Bestimmung kürzester und längster Wege in Graphen

Geht man von den bereits vorgestellten Anwendungsbeispielen aus, so wird es häufig wichtig sein, z.B. den kürzesten Transportweg herauszufinden, der in einem Graphen, der alle möglichen Transportwege zwischen Lieferfirmen aufzeigt, vorhanden ist. Dabei sollen die entsprechenden Weglängen (als Kantenmarkierung im bewerteten Graphen) berücksichtigt werden.
Das Problem: "Kürzeste Wege in Graphen", ist eines der grundlegenden algorithmischen Problemstellungen in der Graphentheorie, das in vielen Anwendungsbereichen auftritt und häufig als Teilproblem in anderen Problemstellungen auftritt. Viele Probleme lassen sich auch direkt als "Problem kürzester Wege" umformulieren. Dabei gibt es eine Reihe von Varianten. Wir wollen hier folgende Problemstellungen ansprechen:

- Aufsuchen der kürzesten Wege von einem vorgegebenen Knoten aus zu allen übrigen Knoten des Graphen:
  "single source shortest paths"
- Aufsuchen der kürzesten Wege zwischen je zwei Knoten in einem Graphen:
  "all pairs shortest paths"

Andere Fragestellungen wären auch die Frage nach kürzesten Wegen in Digraphen (gerichteten Graphen), hier muss jeweils die "Durchlaufrichtung" berücksichtigt werden.
Andererseits kann auch die Frage nach den längsten Wegen in einem Graphen gestellt werden, z.B. zeitlich gesehen in einem Projektablauf oder man fragt nach der Länge von Dienst- und Urlaubswegen.

Bem: Die Bestimmung längster Wege ist nur in einem gerichteten Graphen ohne Schleifen (Schlingen) und Zyklen sinnvoll, da man, wie leicht einzusehen, durch wiederholtes Durchlaufen einer Schleife oder eines Kreises immer einen beliebig langen Weg finden kann. Man kann zeigen, dass die Bestimmung längster Wege in einem Graphen mit den Algorithmen für kürzeste Wege erfolgen kann. Dazu ist eine leichte Modifizierung notwendig, auf die hier nicht eingegangen wird.

Bem: Die Bestimmung von kürzesten (oder auch längsten) Wegen in unbewerteten Graphen, bei dem die Länge eines Weges durch die Anzahl der durchlaufenen Kanten gemessen wird, kann immer auf die Ermittlung kürzester oder längster Wege in einem bewerteten Graphen mit Kantenbewertung 1 zurückgeführt werden.

Gegenstand der weiteren Vorlesung soll die Bestimmung der kürzesten Wege in einem Graphen sein, was für eine Einführung in das Gebiet der Graphentheorie auch ausreichend ist.

Markierungsalgorithmus zur Bestimmung kürzester Wege in Graphen: Algorithmus von Dijkstra

Vorbem.: Der Algorithmus von Dijkstra gehört zu den sogenannten Greedy-Algorithmen.
Greedy-Algorithmen bilden in der Informatik eine spezielle Klasse von Algorithmen. Greedy ist das englische Wort für gierig, diese Algorithmen zeichnen sich dadurch aus, dass sie schrittweise immer denjenigen nachfolgenden Zustand auswählen, der in dem betreffenden Schritt das beste Ergebnis liefert, auf dem dann aufgebaut wird.

Im Jahr 1959 veröffentlichte Edsger W. Dijkstra (1930-2002) diesen Algorithmus.

Der Algorithmus von Dijkstra liefert eine Lösung zum Problem: Auffinden kürzestes Wege von einem Knoten aus zu allen übrigen (single source shortest paths)

Wann ist dieses Problem überhaupt lösbar? Wenn ein zusammenhängender, bewerteter, schlichter Graph vorliegt, in dem die Kantenbewertungen alle nicht negativ sind, dann ist der Algorithmus anzuwenden. Wenn der Graph aus mehreren Zusammenhangskomponenten besteht, dann liefert der Algorithmus keine Lösung.

Grundidee: man wähle immer diejenige Kante aus, die vom Startknoten aus immer den kürzesten Abschnitt (Summe der Kantenbewertungen bis zum betreffenden Schritt minimal) liefert.

Wie das nun genau durchzuführen ist, soll im folgenden kurz theoretisch beschrieben werden und danach an einem konkreten Beispiel durchgeführt werden.

Allgemeine Kurzbeschreibung des Algorithmus:

Gegeben: bewerteter Graph $G = (M, K, v, f)$
mit $f(k) \geq 0$ für alle $k \in K$ (alle Kanten mit positiver Bewertung)

Bezeichnung: $D(x_i)$: Bewertung des Distanz vom Startknoten x_0 zum Knoten x_i

(1) Markiere x_0 (Startknoten) mit $D(x_0) = 0$
(x_0 hat zu sich selbst die Entfernung 0)
Setze $i = 1$, $M_1 := \{x_0\}$ weiter mit (2)

(2) Sei M^* die Menge aller Knoten x_s der Kanten (x_r, x_s), deren Knoten x_r in M_i liegt, d.h.
$M^* = \{x_s \in M \setminus M_i \mid$ es gibt eine Kante (x_r, x_s) mit $x_r \in M_i\}$
Falls $M^* = \emptyset$ weiter mit (6), falls $M^* \neq \emptyset$ weiter mit (3)

Übersetzt für x_0 und den "ersten zweiten" Schritt:
Man betrachtet alle zu x_0 benachbarten Knoten und deren Kantenweste. Es wird dann der Knoten bewertet, der von x_0 über die verbindende Kante den kleinsten Kantenwert hat. Man markiert diese Kante und bewertet den Knoten mit $D(x_i) = f(x_0, x_i)$ (Kantenbewertung)

(3) Bestimme zu jedem $x_s \in M^*$ eine vorläufige Markierung $D^*(x_s) = \min(D(x_r) + f(x_r, x_s))$
(Bem.: Es wird zu jedem neuen Nachbarknoten die kürzeste Entfernung vom Startknoten aus über bereits markierte Kanten ermittelt)
weiter mit (4)

(4) Ermittle aus den unter (3) gelisteten kürzesten Entfernungen den Knoten $x_s^* := x_i$ mit $D(x_i) = D^*(x_i)$, es gilt $D^*(x_s^* := \min(D^*(x_s)))$
Markiere die Kante (x_r, x_i)
weiter mit (5)

(5) Setze $M_{i+1} := M_i \cup \{x_i\}$ und $i := i+1$
weiter mit (2)

