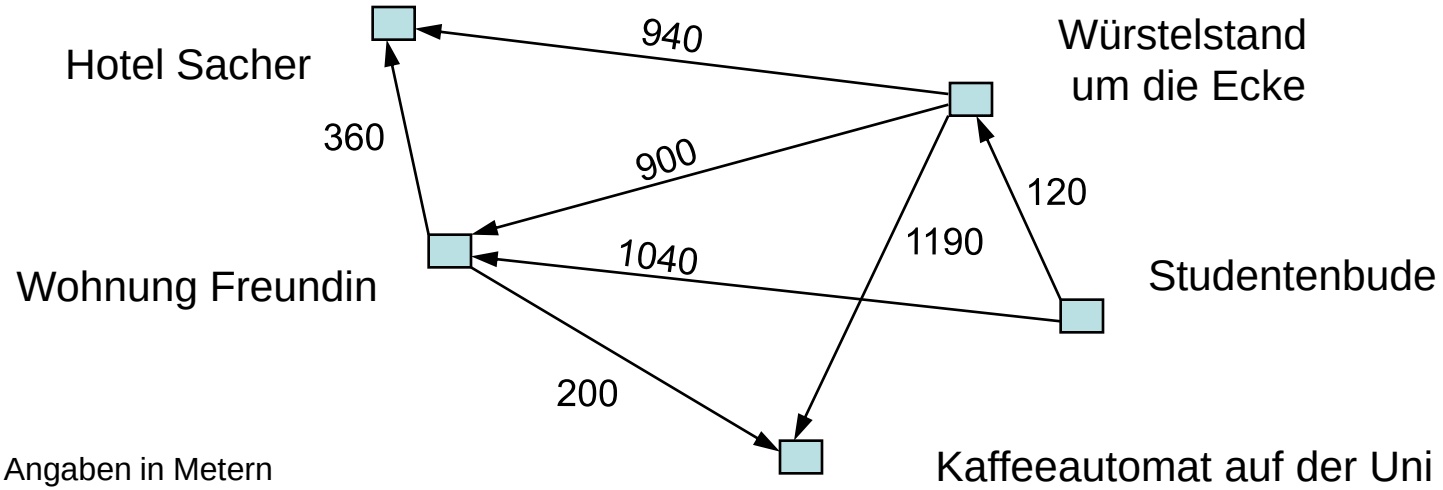
Floyd Algorithmus D(1)

Erstellung der Adjazenzmatrix:

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	**0**
Variable i	**0**
Variable j	**0**
newPl	**undefiniert**



Angaben in Metern

Floyd Algorithmus D(2)

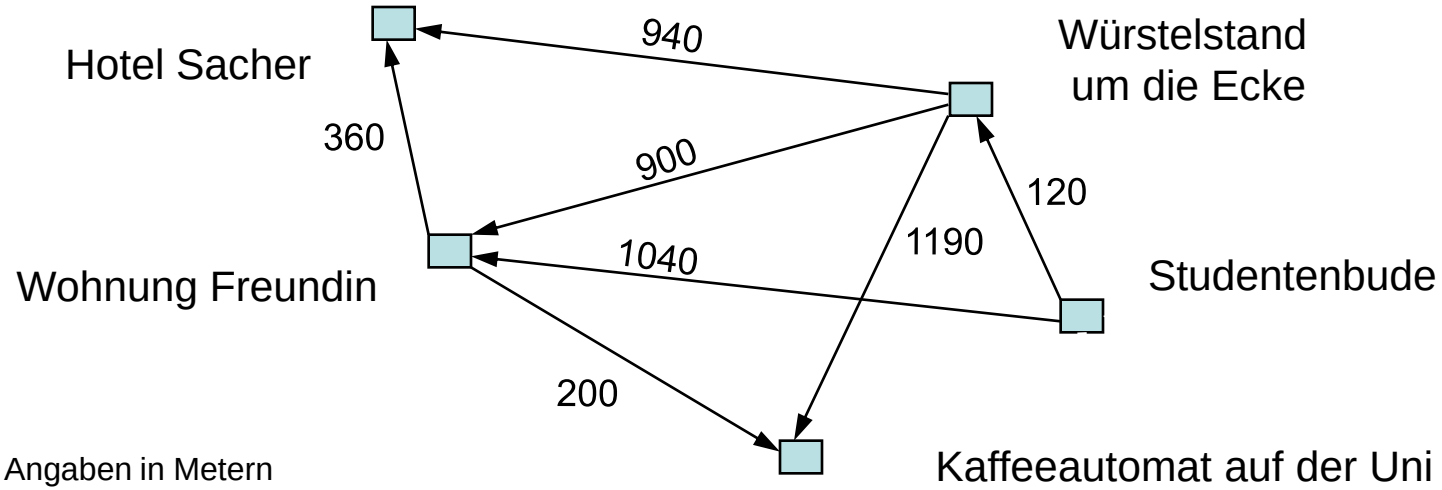
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[i,k] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	**0**
Variable i	**0**
Variable j	**0**
newPl	**undefiniert**



Floyd Algorithmus D(3)

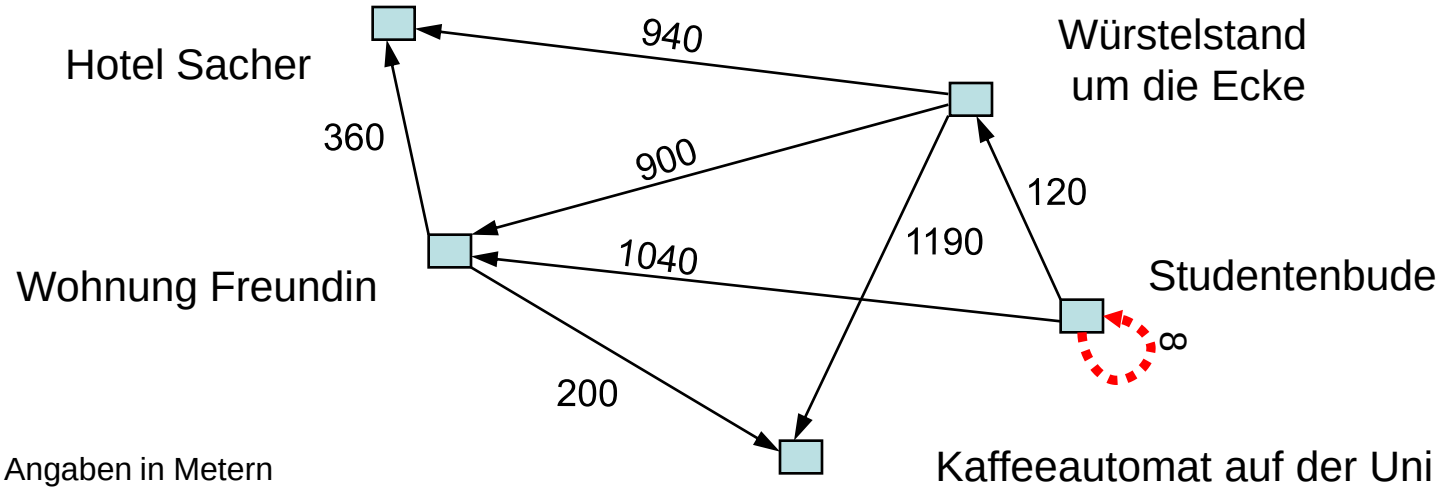
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[0,0] + adjm[0,0];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	0
Variable j	0
newPl	∞



Angaben in Metern

Floyd Algorithmus D(4)

