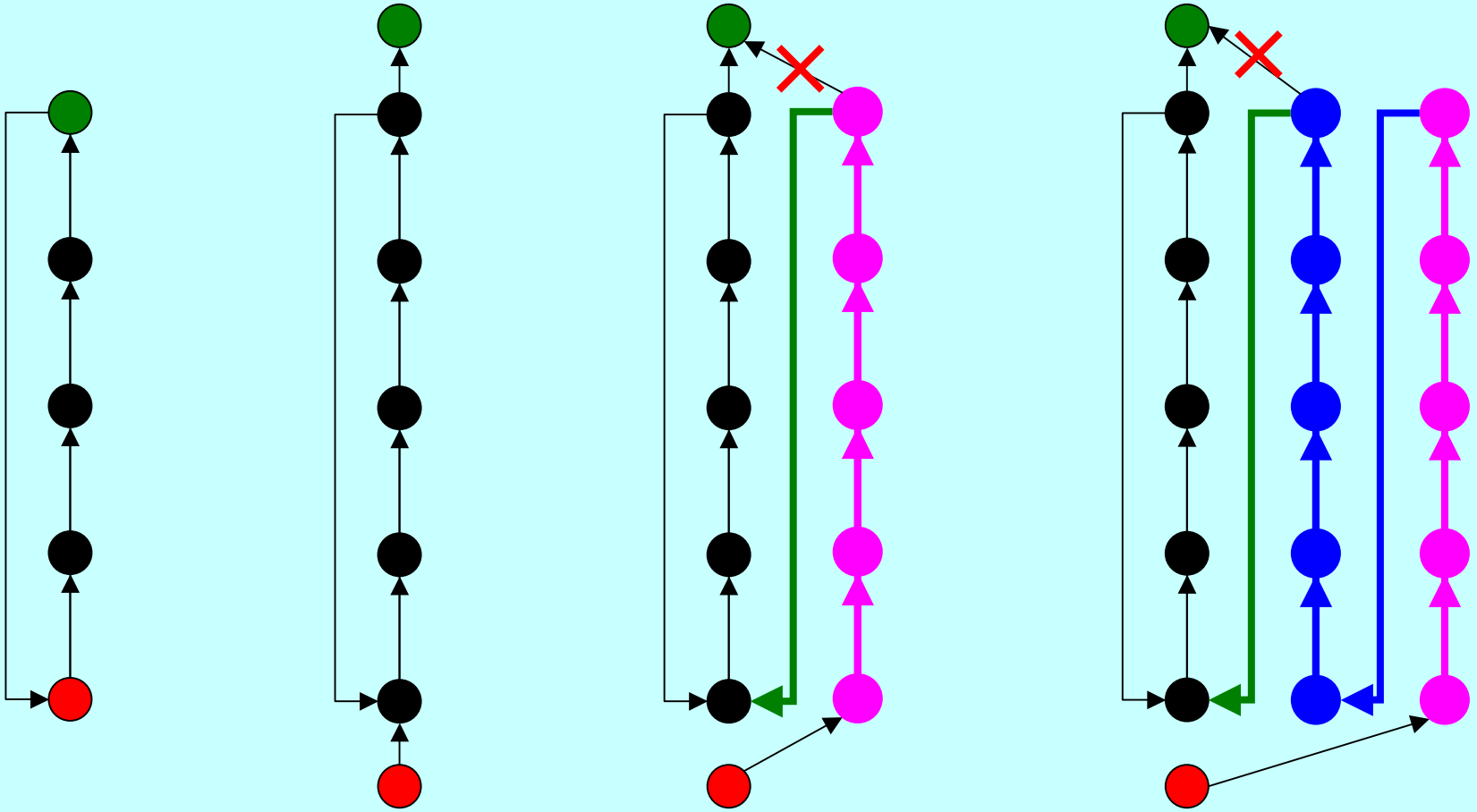
Berechnung des $k+1$ kürzesten Weges:

Erschaffe einen neuen Startknoten und einen neuen Zielknoten. Berechne die k - kürzesten Weg $w_1, .., w_k$ (mit w_i $(e_1^i, .. , e_{m_i}^i)$). Der gemeinsame maximale Anfangsweg $(e_1^l = e_1^k, e_2^l = e_2^k, ... , e_j^l = e_j^k)$ mit $l < k$, j maximal, muß nicht verdoppelt werden. Es werden nur die Kanten $(e_{j+2}^k, .. , e_{m_k-1}^k)$ und die dazugehörigen Knoten verdoppelt. Die Kante e_{j+1}^k über die w_l verlassen wird, bekommt einen neuen Endknoten. Ziehe alle Kanten über die w_k verlassen werden kann.