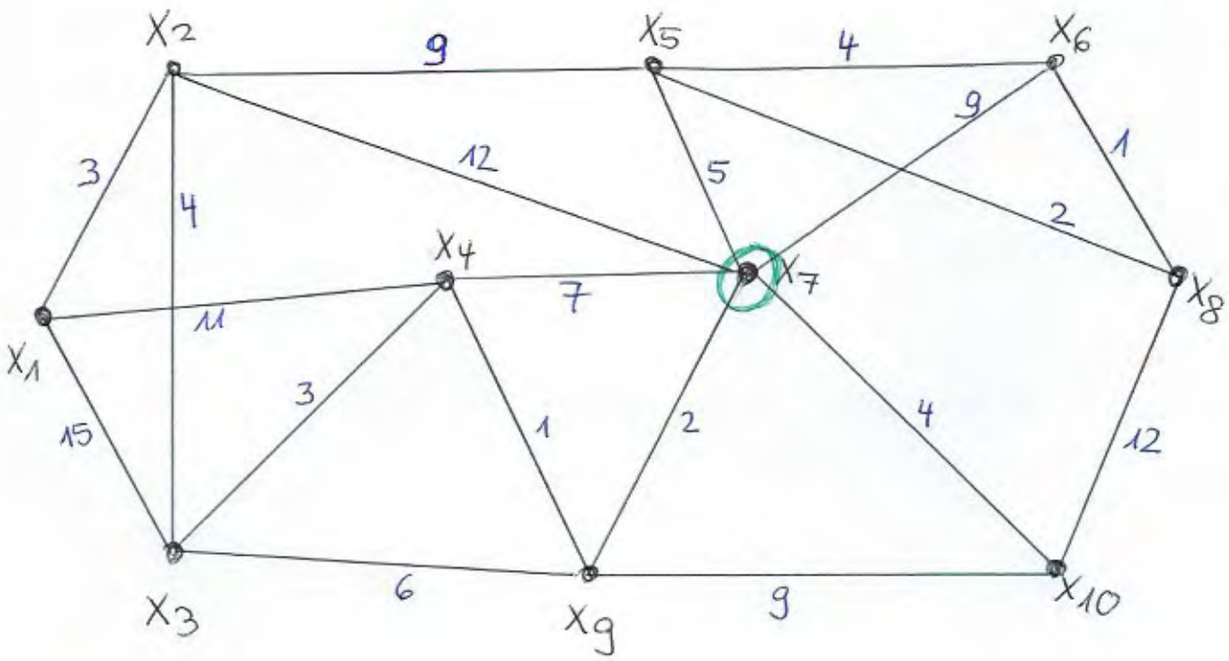
(6) Ende, wenn alle Knoten markiert sind.

Diese theoretische Beschreibung des Dijkstra-Algorithmus wird viel verständlicher, wenn man diesen an einem konkreten Beispiel durchführt.

Dies soll im Folgenden geschehen:

Gegeben ist folgender Graph mit 10 Knoten und den dort angegebenen Kantenbewertungen ("Entfernungen")

Es sollen die kürzesten Wege zu allen anderen Knoten des Graphen von x_7 bestimmt werden.



Schritt (1): $D(X_7) = 0 \quad M_1 := \{X_7\}$

Iteration 1: (2) $M^* = \{X_2, X_4, X_5, X_6, X_9, X_{10}\}$ Menge der zu X_7 benachbarten Knoten

(3) Ermittlung der Entfernungen $D^*(X_S)$, $X_S \in M^*$

X_S	X_2	X_4	X_5	X_6	X_9	X_{10}
$D^*(X_S)$	12	7	5	9	2	10

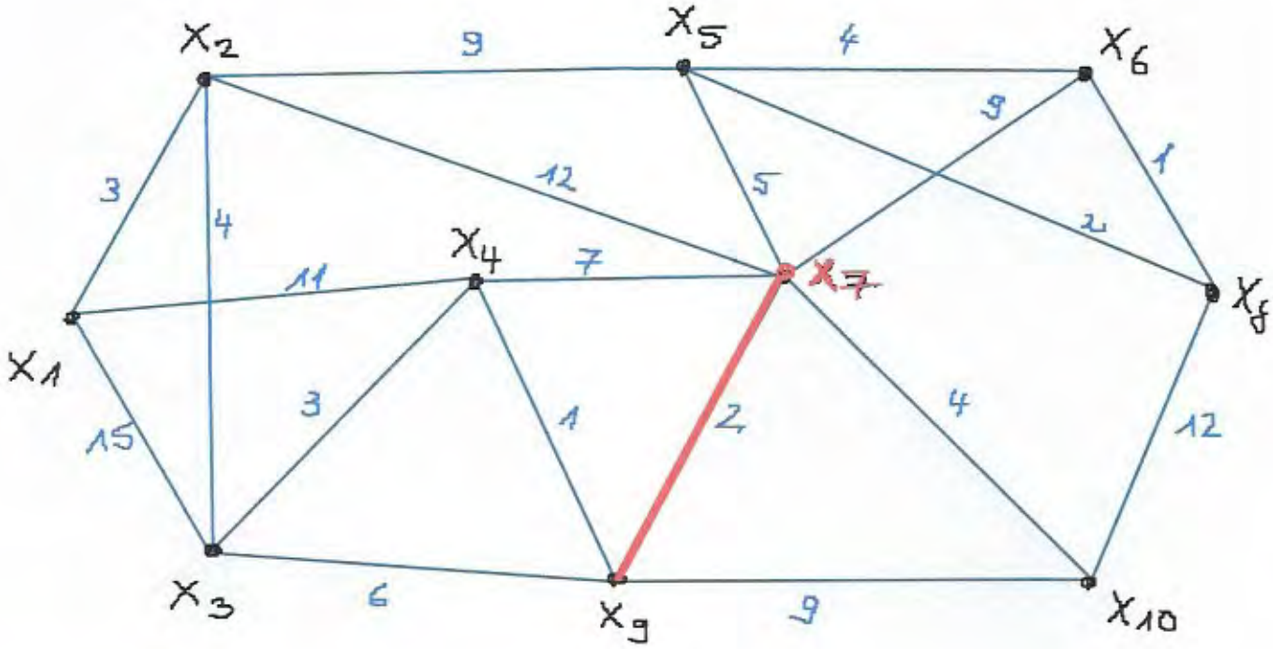
Die kürzeste Entfernung von X_7 ist die nach X_9

Setze $D(X_9) = 2$

(4) Markiere im Graphen die Kante (X_7, X_9)

(5) $M_2 := M_1 \cup \{X_9\} = \{X_7, X_9\}$

Graph nach der ersten Iteration:



Iteration 2:

(2) $M^* = \{X_2, X_3, X_4, X_5, X_6, X_{10}\}$

Menge der zu x_7 und x_9 benachbarten Knoten.