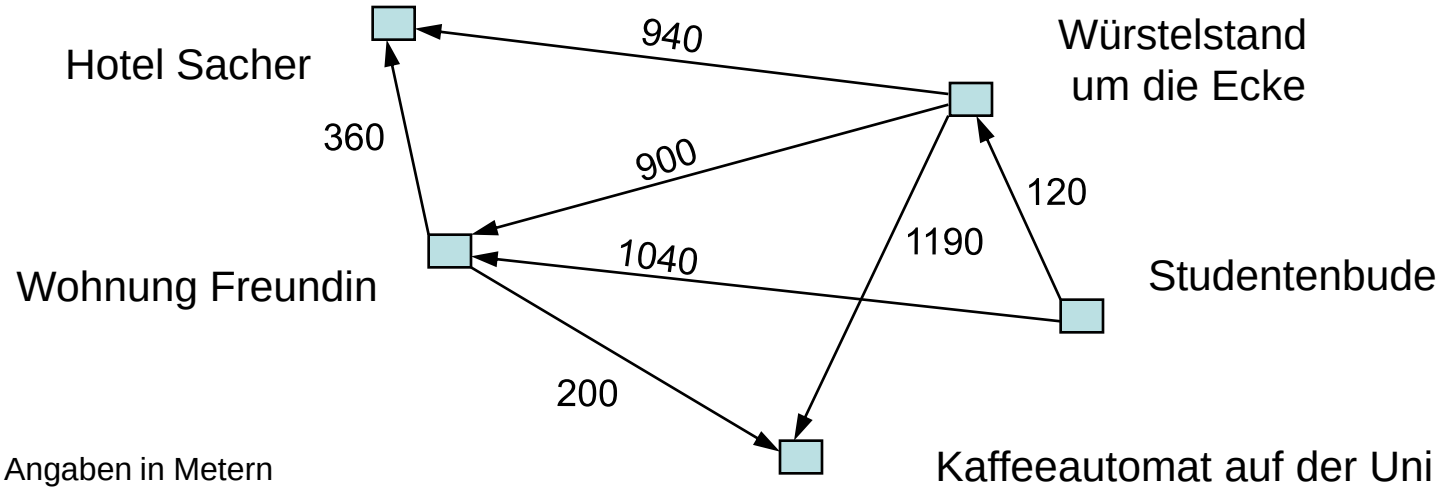
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[i,k] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	0
Variable j	1
newPl	



Angaben in Metern

Floyd Algorithmus D(5)

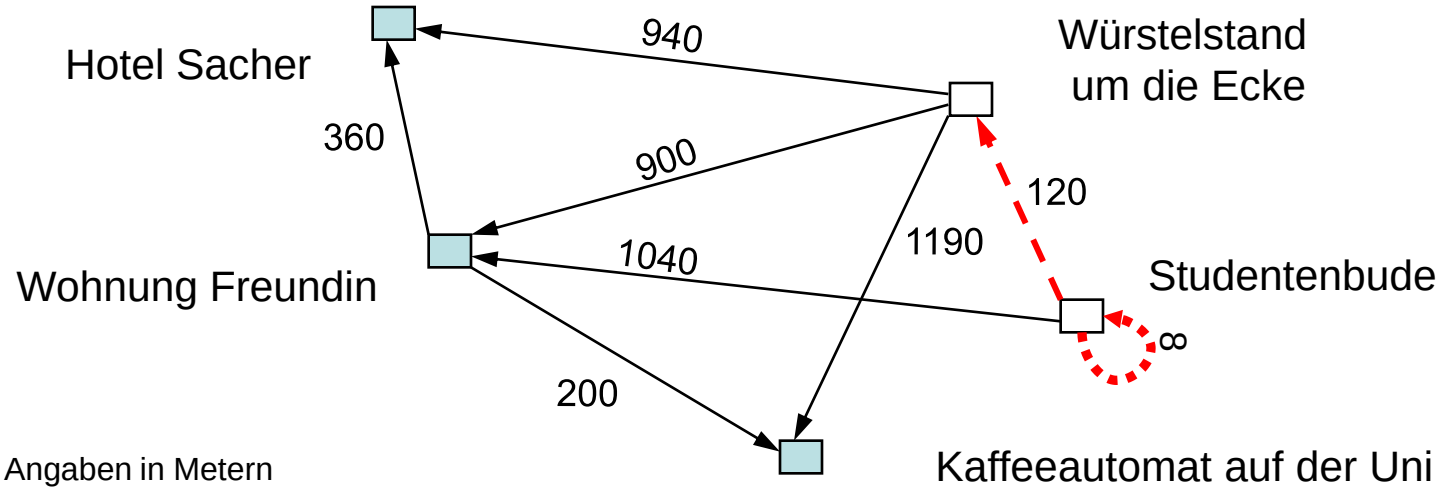
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[0,0] + adjm[0,1];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	0
Variable j	1
newPl	∞



Angaben in Metern

Floyd Algorithmus D(6)

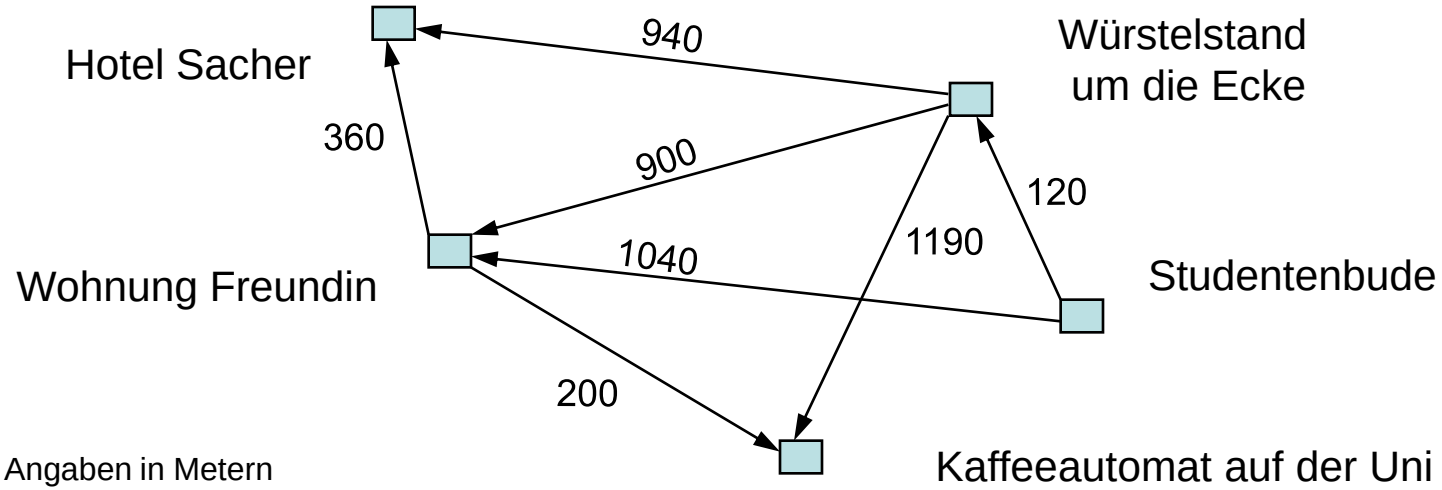
int newPathlength = adjm[0,0] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN >>>FALSE<<<

adjm[i,j] = newPathlength;