Der Aufwand liegt (vermutlich) bei $O((|E| + |V|) \cdot (k\text{-}1) \cdot \log |V|)$.

Der schlimmste Fall

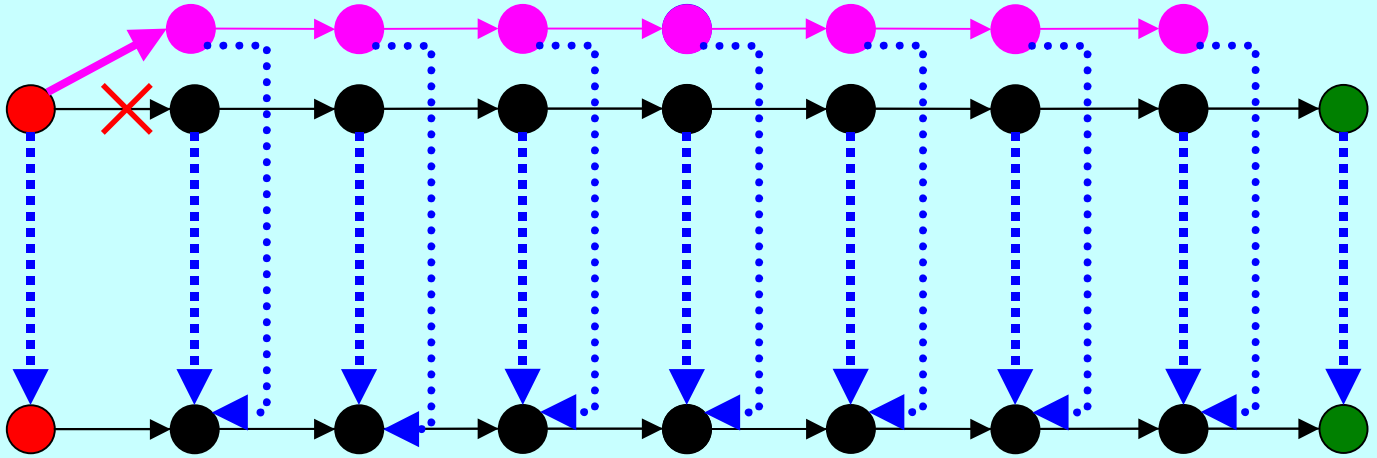
Gegeben sei ein Graph, der nur aus einem Zyklus besteht. Der k - kürzeste Weg von s nach z ergibt sich aus dem kürzesten Weg und dem anschliessenden k-1 maligen Durchlaufen des kompletten Graphen.



Beweis des Algorithmus von Azevedo

- Jeder Weg im erweiterten Graphen ist auch ein Weg im Ausgangsgraphen (die zeigt man über kanonische Projektionen = inverse Abbildung der Wegeverdopplung).

- Alle Wege (außer Wegen, die die k-kürzesten Wege enthalten) im Ausgangsgraphen, sind auch Wege im erweiteten Graphen (man konstruiert den Verlauf eines gegebenen Weges im erweiterten Graph).
  Genau die k-kürzesten Wege sind verboten.



Literatur

- J.A.de Azevedo, J.J. Madeira, E. Q. Martins, F. M. Pires: A shortest path ranking algorithm. AIRO 90 – Models and Methods for Decision Support, Sorrent (Italien), Proceedings of the Annual Conference, Assoziazione Italiana di Ricera Operativa (S. 1001-1011) 10.1990
- E. W. Dijkstra: A note on two problems in connection with graphs. Numerical Mathematics 1, (S. 269-271), 1959
- W. Knödel: Graphentheoretische Methoden und ihre Anwendungen. Springer Verlag Berlin, Heidelberg 1969
- M. Mack: Untersuchung von effizienten Algorithmen zur Bestimmung der k-kürzesten Wege innerhalb von ÖPNV-Verkehrsnetzen. Diplomarbeit Nr. 1374, Fakultät Informatik, Universität Stuttgart, 1996
- W. Schmid: Kürzeste Wege in Straßennetzen mit Wegeverboten. Verlag der Bayerischen Akademie der Wissenschaften, München, 2001