(3) Ermittlung der Entfernungen $D(x_s)$, $x_s \in M^*$ und zwar direkt oder über x_9 erreichbar, und zwar immer die kürzeste. Es sind folgende Fälle zu betrachten:

- $D^*(x_s) = f(x_7, x_s)$ falls x_s nur zu x_7 benachbart ist, also x_2, x_5, x_6 (x_4 z.B. kann auch über x_9 erreicht werden)
- $D^*(x_s) = D(x_9) + f(x_9, x_s)$, falls x_s nur zu x_9 benachbart ist.
- $D^*(x_s) = \min\left(f(x_7, x_s); D(x_9) + f(x_9, x_s)\right)$, falls x_s zu x_7 und x_9 benachbart ist, also bei x_4 und x_{10}

Für x_4 ist $D^*(x_4) = \min\left(f(x_7, x_4);\ D(x_9) + f(x_9, x_4) \right)$

$= \min(7, 3) = 3$

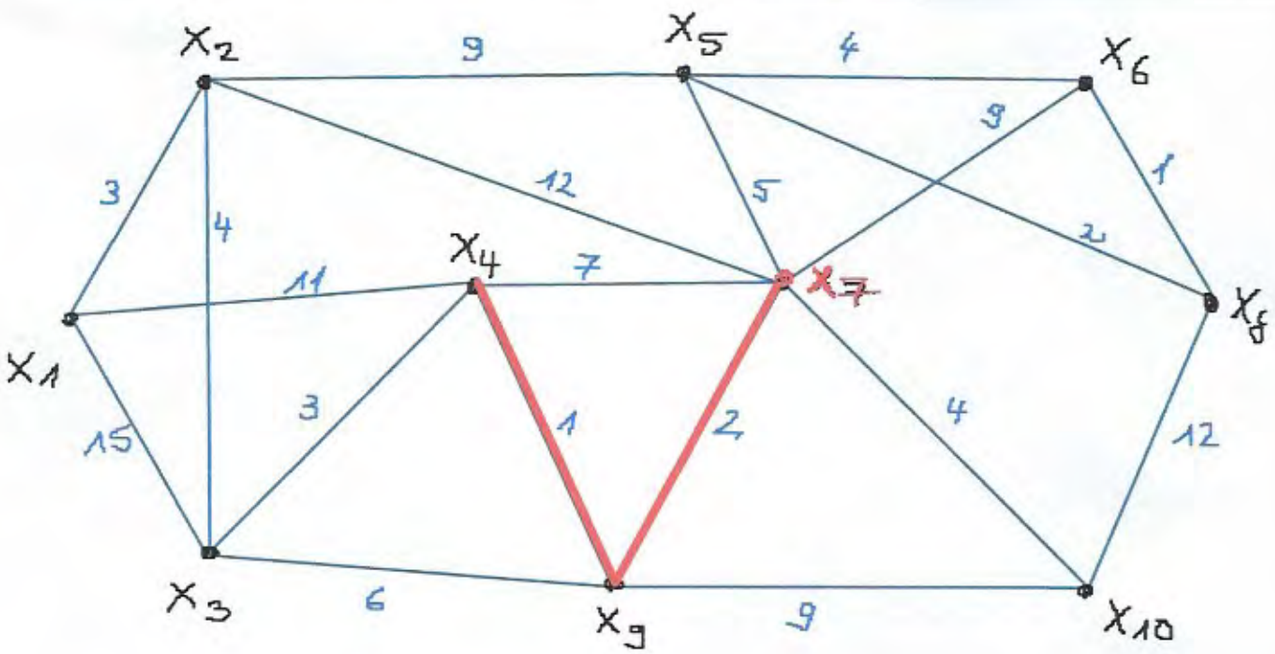
x_s	x_2	x_3	x_4	x_5	x_6	x_{10}
$D^*(x_s)$	12	8 (2+6)	3	5	9	4

(4) Von allen aus M^* direkt oder über x_9 erreichbare Knoten von x_7 aus hat x_4 die geringste Entfernung.

Setze $D(x_4) = 3$

Markiere die Kante (x_9, x_4)

(5) $M_3 = M_2 \cup \{x_4\} = \{x_7, x_9, x_4\}$



Iteration 3

(2) $M^* = \{x_1, x_2, x_3, x_5, x_6, x_{10}\}$

alle zu x_7, x_9 und x_4 benachbarten Knoten.

(3) Für die Knoten $x_s \in M^*$ sind die direkt von x_7 und die über x_9 und x_4 ermittelbaren kürzesten Entfernungen zu suchen:

Wir führen das exemplarisch für den Knoten x_3 durch: 1. Möglichkeit über x_4: $D(x_4) + f(x_4, x_3) = 3+3 = 6$