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	0
Variable j	1
newPl	∞

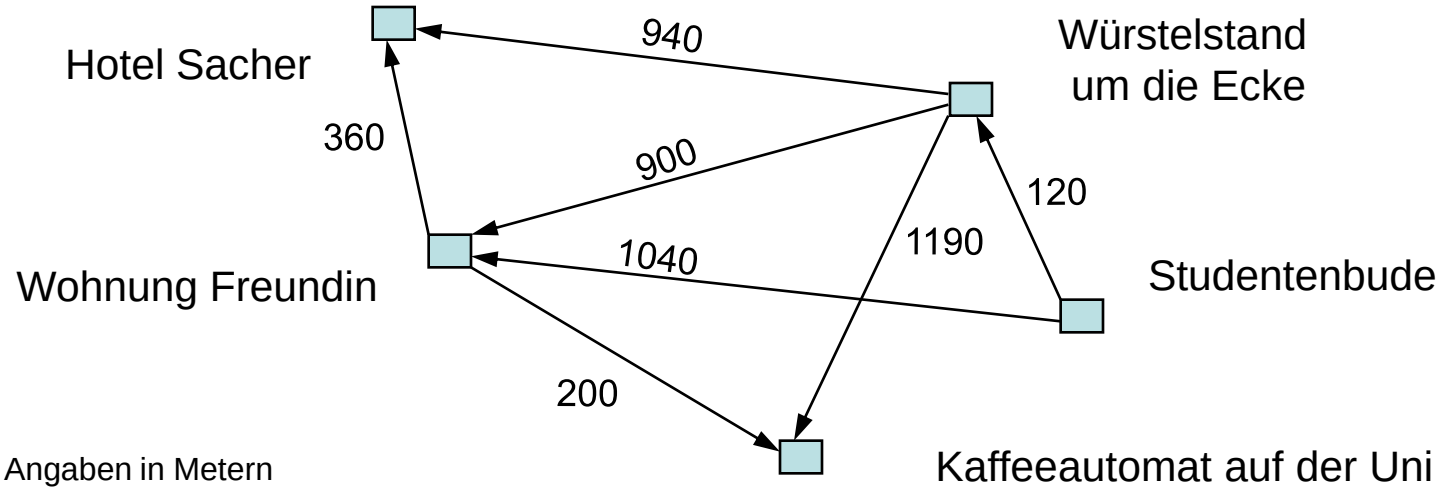


Floyd Algorithmus D(7)

Wir spulen etwas vor und lassen j=2, j=3 ... j=n-1 aus,
und setzen mit k=0, i=1, und j=0 fort...

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	1
Variable j	0
newPl	



Angaben in Metern

Floyd Algorithmus D(8)

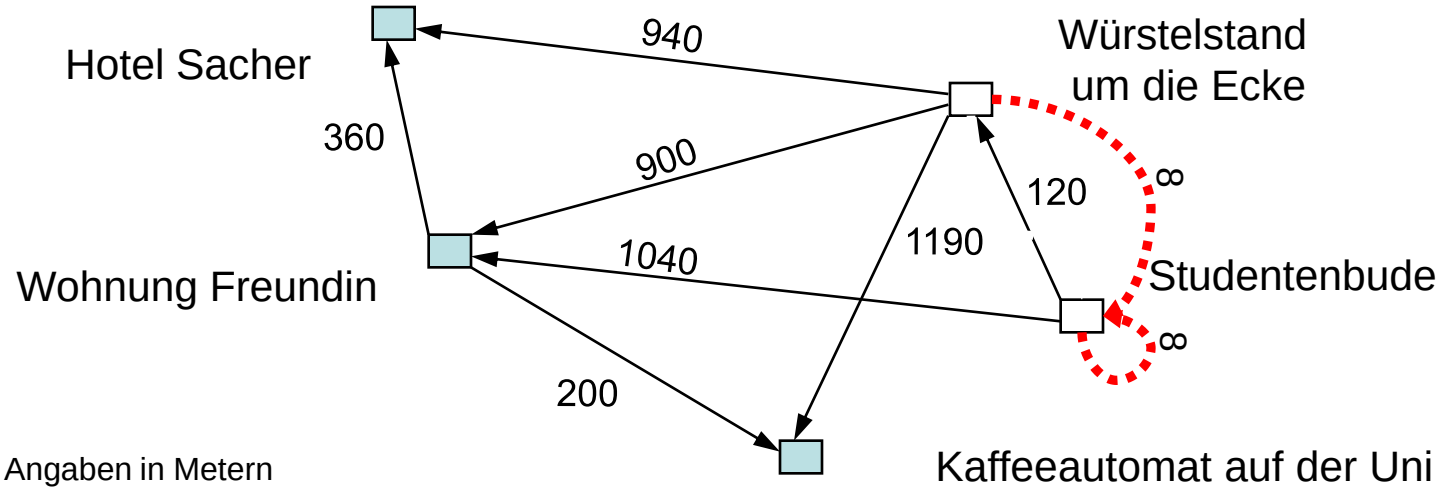
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[1,0] + adjm[0,0];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	1
Variable j	0
newPl	∞



Angaben in Metern

Floyd Algorithmus D(9)

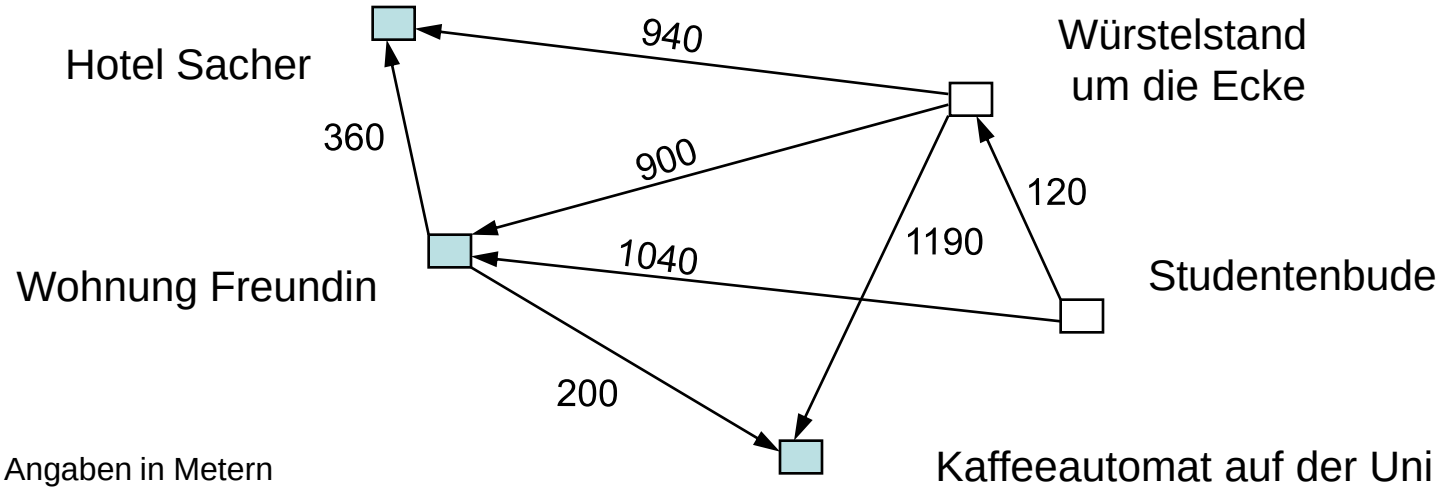
int newPathlength = adjm[0,0] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN >>>FALSE<<<

adjm[i,j] = newPathlength;

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	0
Variable i	1
Variable j	0
newPl	∞



