

Engineering Dijkstra's Shortest Path Algorithm ff

VO Algorithm Engineering

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

8. VO

21.11.2005

Überblick

Dijkstra für APSP mit LSP

Beispiel

Korrektheitsbeweise

Interleaved Dijkstra für APSP

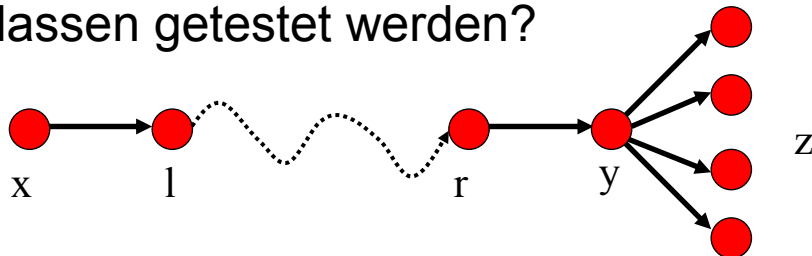
- Sei $\text{dist}(,)$ die Distanzmatrix
- $w(u,v)$ seien die Kosten für jede Kante $(u,v) \in E$
- Initialisierung:
 - Setze Prioritätswarteschlange $Q = \emptyset$

Annahme: Die kürzesten Wege sind eindeutig, d.h. es gibt nur jeweils einen kürzesten (x,y) -Weg zwischen jedem Knotenpaar.

- Für jedes Knotenpaar (x,y) :
 - Falls $(x,y) \in E$: $\text{dist}[x,y] \leftarrow w(x,y)$ Sonst $\text{dist}[x,y] = M$
 - $\text{InsertPrioQ}((x,y) , \text{dist}[x,y])$
- Solange $Q \neq \emptyset$:
 - $(x,y) \leftarrow \text{ExtractMinQ}()$
 - Für jede Kante $(y,z) \in E$, die y verläßt: ★
 - Falls $\text{dist}[x,z] > \text{dist}[x,y] + w(y,z)$:
 - $\text{dist}[x,z] \leftarrow \text{dist}[x,y] + w(y,z)$;
 - $\text{DecreasePrioQ}((x,z) , \text{dist}[x,z])$
 - Für jede Kante $(z,x) \in E$, die nach x zeigt: ★
 - Falls $\text{dist}[z,y] > w(z,x) + \text{dist}[x,y]$:
 - $\text{dist}[z,y] \leftarrow w(z,x) + \text{dist}[x,y]$;
 - $\text{DecreasePrioQ}((z,y) , \text{dist}[z,y])$
- Return $\text{dist}[]$ ★ = edge relaxation

Idee zur Reduktion

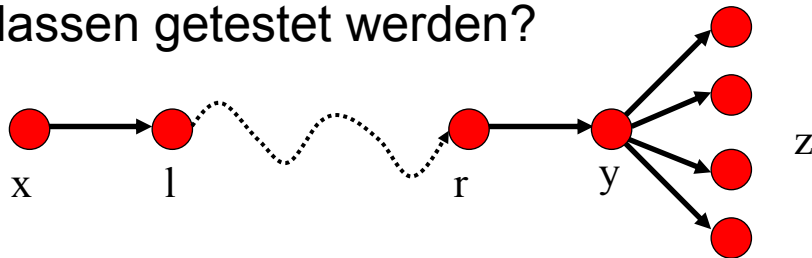
- Müssen tatsächlich alle Kanten (y,z) , die y verlassen getestet werden?



- Nicht, wenn diese nicht zu einem kürzesten Weg von x nach z gehören können.
- Jeder Teil-Weg eines kürzesten Weges ist ein kürzester Weg.
- Wenn $P(x \rightarrow z)$ kürzester Weg ist, dann müssen auch $P(x \rightarrow y)$ und $P(l \rightarrow z)$ kürzeste Wege sein.

Idee zur Reduktion

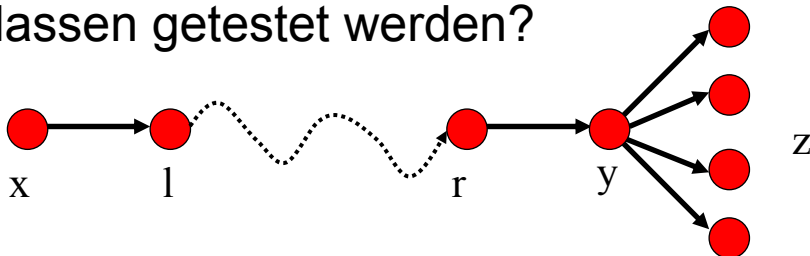
- Müssen tatsächlich alle Kanten (y,z) , die y verlassen getestet werden?



