Prof. Dr. Ir. Joost-Pieter Katoen
Sebastian Junges, Benjamin Kaminski, David Korzeniewski, Tim Quatmann

Übung 10
– Musterlösung –

Hinweise:

- Die Lösungen müssen bis **Donnerstag, den 5. Juli um 16:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Die Übungsblätter **müssen** in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung abgegeben werden.
- Drucken Sie ggf. digital angefertigte Lösungen aus. Abgaben z.B. per Email sind nicht zulässig.
- Namen und Matrikelnummer sowie die **Nummer der Übungsgruppe** sind auf jedes Blatt der Abgabe zu schreiben. Abgaben, die aus mehreren Blättern bestehen **müssen geheftet bzw. getackert** werden! Die **Gruppennummer muss sich auf der ersten Seite oben links** befinden.
- **Bei Nichtbeachten der obigen Hinweise müssen Sie mit erheblichen Punktabzügen rechnen!**

Aufgabe 1 (Ein Graphalgorithmus): (5 + 20 = 25 Punkte)

Gegeben sei der folgende Algorithmus:

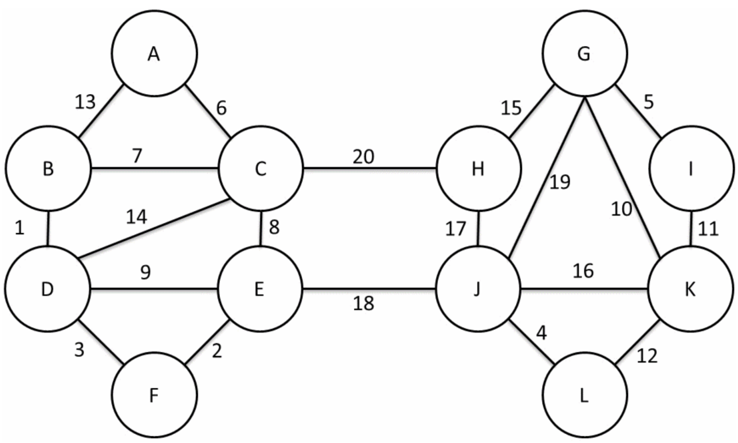
Eingabe: zusammenhängender gewichteter, ungerichteter Graph (V, E, W)

`Erstelle ein Array` E_s `aller Kanten absteigend sortiert nach Kantengewicht`
`for` $i = 0$ `to` $|E| - 1$:
  $e := E_s[i]$
  `Lösche Kante` e `aus` E
  `Falls` (V, E, W) `nicht zusammenhängend`:
    `Füge Kante` e `zu` E `hinzu`

Ausgabe: (V, E)

a) Was berechnet der obige Algorithmus? Begründen Sie ihre Antwort.

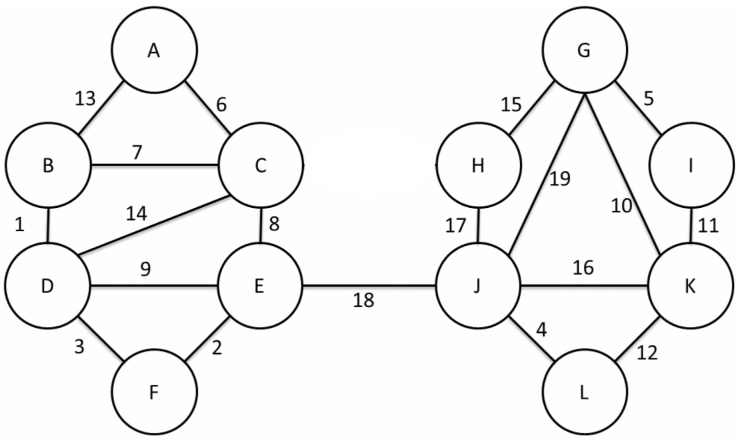
b) Führen Sie den obigen Algorithmus auf dem untenstehenden Graphen aus. Geben Sie den Graphen (V, E) nach jeder Iteration der Schleife an.



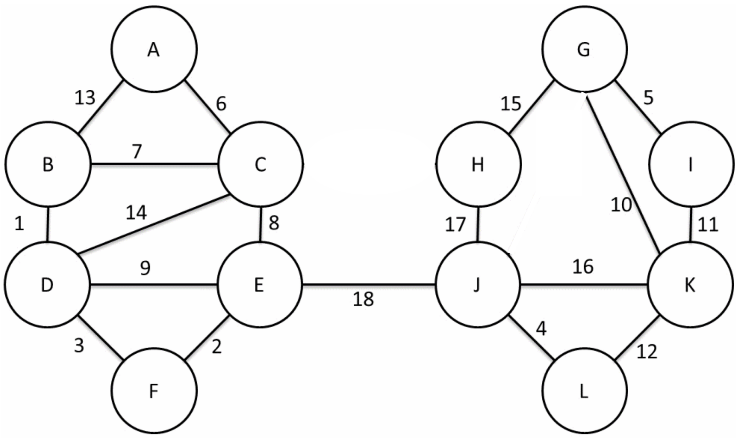
Lösung:

a) Der Algorithmus berechnet einen minimalen Spannbaum. Er geht dabei dual zum Kruskal-Algorithmus vor. Statt in jedem Schritt die nächstbilligste Kante, die keinen Kreis einführt, zum Ergebnis hinzuzufügen wird in jedem Schritt die nächstteuerste Kante, die den Graphen nicht unzusammenhängend macht, gelöscht.