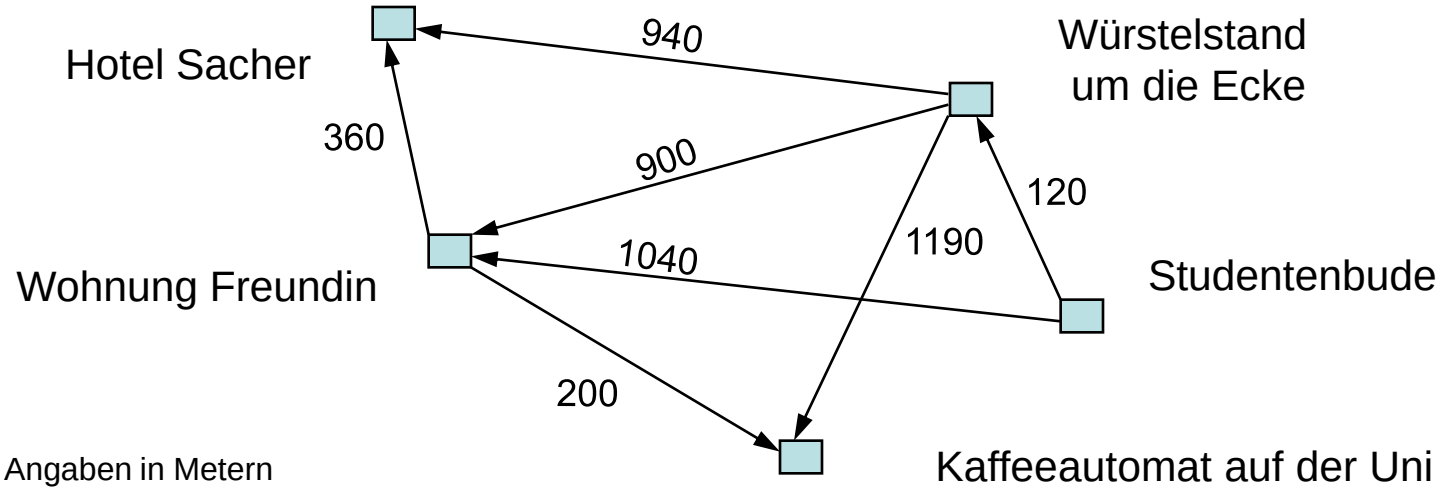
Floyd Algorithmus D(10)

Wir spulen etwas vor und lassen alle weiteren Durchläufe für k=0 aus.

Wir setzen mit k=1, i=0, und j=0 fort...

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	0
newPl	



Angaben in Metern

Floyd Algorithmus D(11)

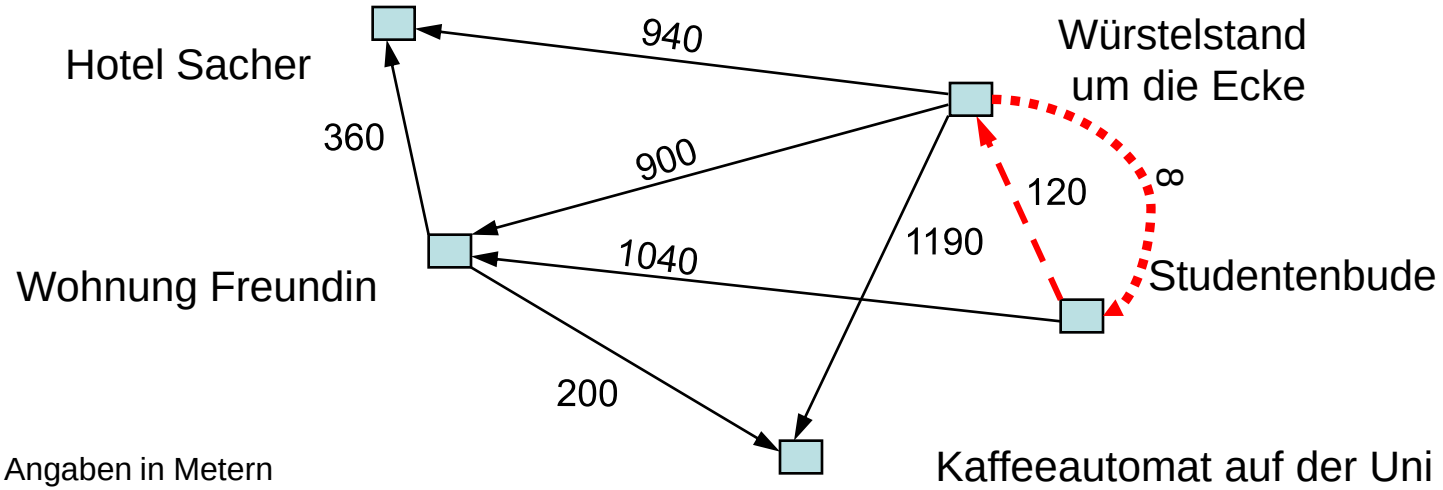
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[0,1] + adjm[1,0];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	0
newPl	∞



Angaben in Metern

Floyd Algorithmus D(12)

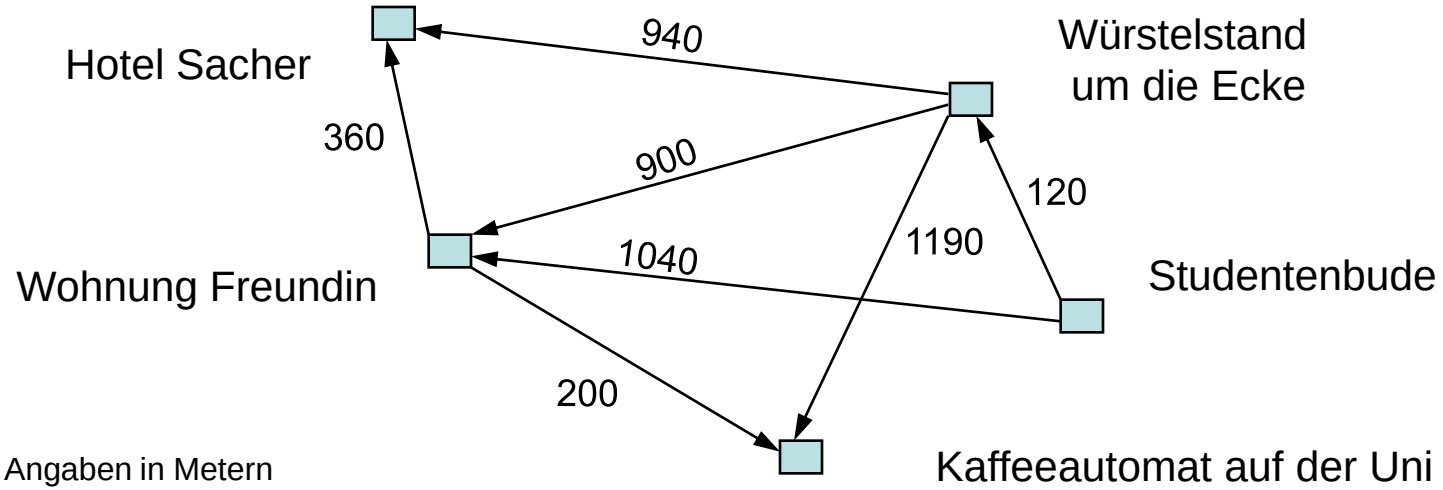
int newPathlength = adjm[0,0] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN >>>FALSE<<<

adjm[i,j] = newPathlength;

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	0
newPl	∞



Floyd Algorithmus D(13)