- IDEE: Teste den Weg $P(x \rightarrow z)$ erst dann, wenn man weiß, dass beide Teilwege $P(x \rightarrow y)$ und $P(l \rightarrow z)$ kürzeste Wege sind.
- Dann ist $P(x \rightarrow z)$ ein Locally Shortest Path.
- Wenn $P(x \rightarrow z)$ kürzester Weg ist, dann müssen auch $P(x \rightarrow y)$ und $P(l \rightarrow z)$ kürzeste Wege sein.

Idee zur Reduktion

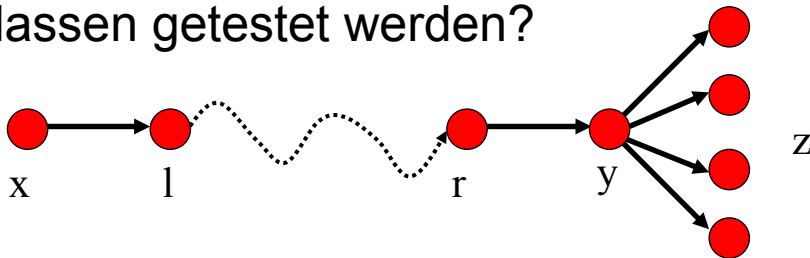
- Müssen tatsächlich alle Kanten (y,z) , die y verlassen getestet werden?



- Ein Weg $P(x \rightarrow z)$ heißt „Lokal Kürzester Weg“ („Locally Shortest Path“, LSP) in G , falls entweder
 - $P(x \rightarrow z)$ aus einem einzigen Knoten besteht, oder
 - jeder echte Teilweg von $P(x \rightarrow z)$ ein kürzester Pfad in G ist.

Änderung des Algorithmus

- Müssen tatsächlich alle Kanten (y,z) , die y verlassen getestet werden?



Für jedes Knotenpaar (x,y) halte Listen R_{xy} und L_{xy} :

$R_{xy} := \{ (y,z) \text{ mit } P(x \rightarrow z) = P(x \rightarrow y)(y,z) \text{ ist ein kürzester Weg} \}$

$L_{xy} := \{ (z,x) \text{ mit } P(z \rightarrow y) = (z,x)P(x \rightarrow y) \text{ ist ein kürzester Weg} \}$

- Diese Listen können gleichzeitig mit `ExtractMinQ()` aufgebaut werden
- Teste nur Kanten in R_{ly} und L_{xr} in Schritten ★

Algorithmus APSP-LSP

Datenstrukturen:

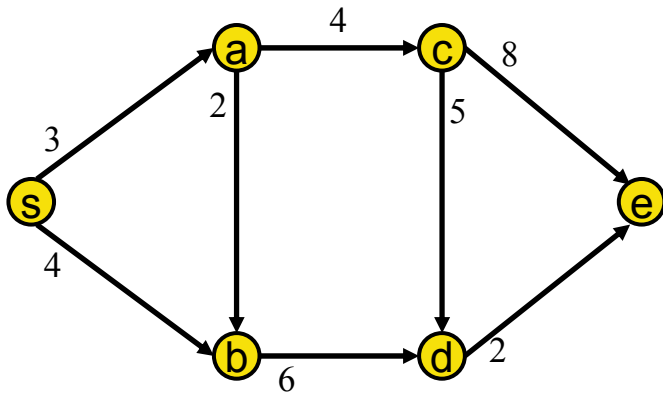
- $\text{dist}[x,y]$: Matrix: Distanzen zwischen x und y
- $\text{pred}(x)$: Array: Vorgängerknoten von Knoten x im Weg $P(x,y)$
- $\text{succ}(x)$: Array: Nachfolgeknoten von Knoten x im Weg $P(x,y)$
- $\text{ListL}(x,y)$: Matrix: Liste der möglichen $P(x,y)$ path extensions nach vorn (*left*)
- $\text{ListR}(x,y)$: Matrix: Liste der möglichen $P(x,y)$ path extensions nach hinten (*right*)

Initialisierung:

- Setze Prioritätswarteschlange $Q = \emptyset$

Annahme: Die kürzesten Wege sind eindeutig, d.h. es gibt nur jeweils einen kürzesten (x,y) -Weg zwischen jedem Knotenpaar.