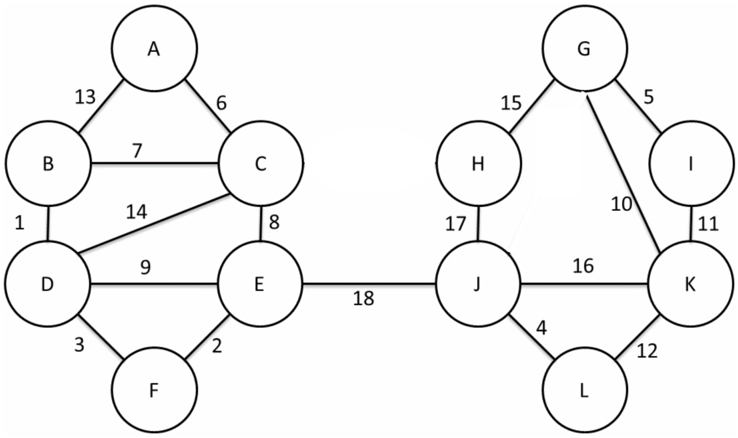
b) Kante 20 wird gelöscht:



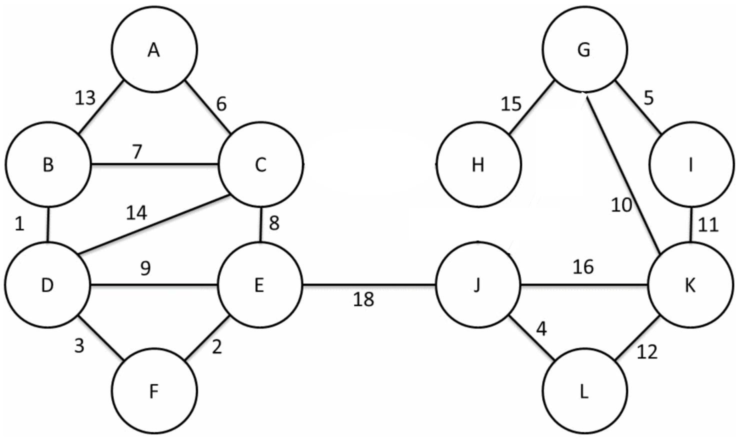
Kante 19 wird gelöscht:



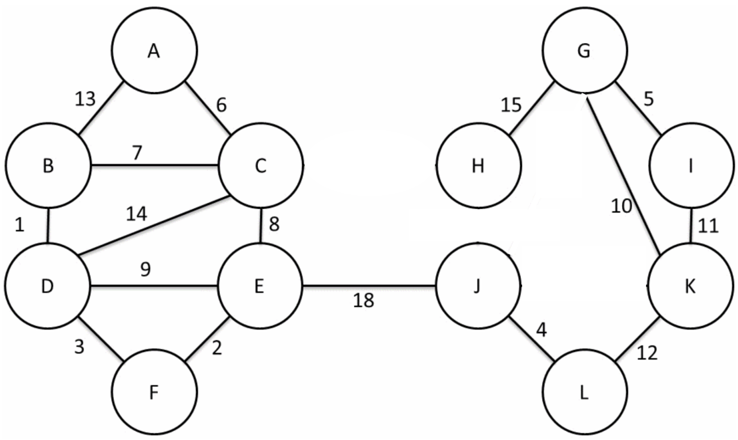
Kante 18 wird *nicht* gelöscht:



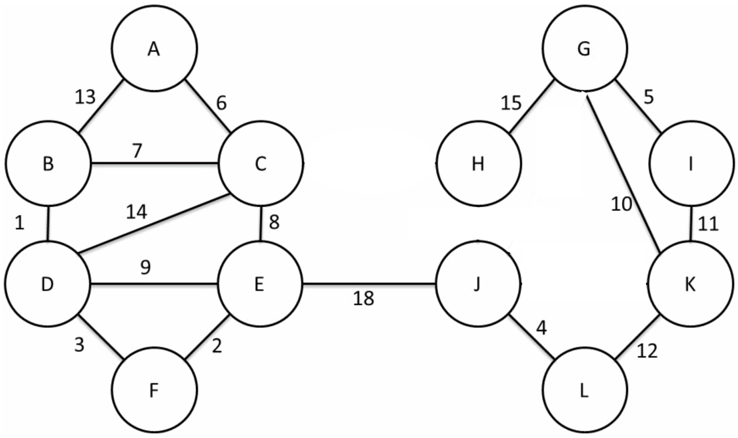
Kante 17 wird gelöscht:



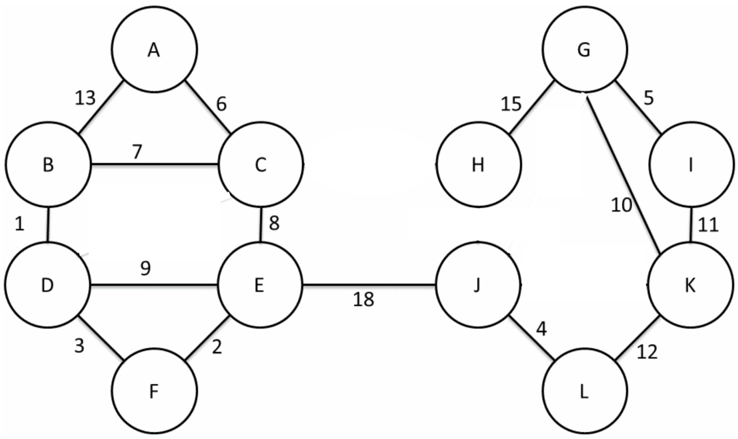
Kante 16 wird gelöscht:



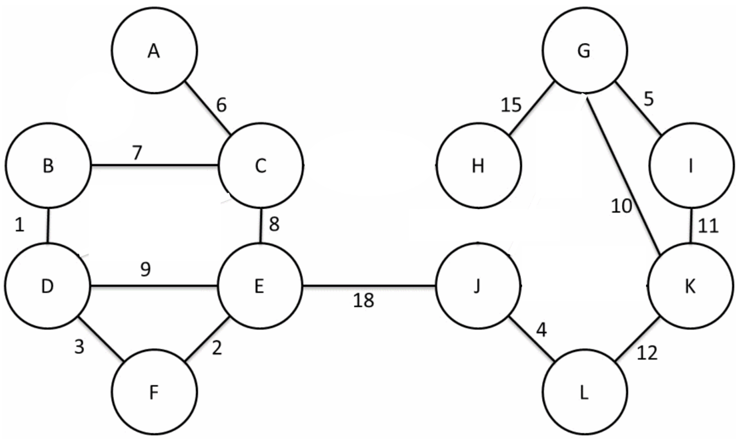
Kante 15 wird *nicht* gelöscht:



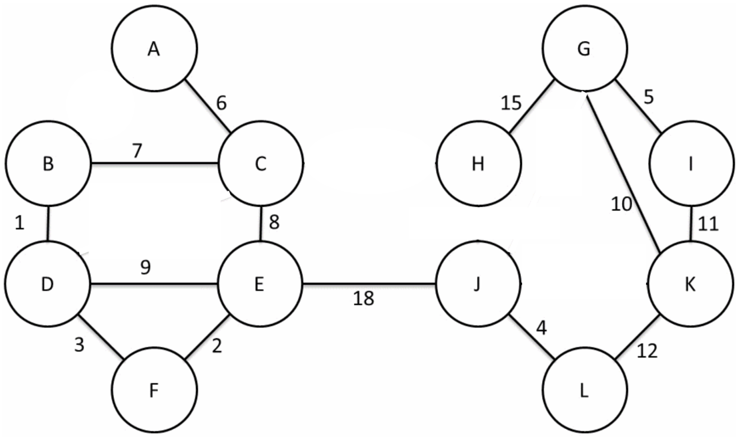
Kante 14 wird gelöscht:



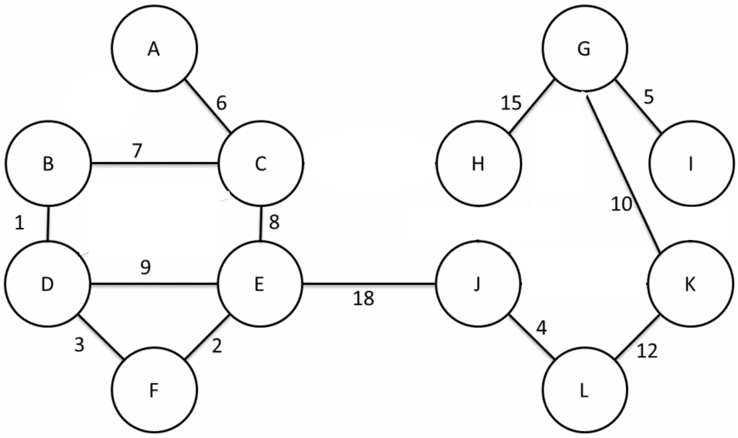
Kante 13 wird gelöscht:



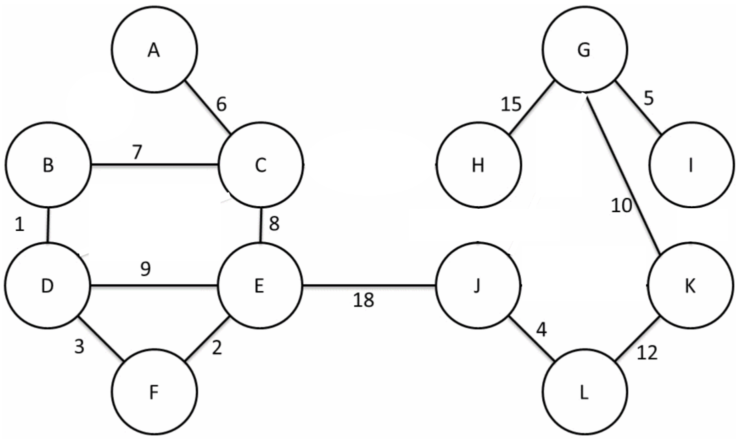
Kante 12 wird *nicht* gelöscht:



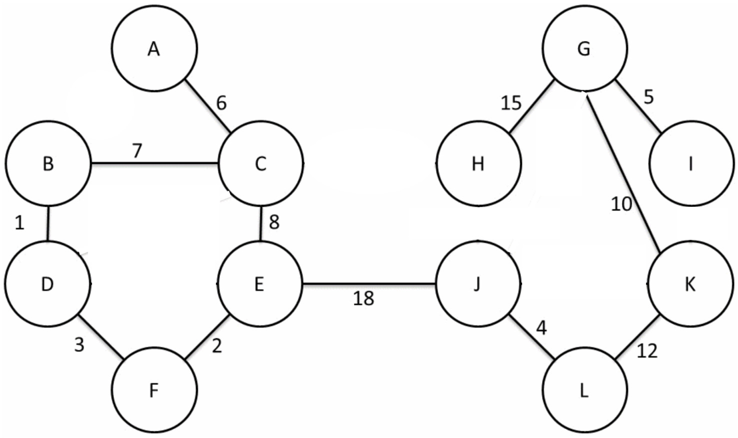
Kante 11 wird gelöscht:



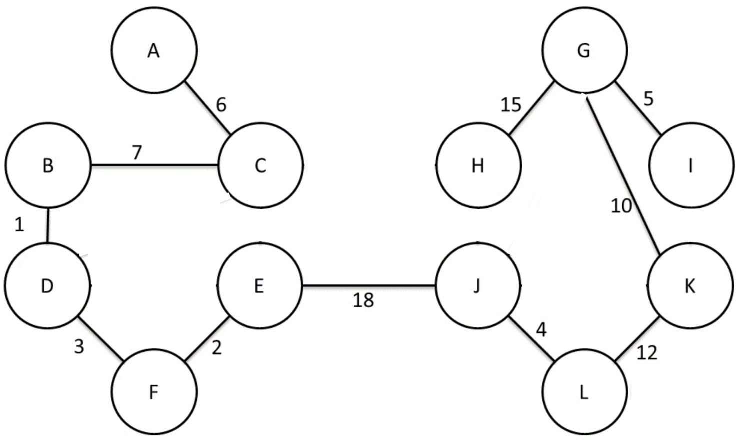
Kante 10 wird *nicht* gelöscht:



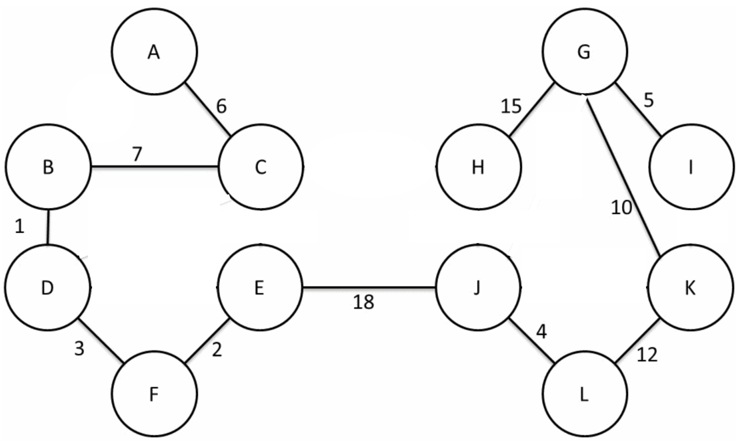
Kante 9 wird gelöscht:



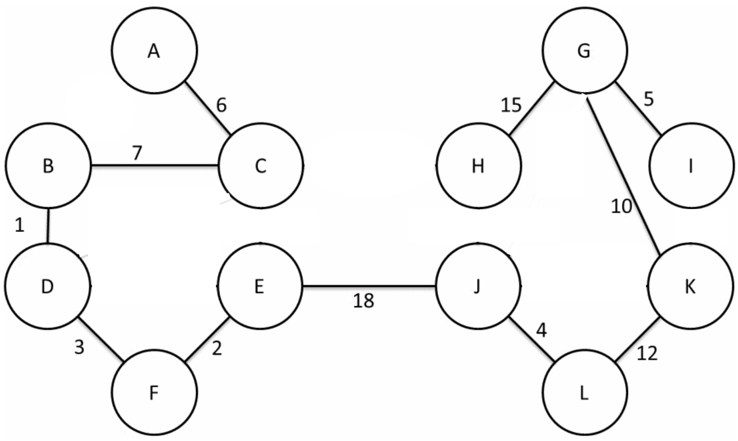
Kante 8 wird gelöscht:



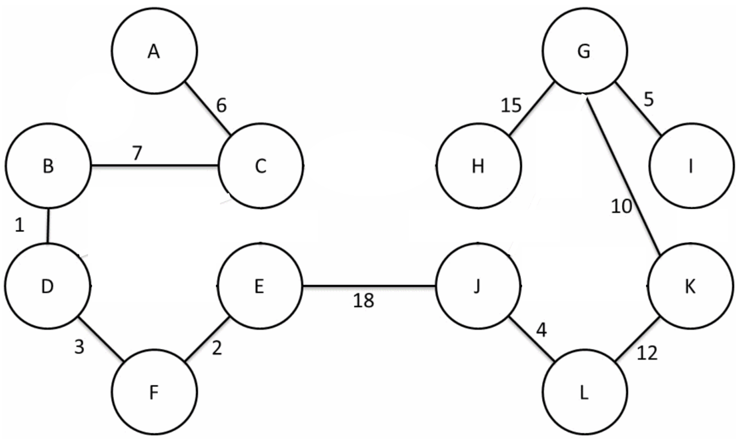
Kante 7 wird *nicht* gelöscht:



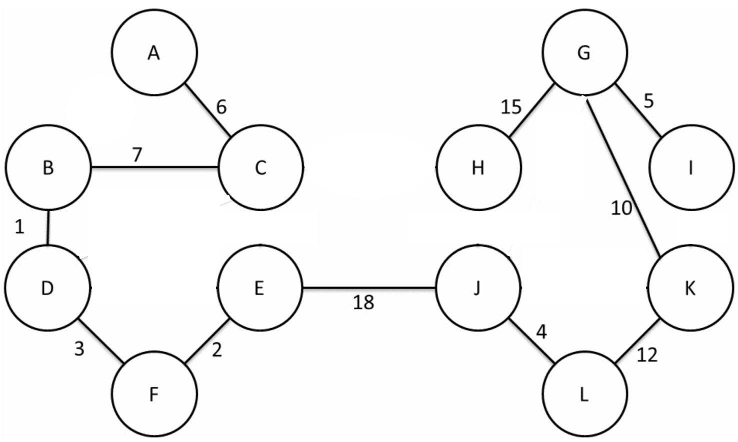
Kante 6 wird *nicht* gelöscht:



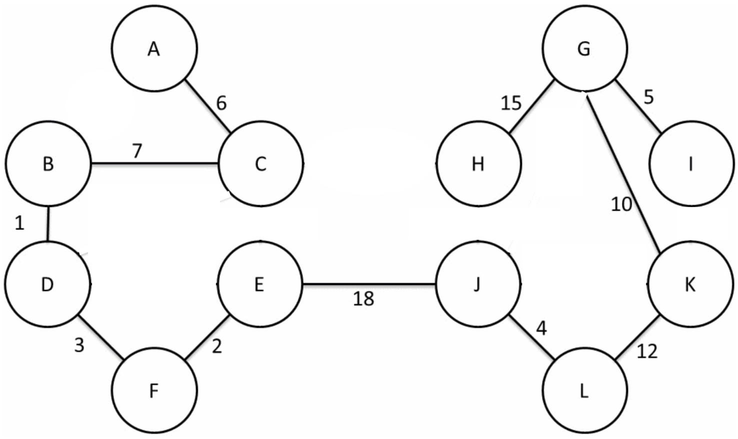
Kante 5 wird *nicht* gelöscht:



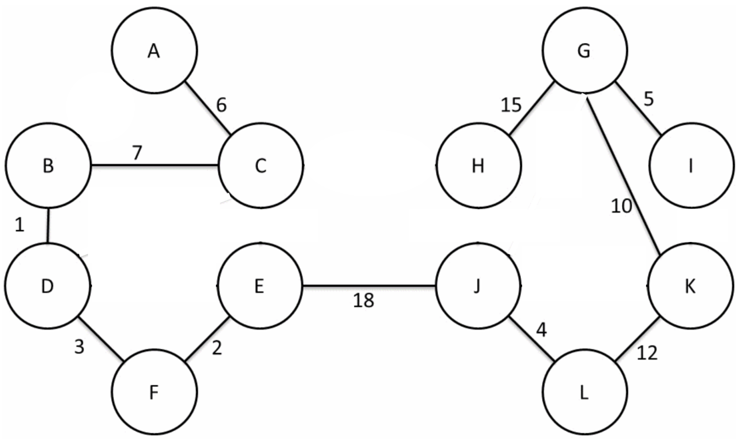
Kante 4 wird *nicht* gelöscht:



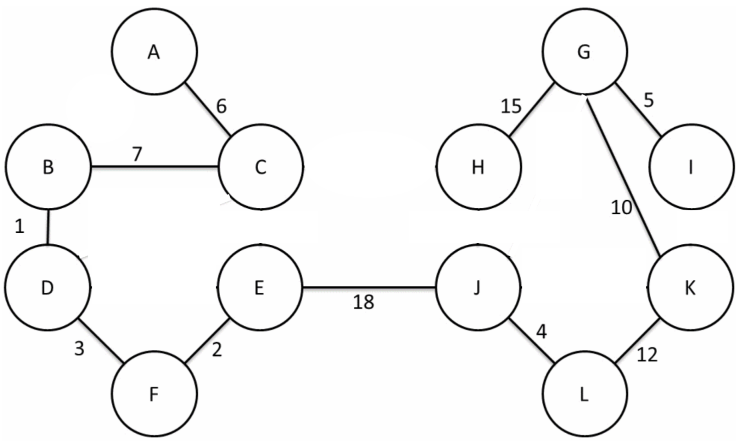
Kante 3 wird *nicht* gelöscht:



Kante 2 wird *nicht* gelöscht:



Kante 1 wird *nicht* gelöscht:



______________________________.

Aufgabe 2 (Minimale Spannbäume): (15 Punkte)

Beweisen oder widerlegen Sie: Der minimale Spannbaum eines Graphen, dessen Kantengewichte alle paarweise verschieden sind, ist eindeutig.

Lösung:

Seien $T_1 = (V, E_1)$ und $T_2 = (V, E_2)$ zwei *unterschiedliche minimale Spannbäume*. Da beide Spannbäume voneinander verschieden sind, ihre Knotenmengen aber übereinstimmen, müssen notwendigerweise die Kantenmengen ungleich sein, d.h. es muss es in einem der beiden Spannbäume eine Kante geben, die in dem anderen Spannbaum nicht vorkommt.

Es sei nun e_1 diejenige Kante mit *kleinstem Gewicht*, die in einem von beiden aber nicht in beiden Spannbäumen vorkommt. Da alle Kanten nach Annahme eindeutig über ihr Gewicht identifiziert werden können, ist diese Kante e_1 auch eindeutig. Wir nehmen im Weiteren ohne Beschränkung der Allgemeinheit an, dass diese Kante e_1 in T_1 ist.