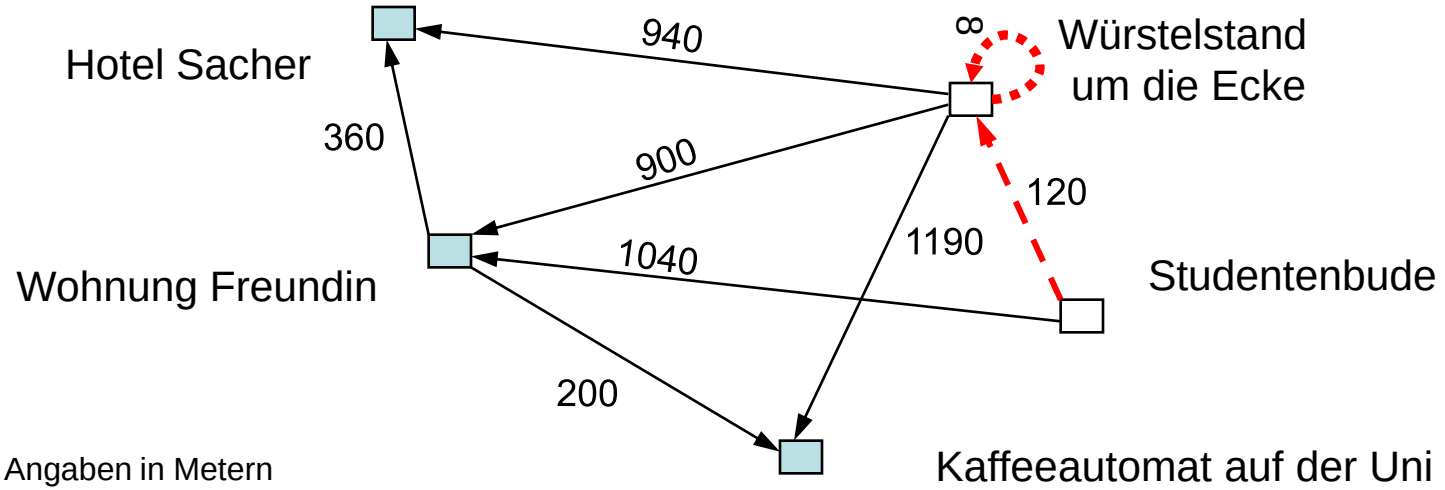
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[0,1] + adjm[1,1];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	1
newPl	∞



Angaben in Metern

Floyd Algorithmus D(14)

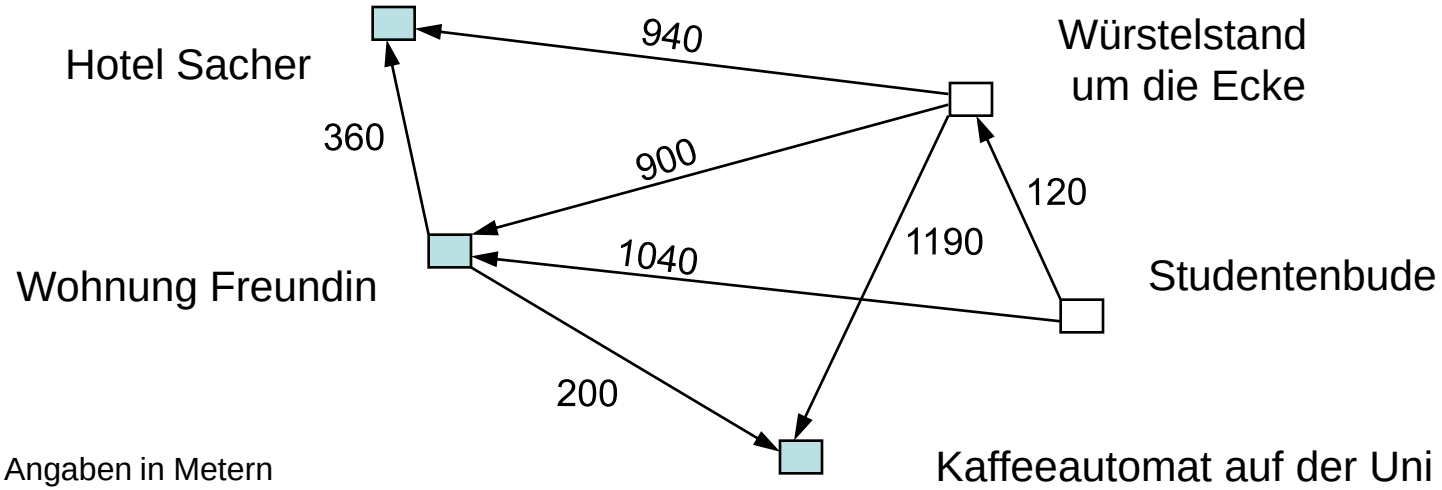
int newPathlength = adjm[0,0] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN >>>FALSE<<<

adjm[i,j] = newPathlength;

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	1
newPl	∞



Floyd Algorithmus D(15)

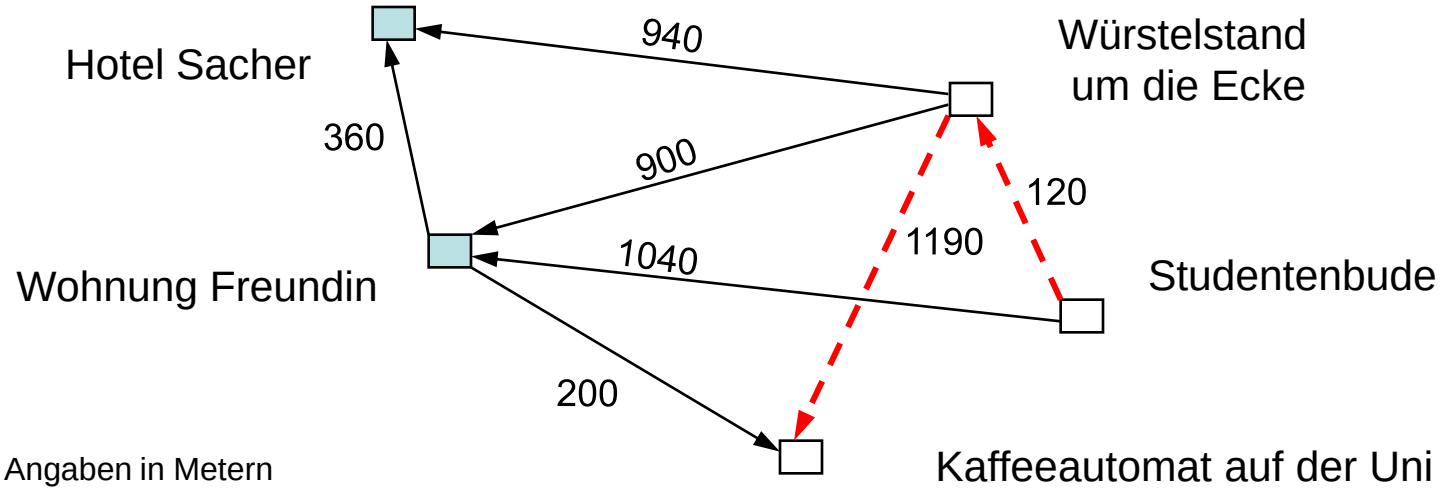
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[0,1] + adjm[1,2];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	2
newPl	1310



Angaben in Metern

Floyd Algorithmus D(16)