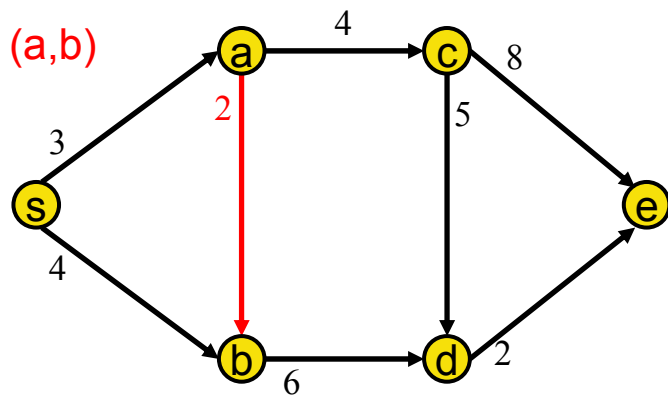
- Für jedes Knotenpaar (x,y) :
 - Falls $(x,y) \in E$: $\text{dist}[x,y] \leftarrow w(x,y)$ Sonst $\text{dist}[x,y] = M$
 - $\text{InsertPrioQ}((x,y) , \text{dist}[x,y])$
- Solange $Q \neq \emptyset$:
 - $(x,y) \leftarrow \text{ExtractMinQ}()$
 - $l \leftarrow \text{succ}(x)$; $r \leftarrow \text{pred}(y)$;
 - $\text{AppendListL}(l,y) \leftarrow x$;
 - $\text{AppendListR}(x,r) \leftarrow y$;
 - Für jede Kante (z,x) in $\text{ListL}(x,r)$:
 - Falls $\text{dist}[z,y] > w(z,x) + \text{dist}[x,y]$:
 - $\text{dist}[z,y] \leftarrow w(z,x) + \text{dist}[x,y]$;
 - $\text{DecreasePrioQ}((z,y) , \text{dist}[z,y])$
 - Für jede Kante (y,z) in $\text{ListR}(l,y)$:
 - Falls $\text{dist}[x,z] > \text{dist}(x,y) + w(y,z)$:
 - $\text{dist}[x,z] \leftarrow \text{dist}[x,y] + w(y,z)$;
 - $\text{DecreasePrioQ}((x,z) , \text{dist}[x,z])$



D	s	a	b	c	d	e
s	0	3	4	M	M	M
a		0	2	4	M	M
b			0	M	6	M
c				0	5	8
d					0	2

L	s	a	b	c	d	e
s						
a						
b						
c						
d						
e						

R	s	a	b	c	d	e
s						
a						
b						
c						
d						
e						11



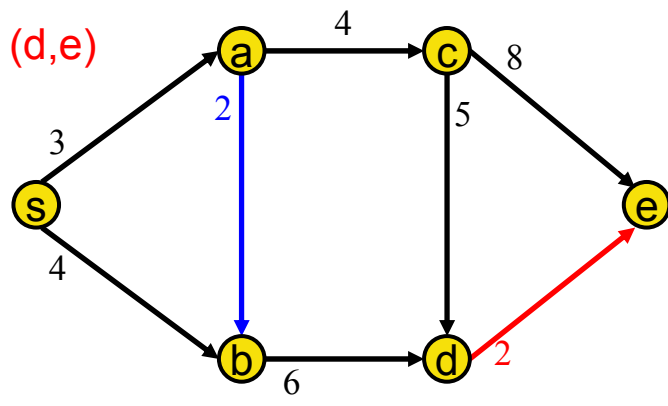
(a,b)

D	s	a	b	c	d	e
s	0	3	4	M	M	M
a		0	2	4	M	M
b			0	M	6	M
c				0	5	8
d					0	2

Teste: L(a,a) und R(b,b):

L	s	a	b	c	d	e
s						
a						
b			a			
c						
d						
e						

R	s	a	b	c	d	e
s						
a		b				
b						
c						
d						
e						

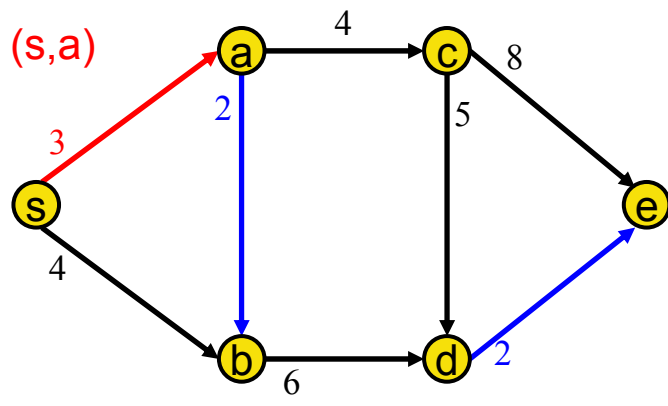


D	s	a	b	c	d	e
s	0	3	4	M	M	M
a		0	2	4	M	M
b			0	M	6	M
c				0	5	8
d					0	2

Teste: L(d,d) und R(e,e):