2. Möglichkeit über x_9: $D(x_9) + f(x_9, x_3) = 2+6 = 8$

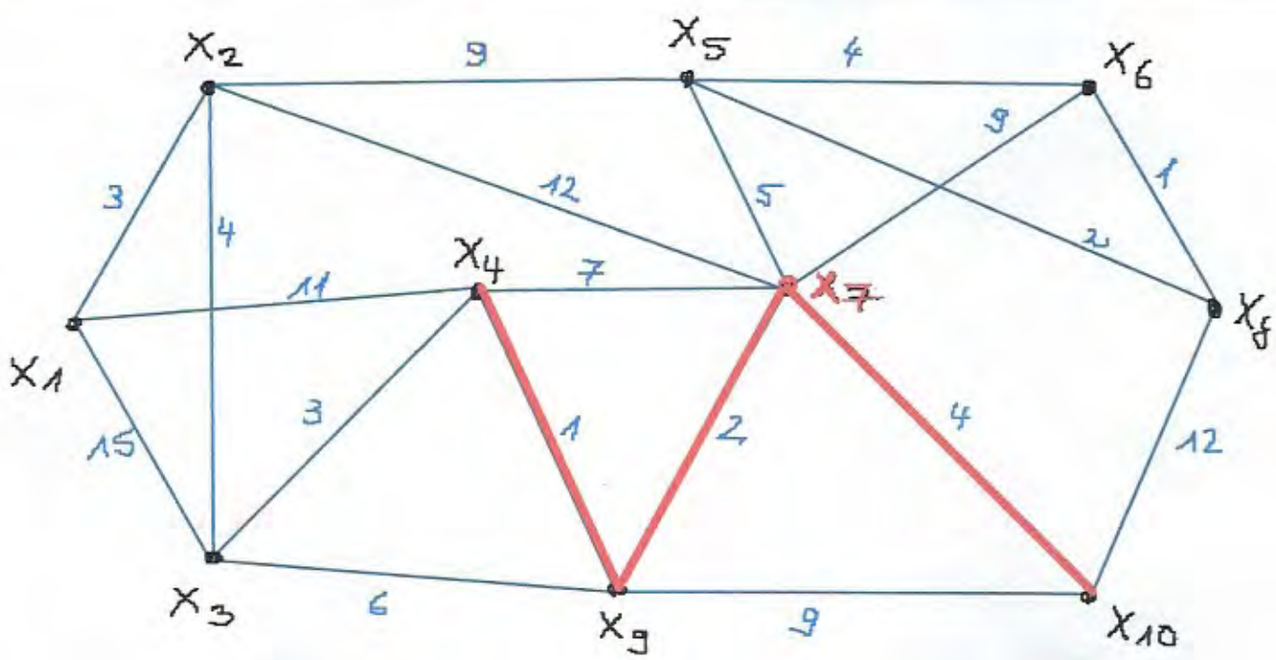
An diesem Beispiel sieht man, dass es wichtig ist, immer sämtliche möglichen Wege in aufsteigender Reihenfolge zu betrachten, da manchmal ein Weg über einen "Umweg" kürzer ist als der direkte Weg.

Wie für x_3 werden auch für $x_1, x_2, x_5, x_6, x_{10}$ die Entfernungen bewertet:

x_s	x_1	x_2	x_3	x_5	x_6	x_{10}
$D^*(x_s)$	14	12	6	5	9	4
	2+1+11		2+1+3			

(4) Es ist $D(x_{10}) = 4$ Markiere die Kante (x_7, x_{10})

(5) $M_4 = M_3 \cup \{x_{10}\} = \{x_7, x_9, x_4, x_{10}\}$



Iteration 4

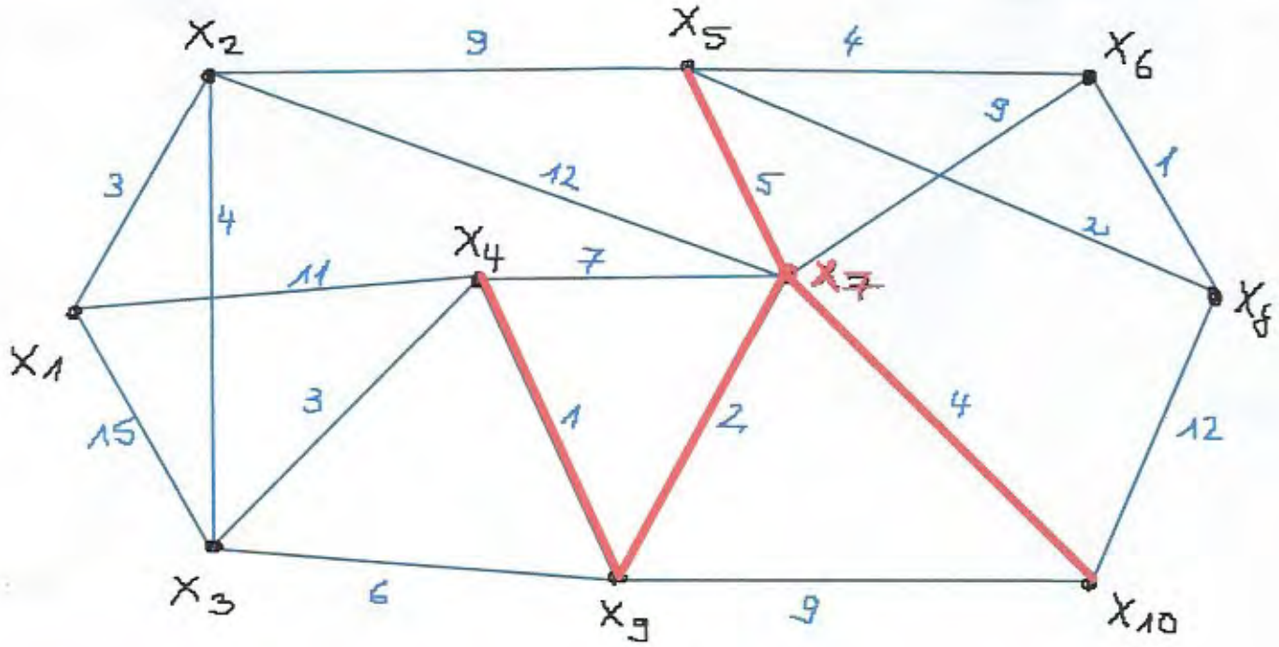
(2) $M^* = \{x_1, x_2, x_3, x_5, x_6, x_8\}$ benachbarte Knoten zu allen bereits bewerteten

(3) Für die Knoten $x_s \in M^*$ wird die direkt von x_7 und/oder über die anderen Knoten aus M_4 ermittelten kürzesten Entfernungen zu notieren:

x_s	x_1	x_2	x_3	x_5	x_6	x_8
$D^*(x_s)$	14 (1+2+11)	12	6 (2+1+3)	5	9	16 (4+12)

(4) Es ist $D(x_5) = 5$ Markiere die Kante (x_7, x_5)