Wir betrachten nun den Graphen $T_2 \cup \{e_1\} := (V, E_2 \cup \{e_1\})$, also den Graphen, der entsteht, wenn wir dem minimalen Spannbaum T_2 die Kante e_1 hinzufügen. Da e_1 nicht in T_2 enthalten ist, T_2 aber bereits ein Baum war, muss $T_2 \cup \{e_1\}$ nun notwendigerweise einen Zykel enthalten. Da T_1 keine Zykel enthält, muss es auf diesem Zykel eine Kante e_2 geben, welche in T_2, aber nicht in T_1 enthalten ist. Da die Kante e_2 in einem, aber nicht beiden Spannbäumen enthalten ist, ist e_2 nicht diejenige, die diese Eigenschaft erfüllt *und* minimales Gewicht hat, denn diese Eigenschaft hat *alleine* die Kante e_1. Es muss also gelten:

$$W(e_1) < W(e_2)$$

Da e_2 Teil eines Zykels in $T_2 \cup \{e_1\}$ ist, können wir die Kante e_2 löschen und erhalten somit einen neuen Spannbaum

$$T^* = \big(V, \ \big(E_2 \setminus \{e_2\}\big) \cup \{e_1\}\big) \ ,$$

welcher wegen $W(e_1) < W(e_2)$ ein *geringeres Gewicht als der bereites minimale Spannbaum* T_2 hat.

Dies ist ein Widerspruch und die Annahme, dass zwei voneinander verschiedene minimale Spannbäume existieren, kann nicht wahr sein.

Aufgabe 3 (Single-Source Single-Target Shortest Path): (25 Punkte)

Betrachten wir zunächst folgende Definition:
Ein Graph mit Koordinaten ist ein 4-Tupel (V, E, W, K) wobei

- (V, E, W) ein gewichteter Graph ist, und
- $K : V \to \mathbb{R} \times \mathbb{R}$ jeden Knoten auf eine Koordinate abbildet.

Zusätzlich nehmen wir an, dass $W(v_1, v_2) \geq \|K(v_1) - K(v_2)\|_2$.

Hinweise:

- Informell bedeutet dies, dass das Gewicht zwischen zwei Knoten mindestens die Länge der Luftlinie ist.

Wir suchen den kürzesten Weg zwischen s und t.
Dazu definieren wir unten zusätzlich eine Funktion $h: V \to \mathbb{R}$.
Wir passen den Dijkstra-Algorithmus so an, dass

- `ExtractMin` nun $\arg\min_{v \in Q} h(v) + \text{dist}[v]$ auswählt, und

- sobald `ExtractMin` t auswählt, terminiert der Algorithmus.

Beweisen oder widerlegen Sie:

a) Für $h(v) = \|K(v) - K(t)\|_1$ berechnet der angepasste Dijkstra-Algorithmus einen kürzesten Pfad von s nach t.

b) Für $h(v) = \|K(v) - K(t)\|_{\max}$ berechnet der angepasste Dijkstra-Algorithmus einen kürzesten Pfad von s nach t.

Hinweise:

- $\|(x, v)\|_1 = |x| + |y|$
- $\|(x, y)\|_2 = \sqrt{x^2 + y^2}$
- $\|(x, y)\|_{\max} = \max\{|x|, |y|\}$

Lösung:

a) Die Aussage ist falsch. Betrachten wir folgendes Gegenbeispiel:

Der Graph ist gegeben durch:

- $V = \{A, B, C, D\}$,
- $E = \{e_1 = \{A, B\}, e_2 = \{B, D\}, e_3 = \{A, C\}, e_4 = \{C, D\}\}$,
-
  - $K(A) = (0, 0)$,
  - $K(B) = (1, 1)$,
  - $K(C) = (3, 4)$,
  - $K(D) = (4, 4)$
-
  - $W(e_1) = \sqrt{2}$
  - $W(e_2) = 3\sqrt{2}$
  - $W(e_3) = 5$
  - $W(e_4) = 1$

wobei wir eine kürzeste Strecke zwischen $s = A$ und $t = D$ suchen. Die kürzeste Strecke ist gegeben durch ABD mit Länge $4\sqrt{2} < 4 \cdot 1.5 = 6$.

Es gilt

- $h(B) = 6$,
- $h(C) = 1$,
- $h(D) = 0$.

Wir fangen an mit A, $\text{dist}[B] = \sqrt{2}$, $\text{dist}[C] = 5$. ExtractMin wählt C $(1+5)$, $\text{dist}[D] = 6$. ExtractMin wählt D und terminiert.

b) Die Aussage ist richtig. Aus der Globalübung folgt, das es reicht zu Zeigen dass f entlang jedem Pfad mit Startknoten s monoton ist.

Sei $\pi = s \dots vv' \dots$.

$$\begin{aligned} f(v') &= dist[v'] + h(v') \\ &= dist[v] + W(v,v') + h(v') \\ &\geq dist[v] + \|K(v) - K(v')\|_2 + h(v') \\ &\geq dist[v] + \|K(v) - K(v')\|_{\max} + h(v') \\ &= dist[v] + \|K(v) - K(v')\|_{\max} + \|K(v') - K(t)\|_{\max} \\ &\geq dist[v] + \|K(v) - K(t)\|_{\max} = dist[v] + h(v) = f(v) \end{aligned}$$

Also ist jeder Pfad Monoton in f, und (siehe GÜ) wählt ExtractMin einen Knoten erst, wenn der Minimale Pfad von s zu diesem Knoten gefunden wurde. Wenn der Algorithmus terminiert, wurde Knoten t gefunden und demnach der kürzeste Weg von s nach t.