L	s	a	b	c	d	e
s						
a						
b			a			
c						
d						
e						d

R	s	a	b	c	d	e
s						
a		b				
b						
c						
d					e	
e						

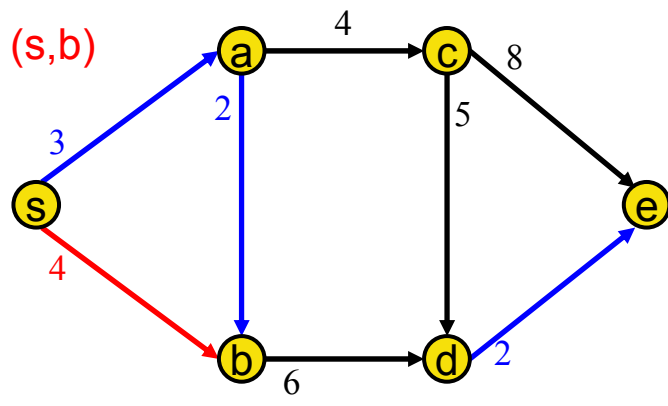


D	s	a	b	c	d	e
s	0	3	4	M	M	M
a		0	2	4	M	M
b			0	M	6	M
c				0	5	8
d					0	2

Teste: L(s,s) und R(a,a): P(s → a → b)

L	s	a	b	c	d	e
s						
a		s				
b			a			
c						
d						
e						d

R	s	a	b	c	d	e
s	a					
a		b				
b						
c						
d					e	
e						

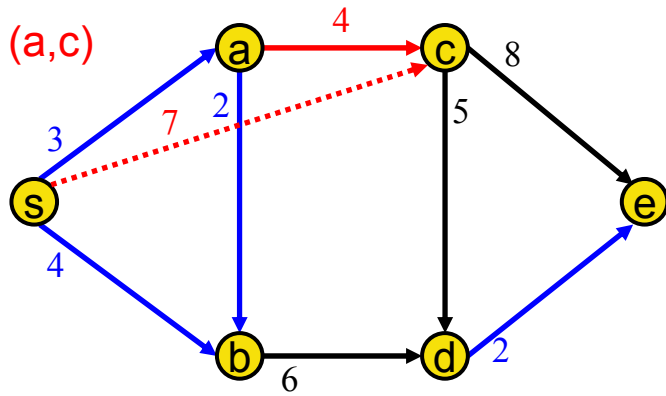


D	s	a	b	c	d	e
s	0	3	4	M	M	M
a		0	2	4	M	M
b			0	M	6	M
c				0	5	8
d					0	2

Teste: L(s,s) und R(b,b):