(5) $M_5 = M_4 \cup \{x_5\} = \{x_7, x_9, x_4, x_{10}, x_5\}$



Iteration 5

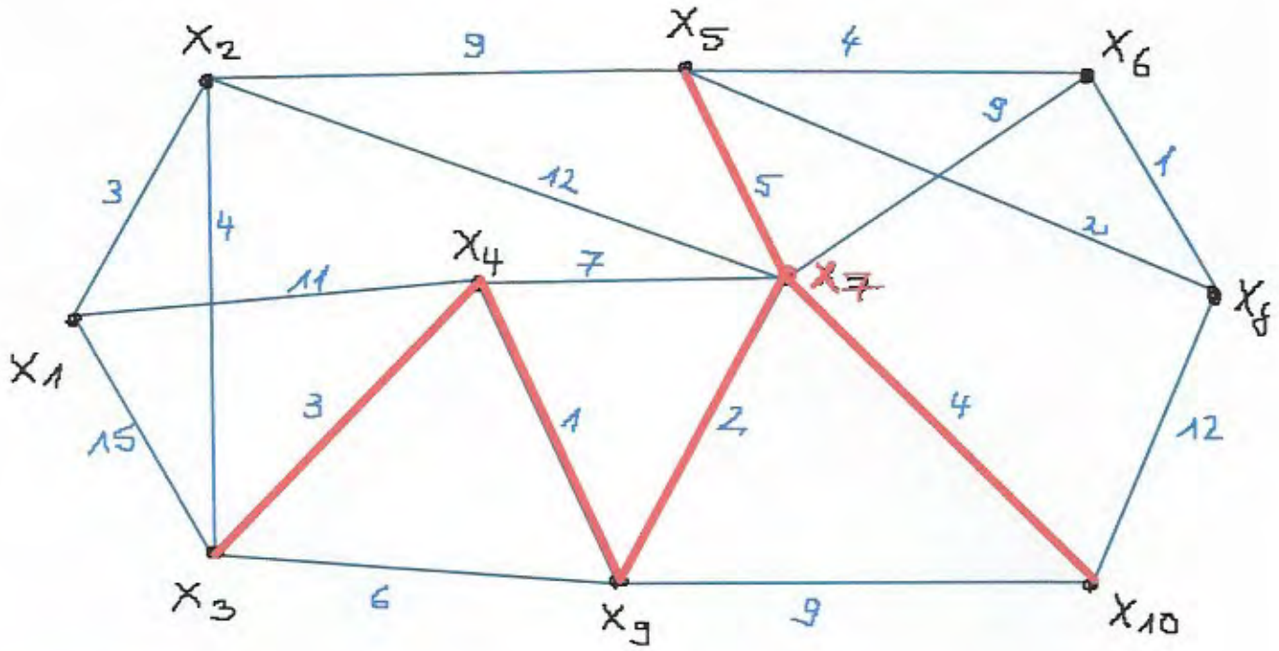
(2) $M^* = \{x_1, x_2, x_3, x_6, x_8\}$ alle benachbarten Knoten zu M_5

(3)

x_s	x_1	x_2	x_3	x_6	x_8
$D^*(x_s)$	14 (2+1+11)	12	6 (2+1+3)	9	7 (5+2)

(4) $D(x_3) = 6$ Markiere die Kante (x_4, x_3)

(5) $M_6 = M_5 \cup \{x_3\} = \{x_7, x_9, x_4, x_{10}, x_5, x_3\}$



Iteration 6

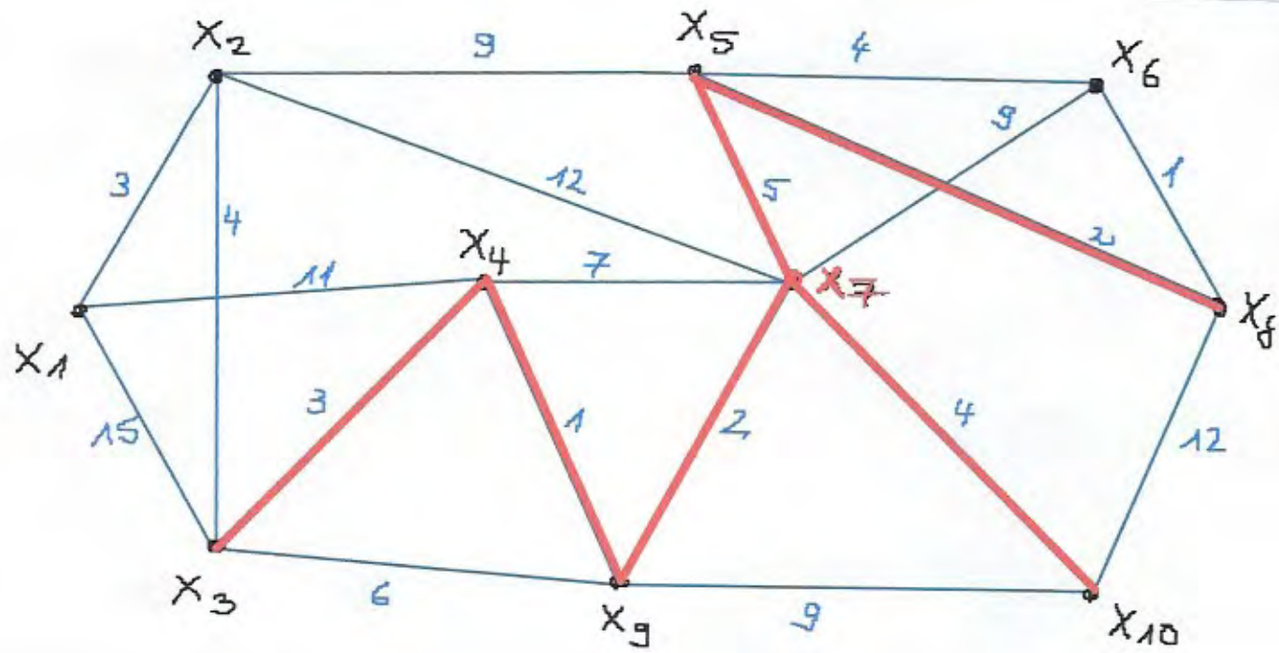
(2) $M^* = \{X_1, X_2, X_6, X_8\}$ alle zu M_6 benachbarten Knoten

(3)

X_s	X_1	X_2	X_6	X_8
$D^*(X_s)$	14	10	9	7

2+1+3+4

(4) $D(X_8) = 7$ Markiere die Kante (X_5, X_8) $M_7 = M_6 \cup \{X_8\}$
$= \{X_3, X_4, X_5, X_7, X_8, X_9, X_{10}\}$