Aufgabe 4 (Schiebepuzzle): (10 Punkte)

Betrachten Sie das n−Schiebepuzzel, wobei ein Quadrat der Größe n mal n gegeben ist. An fast jeder Position in dem Quadrat befindet sich eines der $n^2 - 1$ Puzzelteile, eindeutig beschriftet mit einer Zahl aus dem Interval 1 bis $n^2 - 1$. Es gibt genau eine leere Position. Ein Puzzelteil, das horizontal oder vertikal adjazent zu der leeren Position ist, kann auf die leere Position geschoben werden. Ziel des Puzzles ist es, die Teile an eine vordefinierte Position zu schieben, und zwar in möglichst wenigen Zügen.
In folgendem Beispiel ($n = 3$) ist die leere Position in der zweiten Zeile und der ersten Spalte. Die möglichen Aktionen ergeben sich durch das Verschieben des Puzzelteils 2, 7 oder 4.

2	3	6
	7	8
4	1	5

Formalisieren Sie dieses Problem mit Hilfe von Graphen, und beschreiben Sie, welcher Algorithmus aus der Vorlesung zur Lösung hilfreich ist. Beschreiben Sie zusätzlich, wie groß der Graph in Abhängigkeit von n ist.

Lösung:

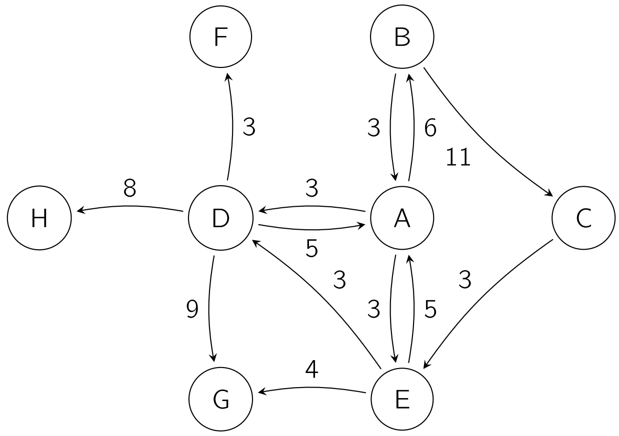
.Wir betrachten den (ungerichteten) Graphen $G = (V, E, W)$:

- $V = \{A \in \{0 \dots n^2\}^{n \times n} | a_{ij} = a_{kl} \iff i = k \wedge j = l\}$,
- $E \subseteq V \times V$, sodass $\{A, B\} \in E$ gdw:
  - $a_{i,j} = 0 \wedge a_{i+1,j} = b_{i,j} \wedge b_{i+1,j} = 0$, oder
  - $a_{i,j} = 0 \wedge a_{i,j+1} = b_{i,j} \wedge b_{i,j+1} = 0$.
- $W(e) = 1$ für alle $e \in E$.

Es gibt circa $(n^2)!$ Knoten in G.
Als Algorithmus zum finden einer minimalen Anzahl Zügen bietet sich entweder eine Breitensuche oder auch Dijkstra an.

Aufgabe 5 (Dijkstra-Algorithmus): (10 Punkte)

Betrachten Sie den folgenden Graphen:



Führen Sie den *Dijkstra* Algorithmus auf diesem Graphen mit dem *Startknoten A* aus. Falls mehrere Knoten für die nächste Iteration zur Wahl stehen, werden die Knoten dabei in alphabetischer Reihenfolge betrachtet. Füllen Sie dazu die nachfolgende Tabelle aus:

Knoten	A						
B							
C							
D							
E							
F							
G							
H							

Lösung: ______________________________

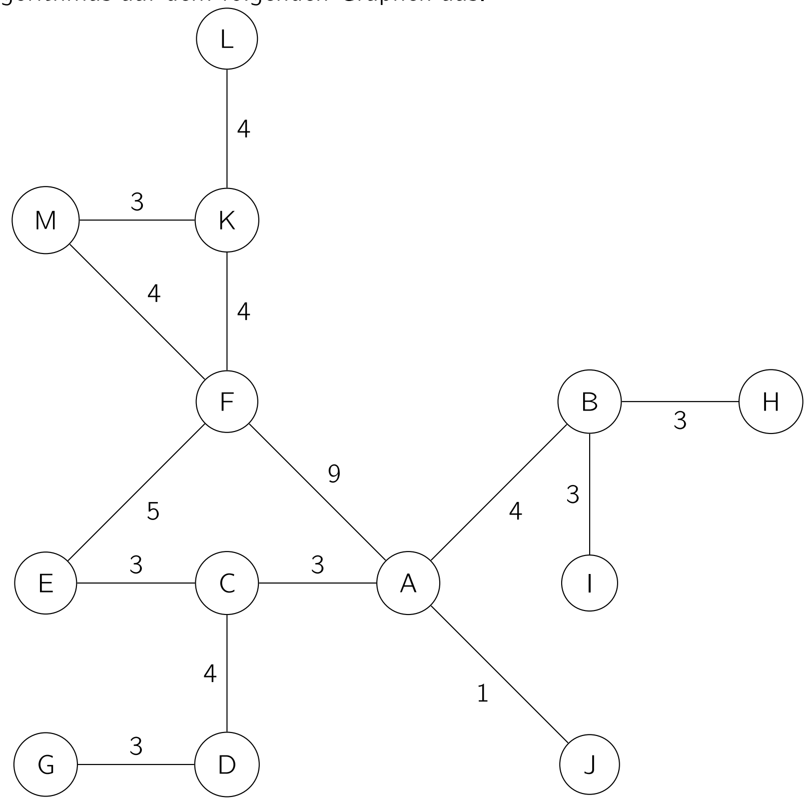
______________________________.