L	s	a	b	c	d	e
s						
a		s				
b			s,a			
c						
d						
e						d

R	s	a	b	c	d	e
s	b,a					
a		b				
b						
c						
d					e	
e						15

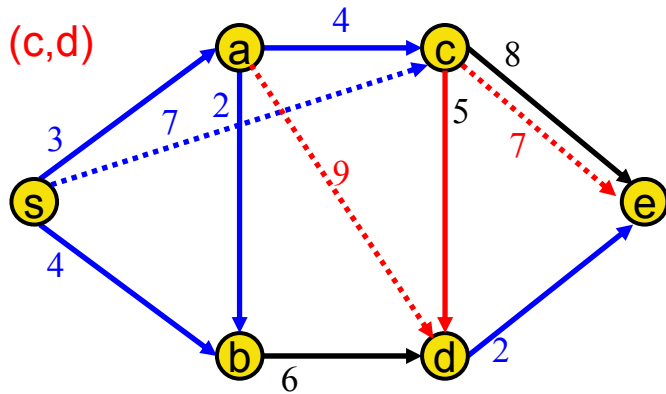


D	s	a	b	c	d	e
s	0	3	4	7	M	M
a		0	2	4	M	M
b			0	M	6	M
c				0	5	8
d					0	2

Teste: L(a,a) und R(c,c): $s \rightarrow a \rightarrow c$

L	s	a	b	c	d	e
s						
a		s				
b			s,a			
c				a		
d						
e						d

R	s	a	b	c	d	e
s	b,a					
a		c,b				
b						
c						
d					e	
e						

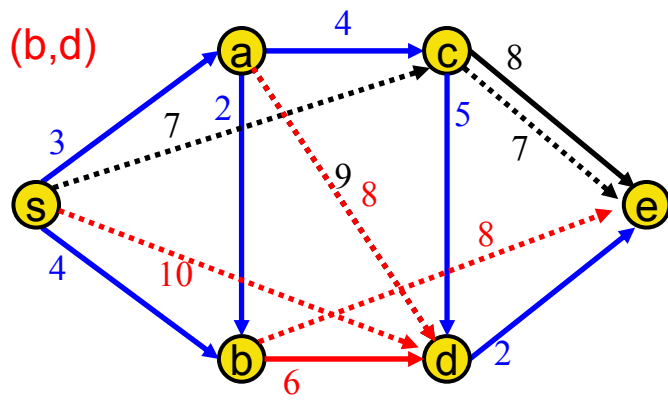


D	s	a	b	c	d	e
s	0	3	4	7	M	M
a		0	2	4	9	M
b			0	M	6	M
c				0	5	7
d					0	2

Teste: L(c,c) und R(d,d): $a \rightarrow c \rightarrow d, c \rightarrow d \rightarrow e$

L	s	a	b	c	d	e
s						
a		s				
b			s,a			
c				a		
d					c	
e						d

R	s	a	b	c	d	e
s	b,a					
a		c,b				
b						
c				d		
d					e	
e						

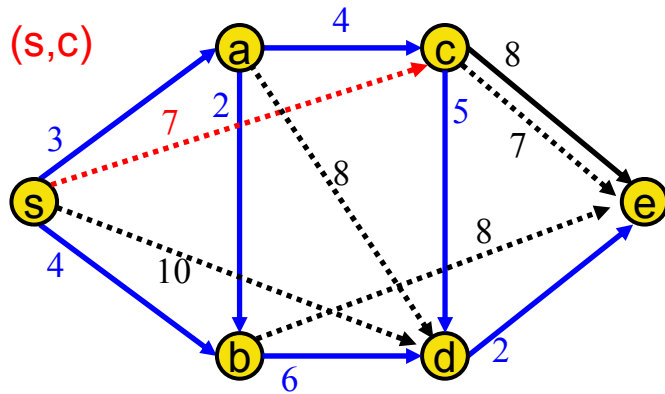


D	s	a	b	c	d	e
s	0	3	4	7	10	M
a		0	2	4	8	M
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(b,b) und R(d,d): $s \rightarrow b \rightarrow d$, $a \rightarrow b \rightarrow d$, $b \rightarrow d \rightarrow e$

L	s	a	b	c	d	e
s						
a		s				
b			s,a			
c				a		
d					b,c	
e						d

R	s	a	b	c	d	e
s	b,a					
a		c,b				
b			d			
c				d		
d					e	
e						

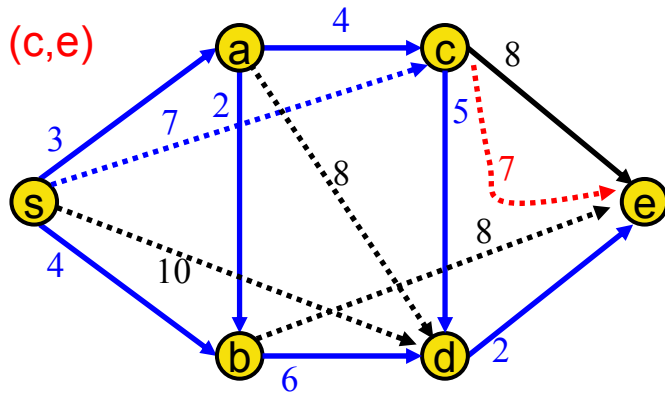


D	s	a	b	c	d	e
s	0	3	4	7	10	M
a		0	2	4	8	M
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(s,a) und R(a,c): NICHT: $s \rightarrow a \rightarrow c \rightarrow d$, $s \rightarrow a \rightarrow c \rightarrow e$

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a			
c				a		
d					b,c	
e						d