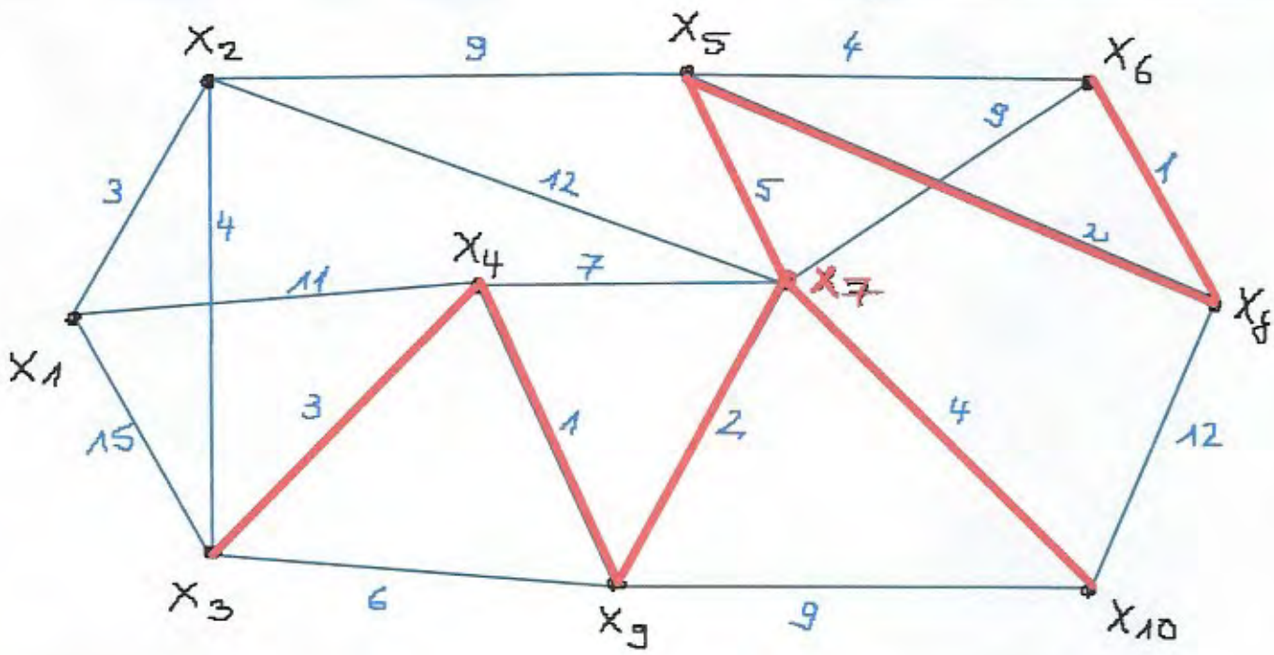
Iteration 7

(2) $M^* = \{x_1, x_2, x_6\}$ alle zu M_7 benachbarten Knoten

(3)

x_S	x_1	x_2	x_6
$D^*(x_S)$	14	10	8
	2+11	2+3+4	5+2+1

(4) $D(x_6) = 8$ Markiere die Kante (x_8, x_6) $M_8 = M_7 \cup \{x_6\}$



Iteration 8

(2) $M^* = \{x_1, x_2\}$ alle zu M_8 benachbarten Knoten

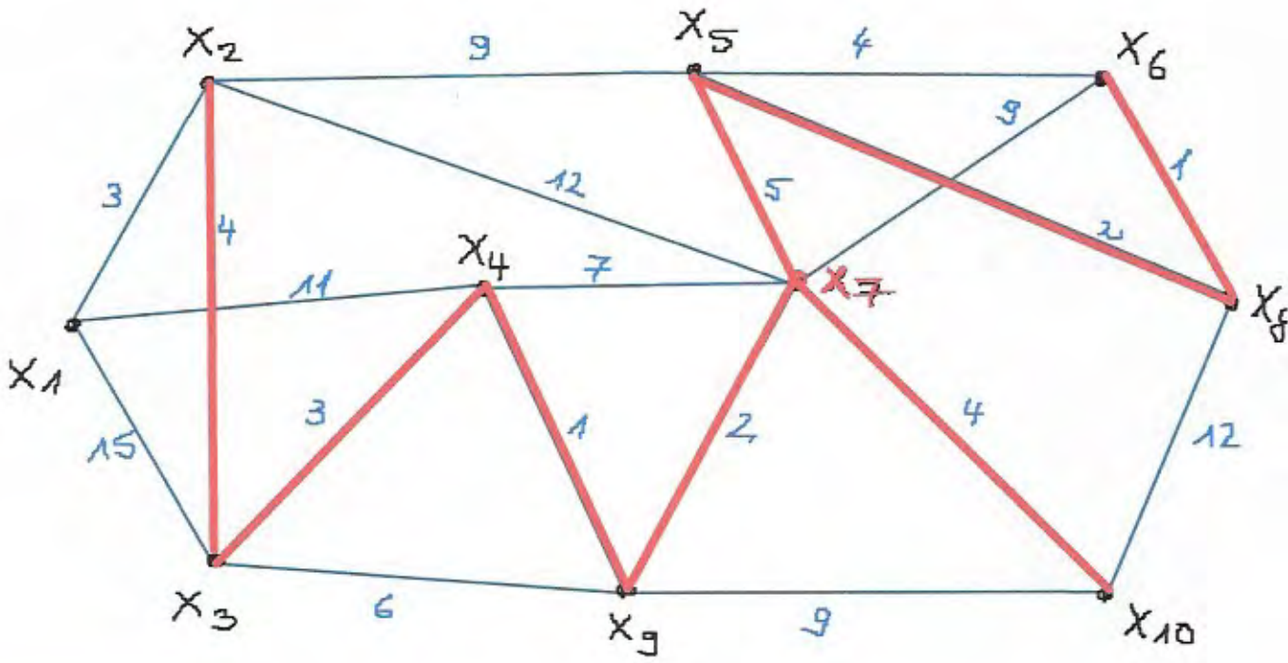
(3)

x_S	x_1	x_2
$D^*(x_S)$	14	10
	2+1+11	2+1+4

(4) $D(x_2) = 10$

Markiere die Kante (x_3, x_2)

$M_9 = M_8 \cup \{x_2\} = \{x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$



Iteration 9

(2) $M^* = \{x_1\}$ (3)