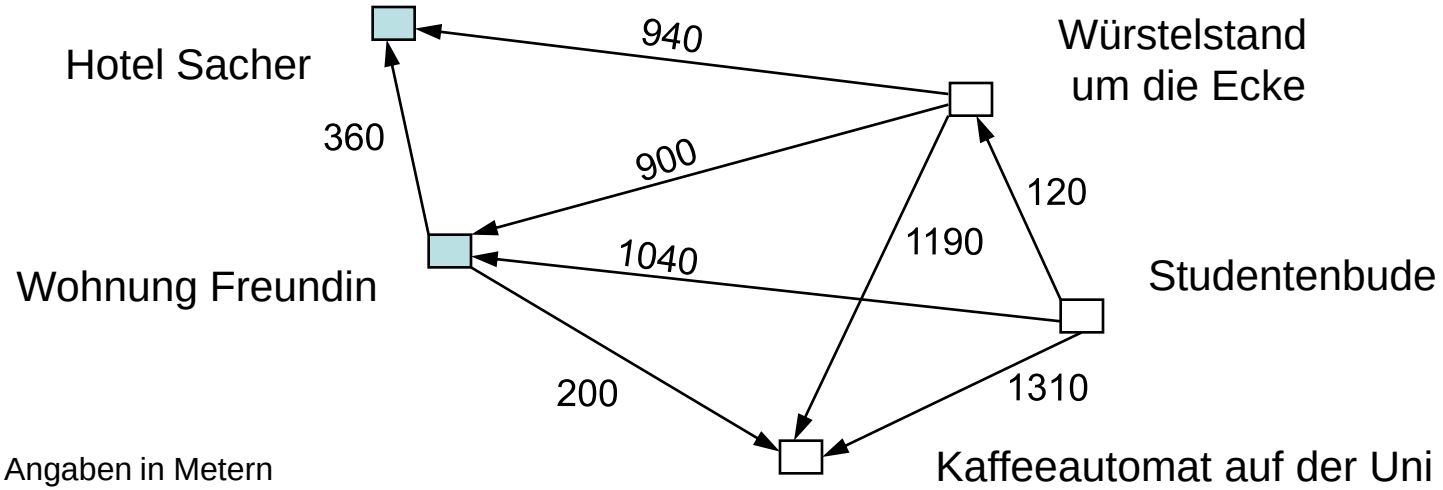
int newPathlength = adjm[0,0] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN >>>TRUE<<<

adjm[i,j] = newPathlength;

	SB	WS	KA	WF	HS
SB	∞	120	∞	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	**1**
Variable i	**0**
Variable j	**2**
newPl	**1310**



Angaben in Metern

Floyd Algorithmus D(17)

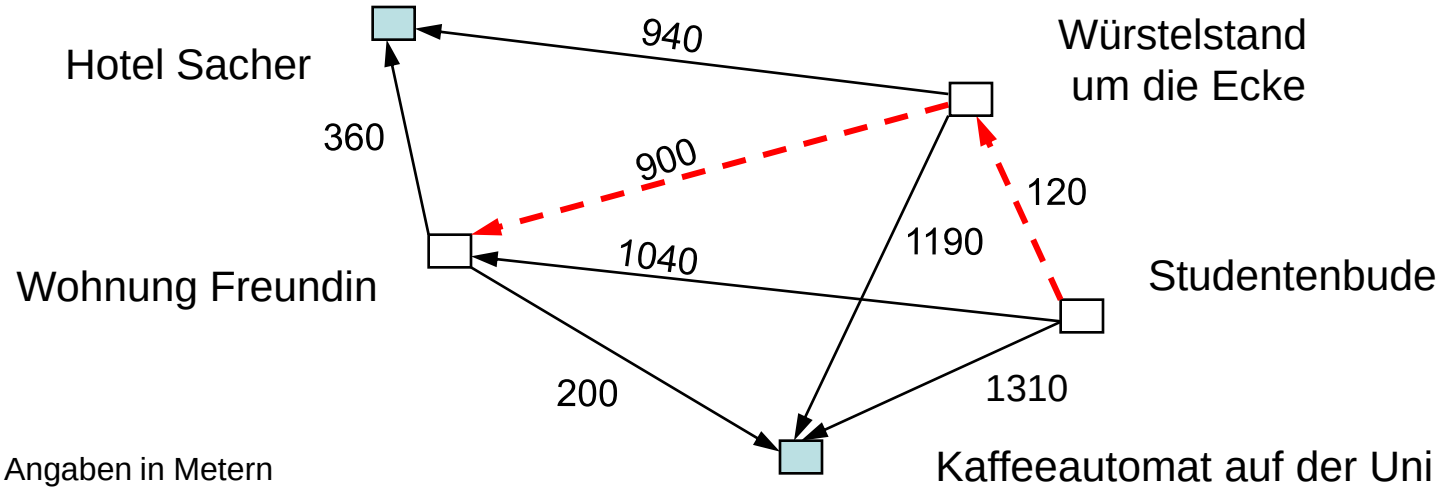
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[0,1] + adjm[1,3];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	1310	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	3
newPl	1020



Angaben in Metern

Floyd Algorithmus D(18)

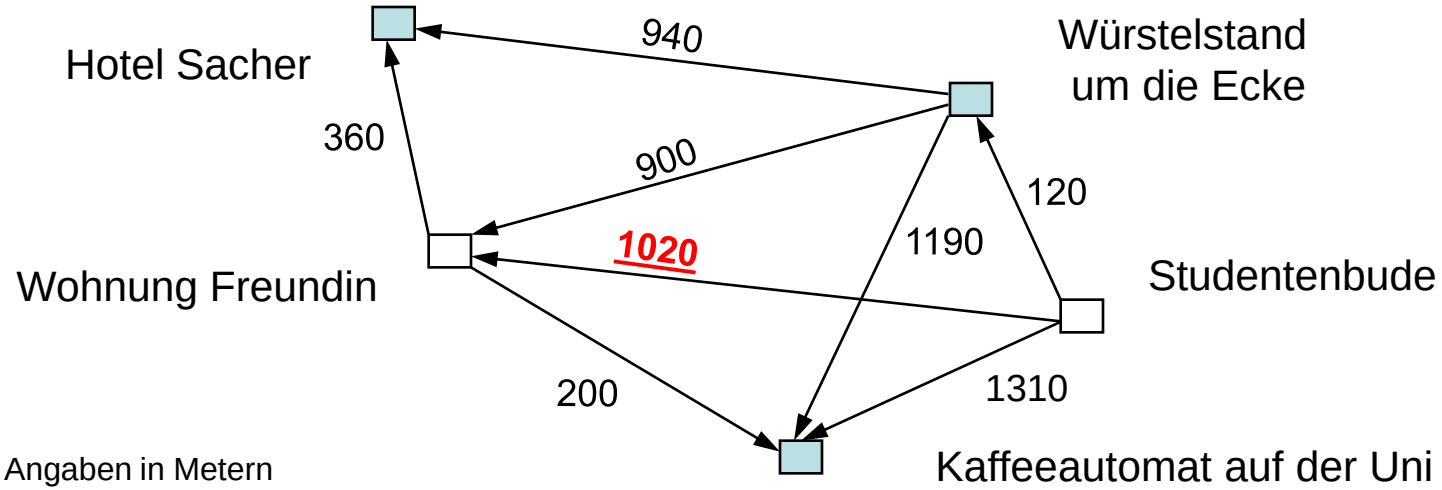
IF newPathlength < adjm[i,j] THEN

adjm[i,j] = newPathlength; >>>TRUE<<<

END;

	SB	WS	KA	WF	HS
SB	∞	120	1310	1020	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	3
newPl	1020



Angaben in Metern

Floyd Algorithmus D(19)