R	s	a	b	c	d	e
s		b,a	c			
a			c,b			
b				d		
c					d	
d						e
e						

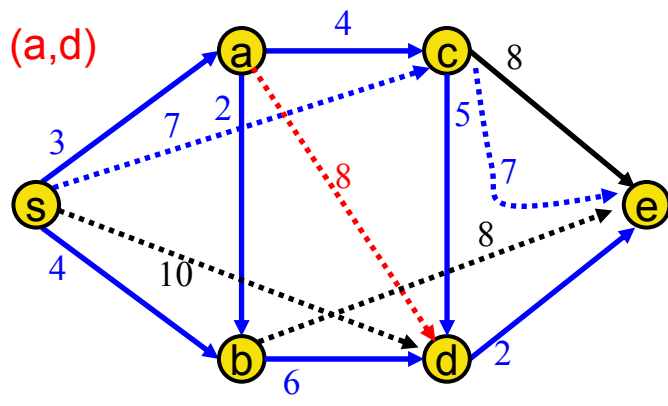


D	s	a	b	c	d	e
s	0	3	4	7	10	M
a		0	2	4	8	M
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(c,d) und R(d,e): **NICHT: a → c → d → e**

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a			
c				a		
d					b,c	c
e						d

R	s	a	b	c	d	e
s		b,a	c			
a			c,b			
b				d		
c					d	e
d						e
e						

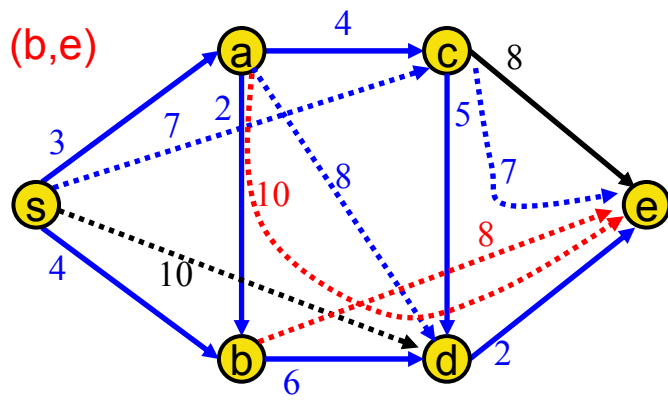


D	s	a	b	c	d	e
s	0	3	4	7	10	M
a		0	2	4	8	M
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(a,b) und R(b,d): NICHT: $a \rightarrow b \rightarrow d \rightarrow e, s \rightarrow a \rightarrow b \rightarrow d$

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		a	
c				a		
d					b,c	c
e						d

R	s	a	b	c	d	e
s		b,a	c			
a			c,b	d		
b				d		
c					d	e
d						e
e						

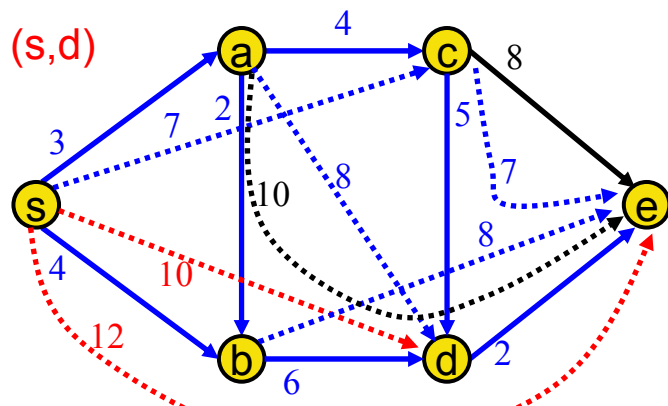


D	s	a	b	c	d	e
s	0	3	4	7	10	M
a		0	2	4	8	10
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(b,d) und R(d,e): $a \rightarrow b \rightarrow d \rightarrow e$, NICHT: $s \rightarrow b \rightarrow d \rightarrow e$

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		a	
c				a		
d					b,c	b,c
e						d