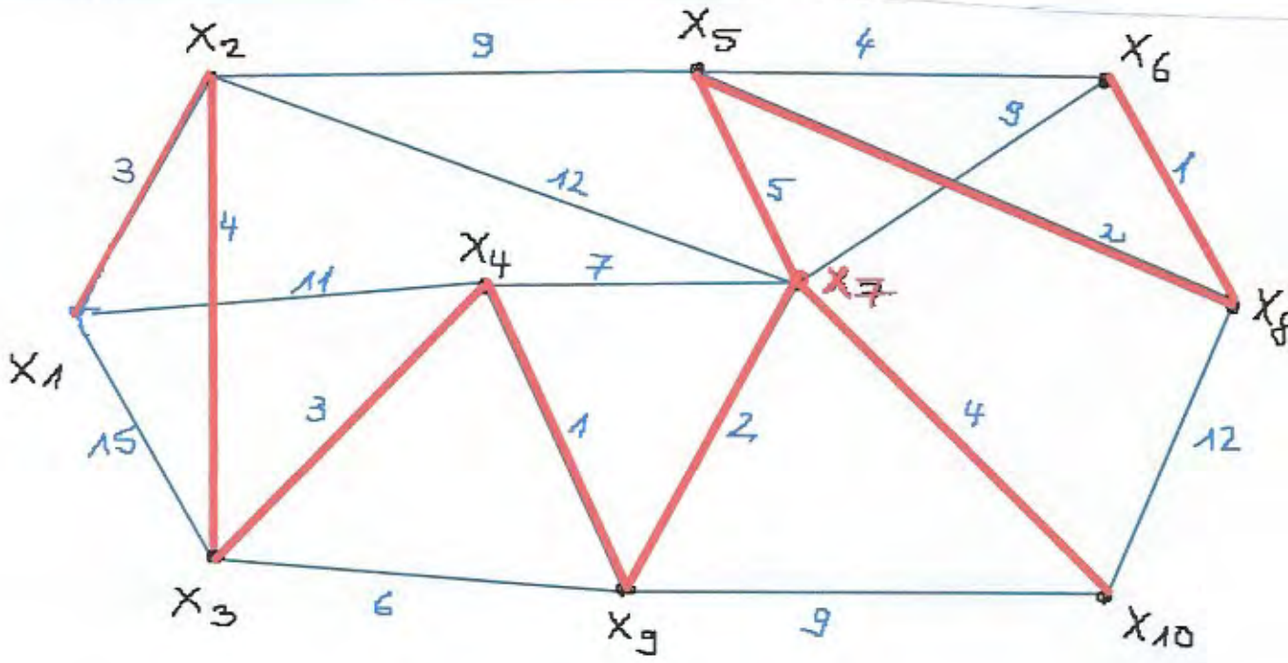
x_s	x_1
$D^*(x_s)$	13

$2+1+3+4+3$

(4) $D(x_1) = 13$

Markiere die Kante (x_2, x_1)

Damit sind alle Knoten markiert und der Algorithmus zu Ende.



Damit sind die kürzesten Wege vom Startknoten x_7 aus zu allen anderen Knoten des Graphen ermittelt.
Abschliessend kann man nochmals alle Bewertungen darstellen:

$D(x_1) = 13$ (Iteration 9)
$D(x_2) = 10$ (Iteration 8)
$D(x_3) = 6$ (Iteration 5)
$D(x_4) = 3$ (Iteration 2)
$D(x_5) = 5$ (Iteration 4)
$D(x_6) = 8$ (Iteration 7)
$D(x_8) = 7$ (Iteration 6)
$D(x_9) = 2$ (Iteration 1)
$D(x_{10}) = 4$ (Iteration 3)

x_7 Startknoten $D(x_7) = 0$

Es wurde hier anhand des Beispiels der ausführlichste Weg beschrieben, von einem Startknoten aus die kürzesten Wege zu allen anderen Knoten des Graphen zu bestimmen. Häufig wird auch nur die Frage nach dem kürzesten Weg von einem Startknoten zu einem anderen Knoten des Graphen gestellt werden. Der Algorithmus ist dann beendet, wenn genau der gefragte Knoten markiert wurde.

Der Algorithmus von Dijkstra ist nicht nur auf ungerichtete Graphen anwendbar, sondern auch auf Digraphen.
Frage: Worauf muss hier geachtet werden?
Antwort: Wege, die betrachtet werden dürfen nur in Pfeilrichtung durchlaufen werden.
(Stichwort: Einbahnstraße)

Zusammenfassung:

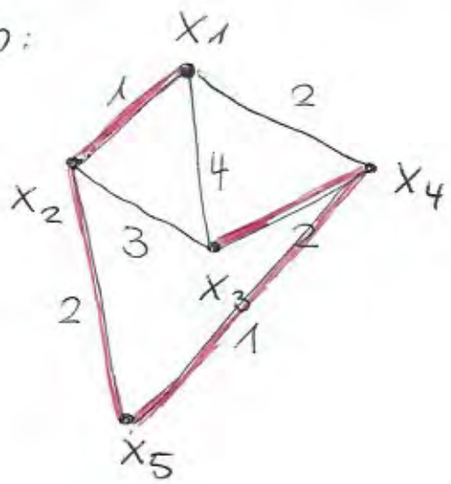
Mit dem Dijkstra-Algorithmus hat man die Regel, dass man immer derjenigen Kante vom Startknoten aus folgt, die den kürzesten Streckenabschnitt darstellt. Erst wenn alle kürzeren Abschnitte berücksichtigt wurden, werden andere Kanten verfolgt. Das garantiert, dass bei Erreichen eines Knotens kein kürzerer Pfad existieren kann.

Frage: Wie würden Sie den Unterschied zwischen dem Ergebnis des Algorithmus von Kruskal und dem Ergebnis des Algorithmus von Dijkstra beschreiben?

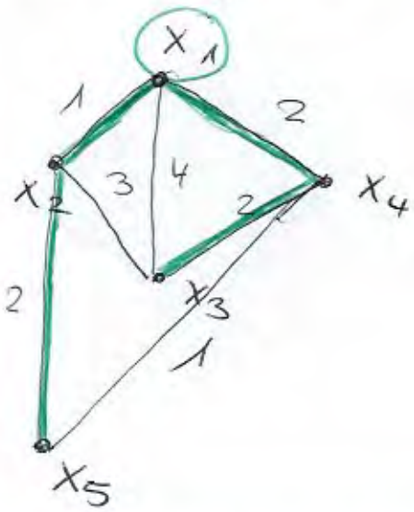
Antwort: Der Algorithmus von Kruskal liefert den minimalen Spannbaum (MST) für einen bestimmten Graphen, also einen Baum, der alle Knoten miteinander verbindet, wobei die Summe der Kantenbewertungen ein Minimum ist.

Der Algorithmus von Dijkstra ermittelt jeweils die minimale Summe (Abstand) der Kantenbewertungen von einem vorgegebenen Startknoten zu allen Knoten des Graphen. Man erhält also jeweils einen Baum mit minimaler Kantenbewertung vom Startknoten zu einem bestimmten Knoten des Graphen.