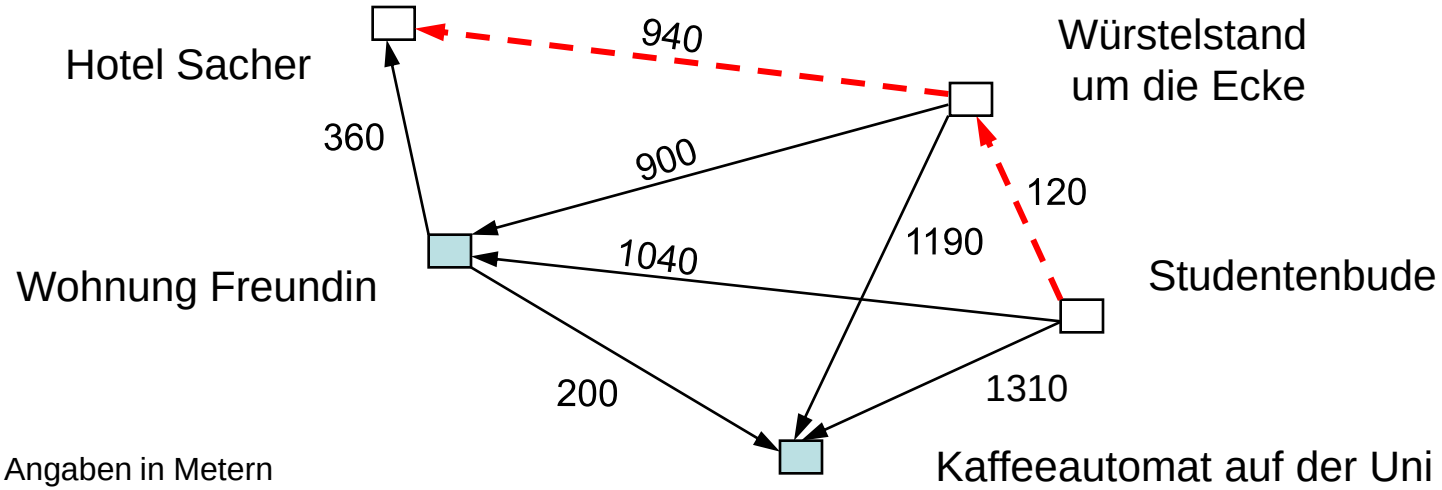
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[i,k] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	120	1310	1040	∞
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	4
newPl	1060



Angaben in Metern

Floyd Algorithmus D(20)

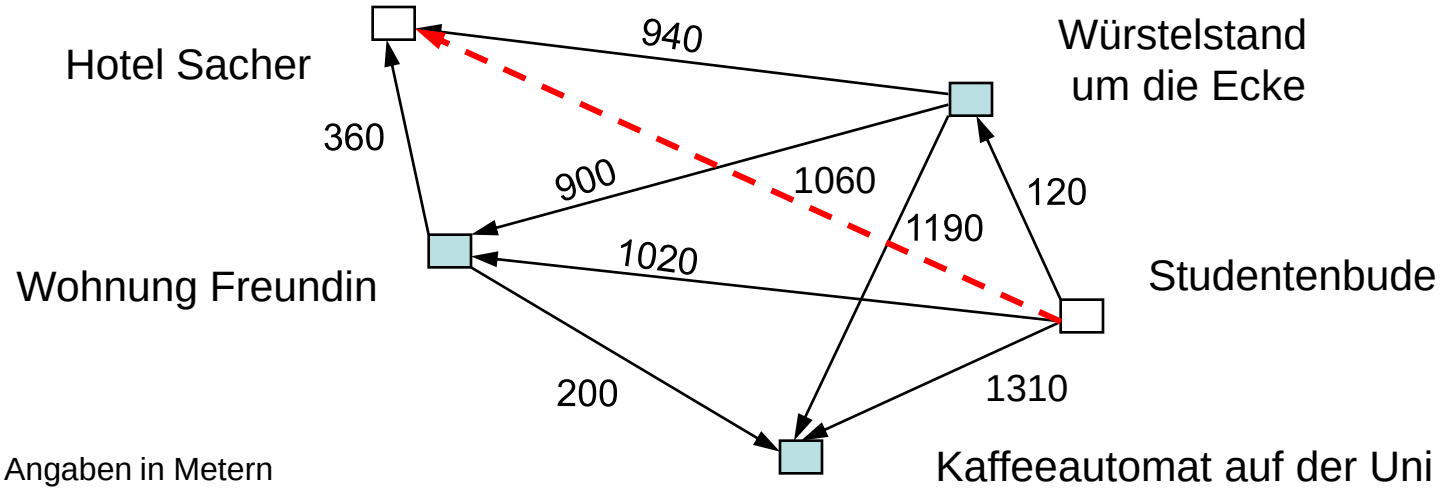
IF newPathlength < adjm[i,j] THEN

adjm[i,j] = newPathlength; >>>TRUE<<<

END;

	SB	WS	KA	WF	HS
SB	∞	120	1310	1020	1060
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	4
newPl	1060



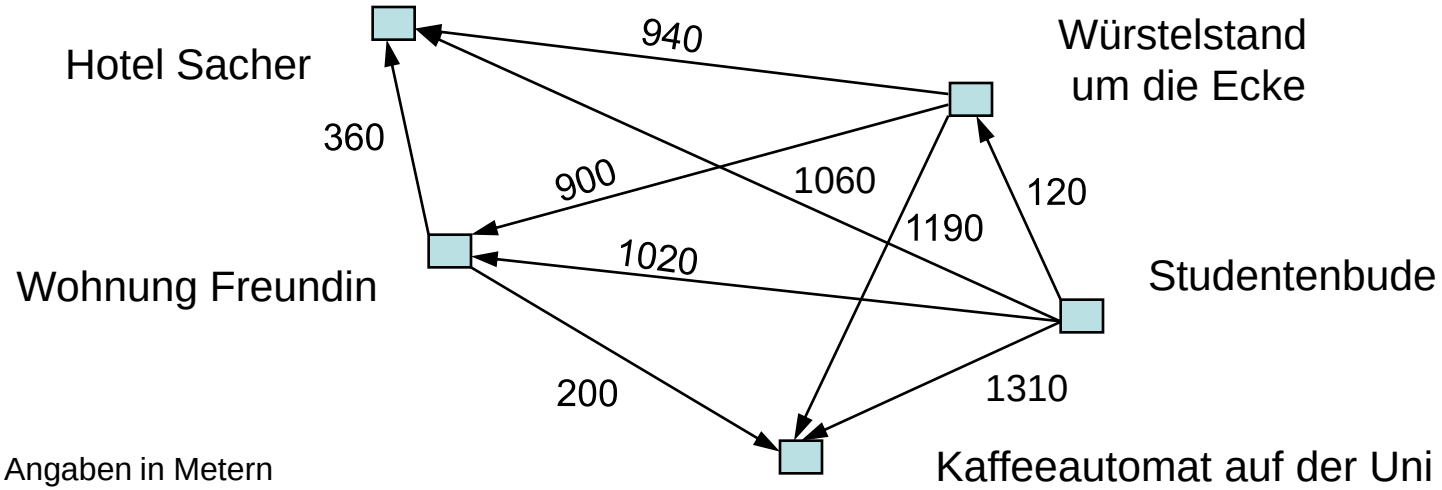
Angaben in Metern

Floyd Algorithmus D(21)

In den folgenden Folien werden nur noch Schritte angezeigt, die unmittelbar die Adjazenzmatrix bzw. den Graphen ändern.