R	s	a	b	c	d	e
s		b,a	c			
a			c,b	d		
b				d		e
c					d	e
d						e
e						

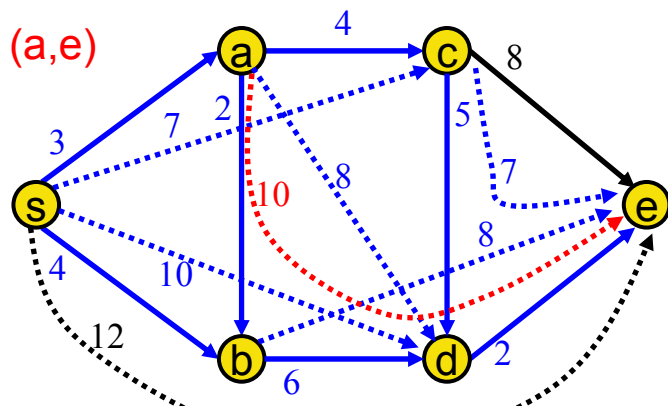


D	s	a	b	c	d	e
s	0	3	4	7	10	12
a		0	2	4	8	10
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(s,b) und R(b,d): $s \rightarrow b \rightarrow d \rightarrow e$

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		s,a	
c				a		
d					b,c	b,c
e						d

R	s	a	b	c	d	e
s		b,a	c	d		
a			c,b	d		
b				d		e
c					d	e
d						e
e						

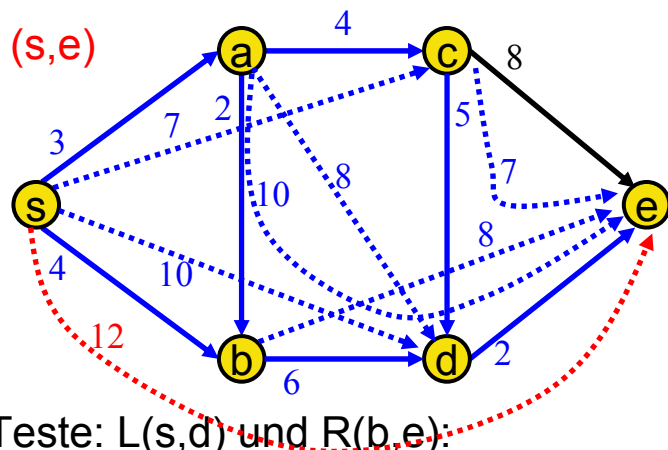


D	s	a	b	c	d	e
s	0	3	4	7	10	12
a		0	2	4	8	10
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(a,d) und R(b,e): NICHT $s \rightarrow a \rightarrow b \rightarrow d \rightarrow e$

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		s,a	a
c				a		
d					b,c	b,c
e						d

R	s	a	b	c	d	e
s		b,a	c	d		
a			c,b	d		e
b				d		e
c					d	e
d						e
e						

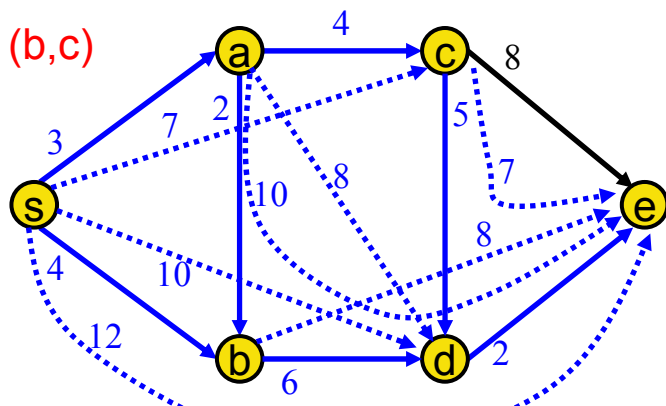


D	s	a	b	c	d	e
s	0	3	4	7	10	12
a		0	2	4	8	10
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(s,d) und R(b,e):

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		s,a	s,a
c				a		
d					b,c	b,c
e						d

R	s	a	b	c	d	e
s	b,a	c	d		e	
a		c,b	d		e	
b			d		e	
c				d	e	
d					e	
e						



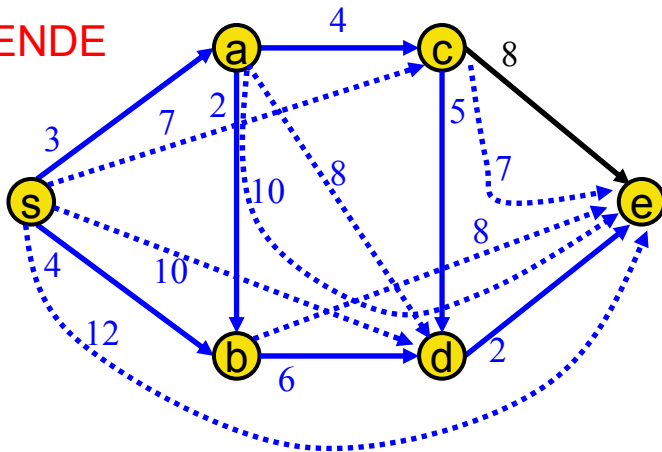
D	s	a	b	c	d	e
s	0	3	4	7	10	12
a		0	2	4	8	10
b			0	M	6	8
c				0	5	7
d					0	2