Bp:



MST
mit dem Algorithmus von Kruskal ermittelt



kürzeste Wege von X_1 aus mit dem Algorithmus von Dijkstra ermittelt

Ausblick auf weitere Anwendungsbereiche der Graphentheorie

Auch folgendes Problem führt auf einen Graphen:

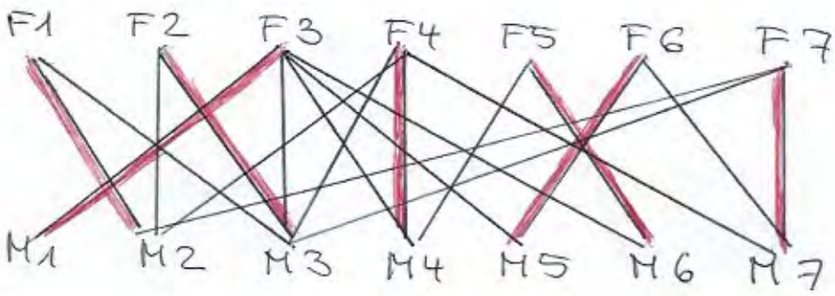
Bp.: Partnervermittlung

7 Frauen
7 Männer

jeder erstellt eine Liste, mit wem er sich vorstellen könnte, eine Partnerschaft einzugehen
Und es sollen "Paare" gebildet werden aus jeweils einer Frau und einem Mann

Bp. für eine Liste:

	Wunschkandidat
F1	M2 M3
F2	M2 M3
F3	M1 M3 M4 M5 M6
F4	M2 M3 M4 M7
F5	M4 M6
F6	M5 M7
F7	M2 M3 M7



Man spricht von einem **bipartiten Graphen** (Zuordnung zwischen zwei Knotenmengen (Frauen, Männer)

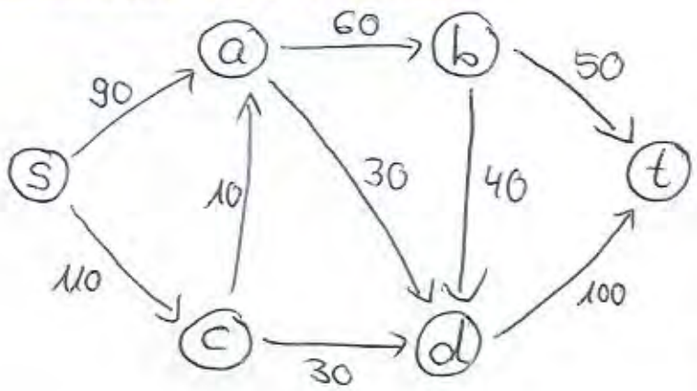
rote Kanten: **perfektes Matching**, jede Frau hat einen Partner aus ihrer Wunschliste

Def. (Matching) (Zuordnung, Paarung)

Ein Matching in einem bipartiten Graphen (Graph mit zwei Knotenmengen, die keinen gemeinsamen Knoten haben) ist eine **Menge von Kanten**, die jeweils einen Knoten aus der einen Menge mit einem Knoten aus der anderen Menge verbinden. Die **Kanten** sind **paarweise** verschieden, der Knotengrad ist jeweils 1.

Ein **maximales Matching** ist ein Matching mit der größtmöglichen Anzahl von Kanten.
Ein **perfektes Matching** ist ein Matching, das alle Knoten des Graphen überdeckt.
Die Theorie zum Auffinden von maximalen (oder auch perfekten) Matchings ist sehr umfangreich. Auch hier gibt es **Algorithmen** für das Auffinden von maximalen bzw. perfekten Matchings, die Adjazenzmatrizen spielen dabei auch eine große Rolle.
Darauf soll hier nicht weiter eingegangen werden.

Und zum Schluss:



dieser Graph stellt einen Paketzustellbereich von (s) nach (t) dar, die Kantenbewertungen repräsentieren die Anzahl der Pakete die von einer Stadt in die andere täglich befördert werden können

Durch die Kantenbewertungen werden hier Kapazitäten angegeben, die beim Weiterleiten nicht überschritten werden dürfen.

Wenn pro Tag möglichst viele Pakete von (S) nach (t) transportiert werden sollen, dann sucht man von der Graphentheorie aus betrachtet, einen maximalen Fluss von (S) nach (t)

Auch hier gibt es wieder Algorithmen, die in einem solchen "Netzwerk" einen maximalen Fluss finden.

Auch hier soll nicht weiter auf das Thema eingegangen werden.

Damit ist die Einführung in das Thema Graphentheorie beendet.

Lösung Blatt 10 Aufgabe 13

Start: $D(a) = 0$ $\quad M_1 = \{a\}$

$M^* := \{c, k, e, h\}$ Menge der zu a benachbarten Knoten

Iteration 1:

x_S	c	k	e	h
$D^*(x_S)$	9	7	1	3

Markiere die Kante (a, e)

$D(e) = 1$ $\quad M_2 = \{a, e\}$

Iteration 2: $M^* := \{c, k, d, h, i, m\}$

Markiere die Kante (a, h)

$D(h) = 3$ $\quad M_3 = \{a, e, h\}$

x_S	c	k	d	h	i	m
$D^*(x_S)$	9	7	6	3	8	7

Iteration 3: $M^* = \{c, k, d, m, i\}$

Markiere die Kante (e, d)

$D(d) = 6$ $\quad M_4 = \{a, e, h, d\}$

x_S	c	k	d	m	i
$D^*(x_S)$	9	7	6	7	7

Iteration 4: $M^* := \{c, k, i, m\}$

Markiere die Kanten (a, k), (h, i) und (e, m)

$D(k) = 7$ $\quad D(i) = 7$ $\quad D(m) = 7$

x_S	c	k	i	m
$D^*(x_S)$	9	7	7	7

Iteration 5: $M^* := \{c, f, j, l\}$

Markiere die Kante (d, c)

$D(c) = 8$

x_S	c	f	j	l
$D^*(x_S)$	8	30	9	13

Iteration 6: $M^* := \{b, j, l, f\}$

Markiere die Kante (k, j)

$D(j) = 9$

x_S	b	j	l	f
$D^*(x_S)$	11	9	13	30

Iteration 7: $M^* := \{l, f, g, b\}$

Markiere die Kante (j, g)

$D(g) = 5$

x_S	l	f	g	b
$D^*(x_S)$	13	14	5	11

Iteration 8: $M^* := \{f, b, l\}$

Markiere die Kante (c, b)

$D(b) = 11$

x_S	f	b	l
$D^*(x_S)$	14	11	13

Iteration 9: $M^* := \{l, f\}$

Markiere Kante (k, l)

$D(l) = 13$

x_S	l	f
$D^*(x_S)$	13	14

Iteration 10: $M^* := \{f\}$ $\quad x_S$ f, $D^*(x_S)$ 14

Markiere Kante (j, f)

$D(f) = 14$

Aufgabe 13

Bestimmen Sie mit Hilfe des **Dijkstra-Algorithmus** die kürzesten Wege vom Knoten (a) zu allen anderen Knoten.