	SB	WS	KA	WF	HS
SB	∞	120	1310	1020	1060
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	1
Variable i	0
Variable j	4
newPl	1060



Angaben in Metern

Floyd Algorithmus D(22)

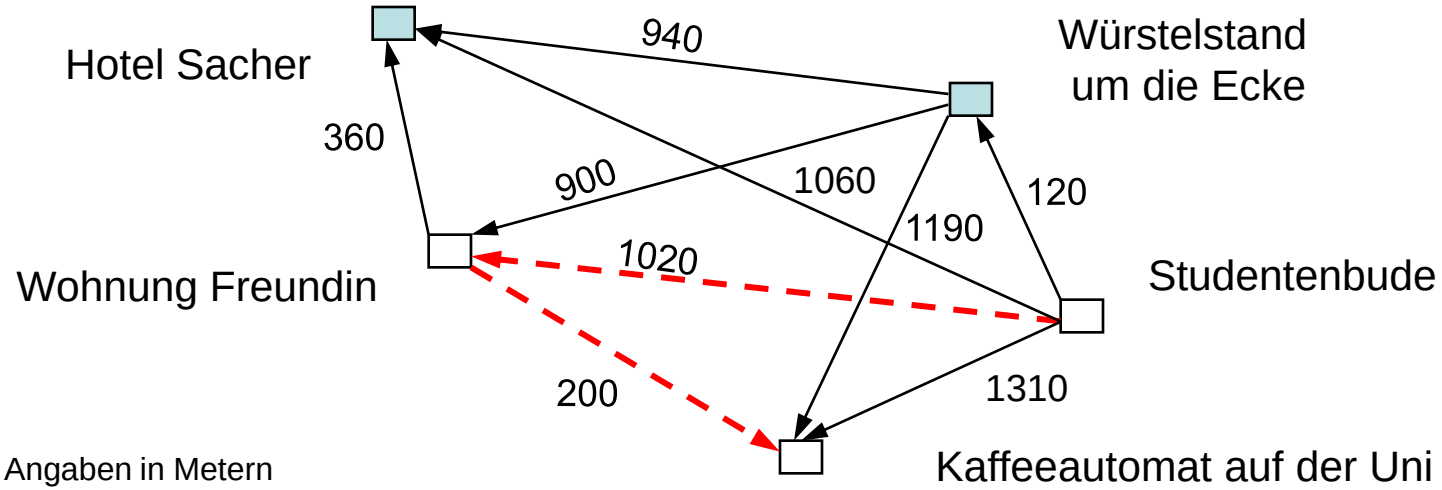
FOR j := 0 TO n - 1 DO

int newPathlength = adjm[i,k] + adjm[k,j];

IF newPathlength < adjm[i,j] THEN

	SB	WS	KA	WF	HS
SB	∞	**120**	**1310**	**1020**	**1060**
WS	∞	∞	**1190**	**900**	**940**
KA	∞	∞	∞	∞	∞
WF	∞	∞	**200**	∞	**360**
HS	∞	∞	∞	∞	∞

	DL
Variable k	**3**
Variable i	**0**
Variable j	**2**
newPl	**1220**



Angaben in Metern

Floyd Algorithmus D(23)

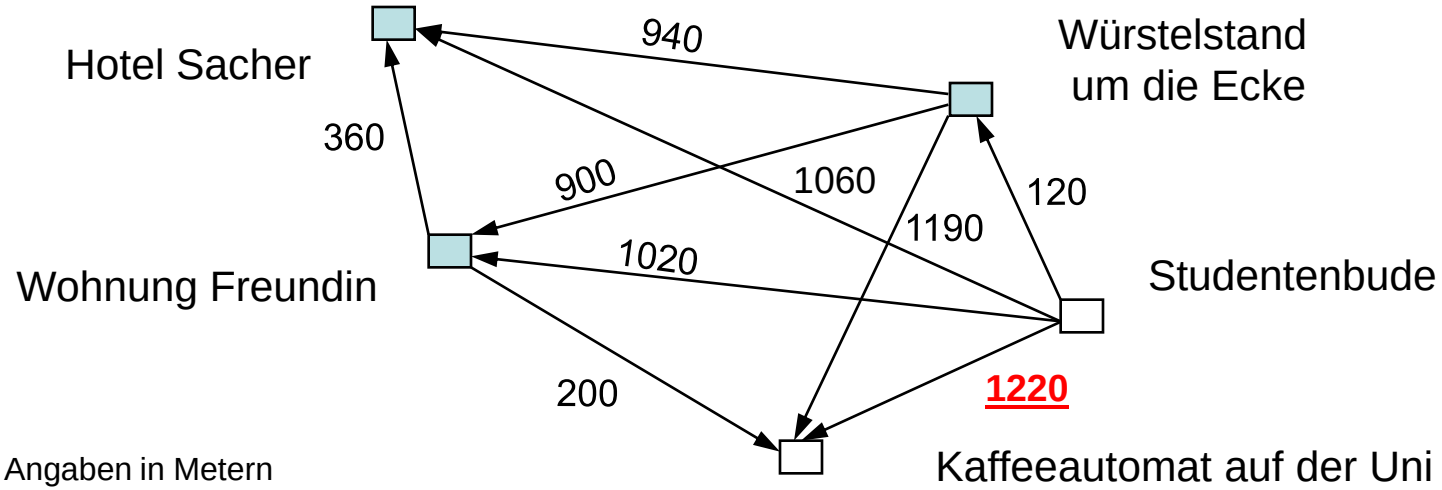
IF newPathlength < adjm[i,j] THEN

adjm[i,j] = newPathlength;

END;

	SB	WS	KA	WF	HS
SB	∞	120	1220	1020	1060
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	3
Variable i	0
Variable j	2
newPl	1220



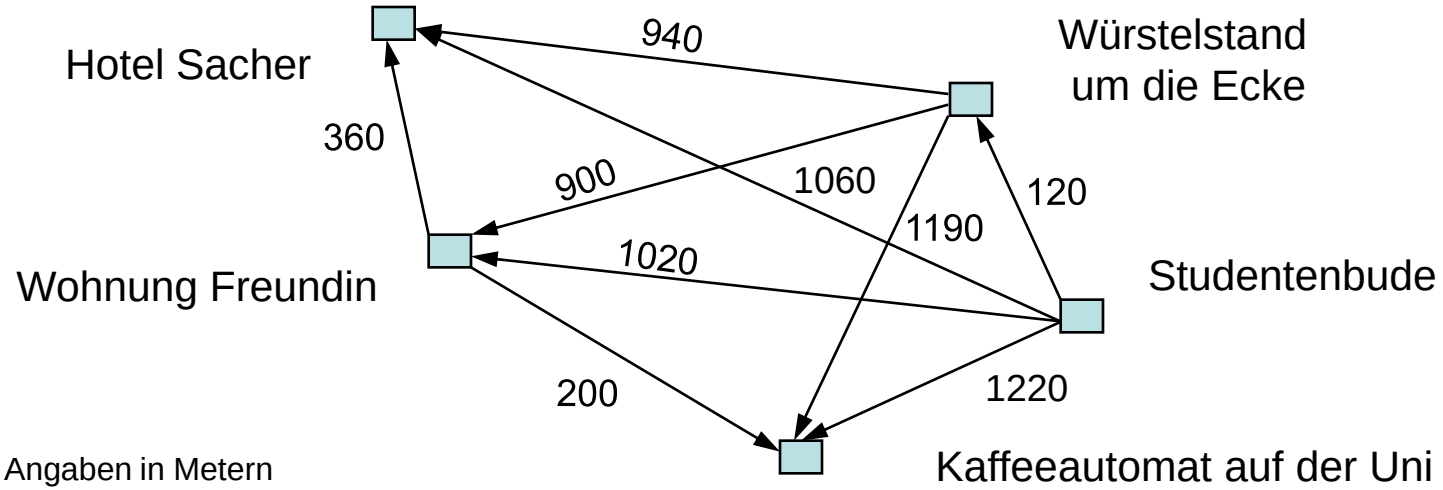
Angaben in Metern

Floyd Algorithmus D(24)

READY

	SB	WS	KA	WF	HS
SB	∞	120	1220	1020	1060
WS	∞	∞	1190	900	940
KA	∞	∞	∞	∞	∞
WF	∞	∞	200	∞	360
HS	∞	∞	∞	∞	∞

	DL
Variable k	4
Variable i	4
Variable j	4
newPl	∞



Angaben in Metern