Teste: L(s,d) und R(b,e):

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		s,a	s,a
c				a		
d					b,c	b,c
e						d

R	s	a	b	c	d	e
s	b,a	c	d		e	
a		c,b	d		e	
b			d		e	
c				d	e	
d					e	
e						

ENDE



D	s	a	b	c	d	e
s	0	3	4	7	10	12
a		0	2	4	8	10
b			0	M	6	8
c				0	5	7
d					0	2

L	s	a	b	c	d	e
s						
a		s		s		
b			s,a		s,a	s,a
c				a		
d					b,c	b,c
e						d

R	s	a	b	c	d	e
s	b,a	c	d		e	
a		c,b	d		e	
b			d		e	
c				d	e	
d					e	
e						

Analyse: Beobachtungen

Beob1: Jeder kürzeste Weg $P(x,y)$ der Länge mind. 1 gehört zur Liste $ListL(l,y)$ und zur Liste $ListR(x,r)$.

Beob. 2: Durch LSP-Restriktion geht kein kürzester Weg verloren.

Denn: Sei $x \rightarrow l \rightarrow \dots \rightarrow r \rightarrow y$ ein kürzester Weg.

Annahme: $l \rightarrow \dots \rightarrow y$ wird zuerst aus Q extrahiert,
dann: $AppendListR(l,r) \rightarrow y$.

Wenn dann später $x \rightarrow \dots \rightarrow r$ bei Q extrahiert, dann
wird aufgerufen: edge relaxation von $ListR(l,r)$.

Analyse: Korrektheit

Theorem: Nach Ablauf des Algorithmus APSP-LSP enthält $\text{dist}()$ für alle Knotenpaare die korrekten Distanzen.

Beweis: Ind.Beginn: wahr für kürzeste Wege der Länge 1.

- Ind.Ann: wahr für kürzeste Wege der Länge $\leq k-1$
- Ind.Schritt: z.z.: wahr für Wege der Länge k :
 - Betrachte kürzesten Weg der Länge k : $x \rightarrow l \rightarrow \dots \rightarrow r \rightarrow y$
 - wahr für Teilwege $x \rightarrow l \rightarrow \dots \rightarrow r$ und $l \rightarrow \dots \rightarrow r \rightarrow y$
 - o.E: sei $l \rightarrow \dots \rightarrow r \rightarrow y$ zuerst extrahiert, dann wird y in $\text{ListR}(l,r)$ eingetragen
 - später wird $x \rightarrow l \rightarrow \dots \rightarrow r$ extrahiert, dann werden bei (★) alle Elemente in $\text{ListR}(l,r)$ getestet, u.a. auch y : also der Weg $x \rightarrow l \rightarrow \dots \rightarrow r \rightarrow y$ und erhält somit korrekte Distanz.

Analyse: Laufzeit

Lemma: Im Schritt „edge relaxation“ werden nur LSPs getestet.

Annahme: Der Weg $x \rightarrow l \rightarrow \dots \rightarrow r \rightarrow y$ sei kein LSP.

Dann ist entweder $x \rightarrow l \rightarrow \dots \rightarrow r$ oder $l \rightarrow \dots \rightarrow r \rightarrow y$ kein kürzester Weg. Ann: letzterer.

Dann wird dieser nie in `ExtractMinQ()` extrahiert, d.h. y wird nicht in `ListR(l,r)` eingetragen.

Falls irgendwann der kürzeste Weg $x \rightarrow \dots \rightarrow r$ extrahiert werden sollte (falls dieser kürzester Weg wäre), dann ist y nicht in `ListR(l,r)` enthalten, dies wäre aber die einzige Möglichkeit gewesen, den Weg nach y zu verlängern, d.h. um $x \rightarrow l \rightarrow \dots \rightarrow r \rightarrow y$ zu testen.

Von Engineering zu Theorie

- Implementierungen:
<http://www.dis.uniroma1.it/~demetres/experim/dsp/>
- Entdeckung: LSPs sind bei dynamischen Graphen hilfreich!
- → neuer Algorithmus mit asymptotisch bester Laufzeit: Nächste VO