Knoten	A	D	E	B	F	G	H
B	6	6	6	–	–	–	–
C	∞	∞	∞	17	17	17	17
D	3	–	–	–	–	–	–
E	3	3	–	–	–	–	–
F	∞	6	6	6	–	–	–
G	∞	12	7	7	7	–	–
H	∞	11	11	11	11	11	–

Die grau unterlegten Zellen markieren, an welcher Stelle für welchen Knoten die minimale Distanz sicher berechnet worden ist.

Aufgabe 6 (Prim-Algorithmus): (15 Punkte)

Führen Sie Prims Algorithmus auf dem folgenden Graphen aus.



Der Startknoten hat hierbei den Schlüssel A. Geben Sie dazu *vor* jedem Durchlauf der äußeren Schleife an,

1. welche Kosten die Randknoten haben (d. h. für jeden Knoten `v` in `pq` die Priorität von `v`, wobei ∞ angibt, dass der entsprechende Knoten noch nicht zum Randbereich gehört)
2. und welchen Knoten `pq.getMin()` wählt, indem Sie den Kosten-Wert des gewählten Randknoten in der Tabelle unterstreichen (wie es in der ersten Zeile bereits vorgegeben ist).

Geben Sie zudem den vom Algorithmus bestimmten minimalen Spannbaum an.

#Iteration	A	B	C	D	E	F	G	H	I
1	<u>0</u>	∞	∞	∞	∞	∞	∞	∞	∞
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									

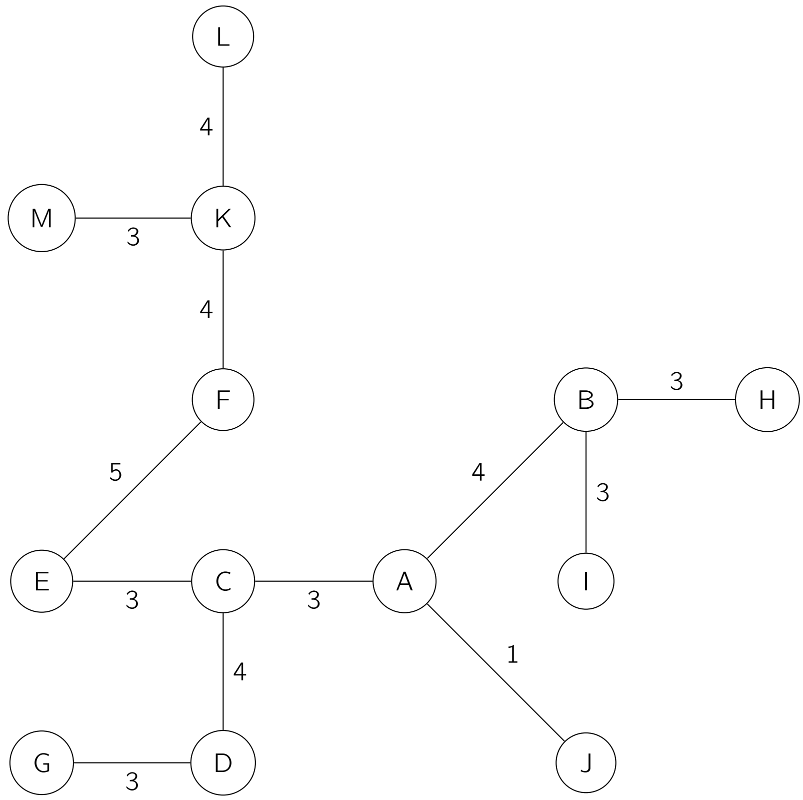
J	K	L	M
∞	∞	∞	∞

Minimaler Spannbau

Lösung:

#Iteration	A	B	C	D	E	F	G	H	I	J	K	L	M
1	$\underline{0}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2		4	3	∞	∞	9	∞	∞	∞	$\underline{1}$	∞	∞	∞
3		4	$\underline{3}$	∞	∞	9	∞	∞	∞		∞	∞	∞
4		4		4	$\underline{3}$	9	∞	∞	∞		∞	∞	∞
5		$\underline{4}$		4		5	∞	∞	∞		∞	∞	∞
6				4		5	∞	$\underline{3}$	3		∞	∞	∞
7				4		5	∞		$\underline{3}$		∞	∞	∞
8				$\underline{4}$		5	∞				∞	∞	∞
9						5	$\underline{3}$				∞	∞	∞
10						$\underline{5}$					∞	∞	∞
11											$\underline{4}$	∞	4
12												4	$\underline{3}$
13												$\underline{4}$	

Hierbei gibt eine unterstrichene Zahl an, in welcher Iteration (zugehöriger Zeilenkopf) welcher Knoten (zugehöriger Spaltenkopf) durch `pq.getMin()` gewählt wurde. Wir erhalten den folgenden minimalen Spannbaum:


L
4
M
3
K
4
F
5
B
3
H
4
3
E
3
C
3
A
I
4
1
G
3
D
J

______________________________.