VL13, Lösung 1

Der Algorithmus von Dijkstra zur Bestimmung der kürzesten (bzw. billigsten) Wege ist grob vereinfacht eine modifizierte Breitensuche, bei der der nächste zu besuchende Knoten nicht anhand der Anzahl der Zwischenstationen, sondern der Distanz (bzw. des Preises) ermittelt wird.

Algorithmus von Dijkstra in Tabellenform (orange = markiert, rot = Änderung):

Neu markierter Knoten	BOS length, pred	BWI length, pred	DTW length, pred	EWR length, pred	LGA length, pred	MCO length, pred	MDW length, pred	MEM length, pred	MIA length, pred	MSY length, pred	STL length, pred	TPA length, pred
BOS	0, BOS	∞, BOS	∞, BOS	49, BOS	79, BOS	∞, BOS	125, BOS	∞, BOS	∞, BOS	∞, BOS	∞, BOS	∞, BOS
EWR	0, BOS	∞, BOS	78, EWR	49, BOS	79, BOS	98, EWR	98, EWR	∞, BOS	∞, BOS	218, EWR	∞, BOS	∞, BOS
DTW	0, BOS	∞, BOS	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	∞, BOS	∞, BOS	218, EWR	∞, BOS	∞, BOS
LGA	0, BOS	∞, BOS	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	∞, BOS	268, LGA	218, EWR	∞, BOS	∞, BOS
MDW	0, BOS	∞, BOS	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	268, LGA	218, EWR	136, MDW	∞, BOS
MCO	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	218, EWR	136, MDW	∞, BOS
MIA	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	218, EWR	136, MDW	166, MIA
STL	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	218, EWR	136, MDW	166, MIA
BWI	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	218, EWR	136, MDW	166, MIA
MEM	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	195, MEM	136, MDW	166, MIA
TPA	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	195, MEM	136, MDW	166, MIA
MSY	0, BOS	147, MCO	78, EWR	49, BOS	79, BOS	98, EWR	97, DTW	166, MDW	127, MCO	195, MEM	136, MDW	166, MIA

Die billigste Route wird jeweils ausgehend vom Ziel über die Verfolgung der Vorgänger ermittelt. Aus der Tabelle kann man die billigsten Flüge ablesen:

BOS-STL:

STL ← pred(STL) = MDW ← pred(MDW) = DTW ← pred(DTW) = EWR ← pred(EWR) = BOS für $136

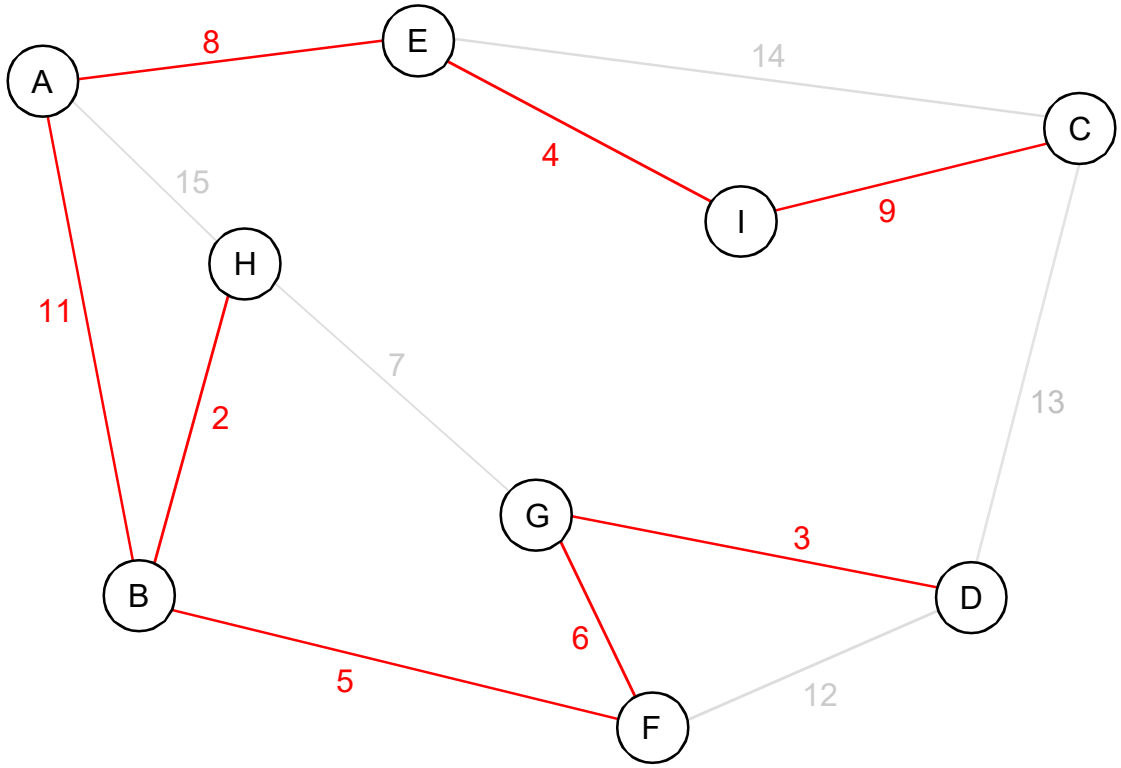
BOS-MIA:

MIA ← pred(MIA) = MCO ← pred(MCO) = EWR ← pred(EWR) = BOS für $127

BOS-MSY:

MSY ← pred(MSY) = MEM ← pred(MEM) = MDW ← pred(MDW) = DTW ← pred(DTW) = EWR ← pred(EWR) = BOS für $195

VL13, Lösung 2



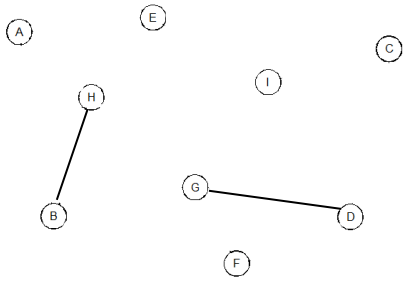
Der Ablauf des Algorithmus inkl. Union-Find-Datenstruktur (es werden nur die Bäume mit mehr als einem Knoten dargestellt) sieht wie folgt aus:

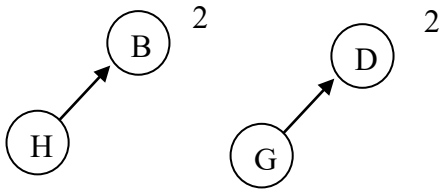
a) Schritt:
{B,H}, {D,G}, {E,I}, {B,F}, {F,G}, {G,H}, {A,E}, {C,I}, {A,B}, {D,F}, {C,D}, {C,E}, {A,H}

b) Schritt:

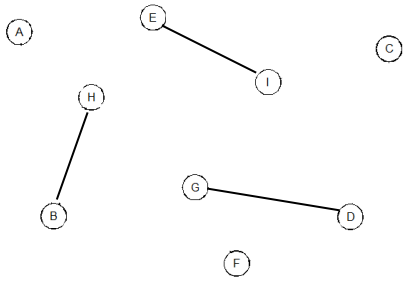
Kante {B,H}: Baum zu B und Baum zu H besitzen gleich viele Knoten; Baum zu H wird willkürlich unter Wurzel des Baumes von B gehängt

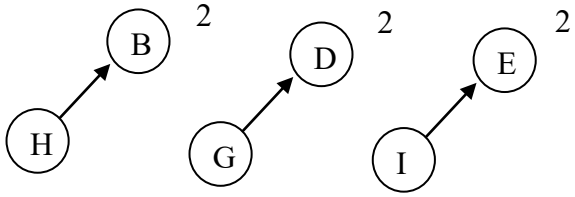
Kante {D,G}: s.o.



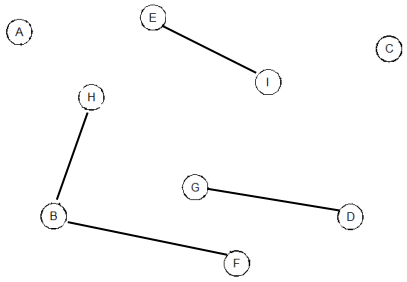


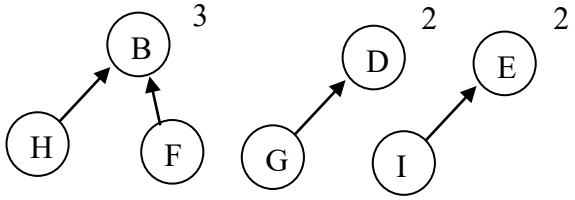
Kante {E,I}: s.o.



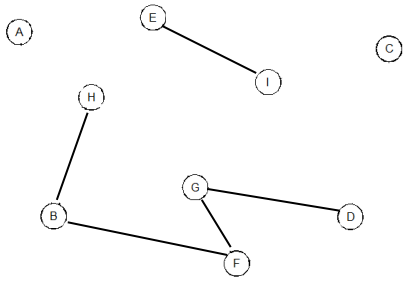


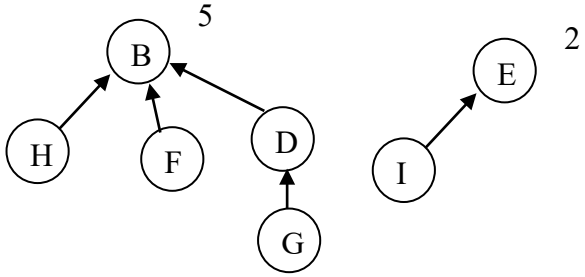
Kante {B,F}: Der Baum zu B besitzt 2 Konten. Der Baum zu F besitzt nur einen Knoten. Daher wird der Baum zu F unter der Wurzel des Baumes von B eingefügt.





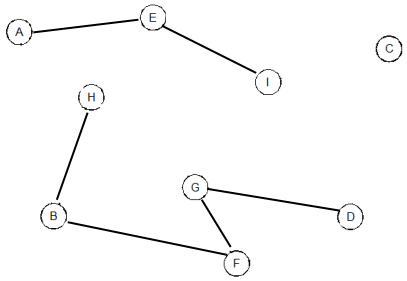
Kante{F,G}: Der Baum zu F besitzt 3 Knoten, der Baum zu G besitzt 2 Knoten. Der Baum zu G wird unter die Wurzel des Baumes zu F gehängt.

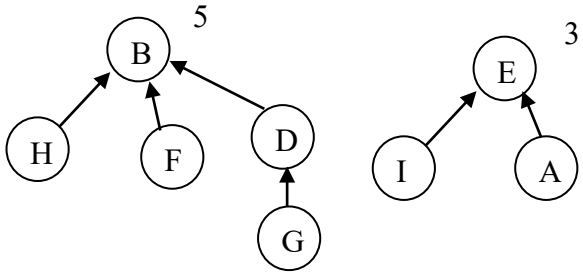




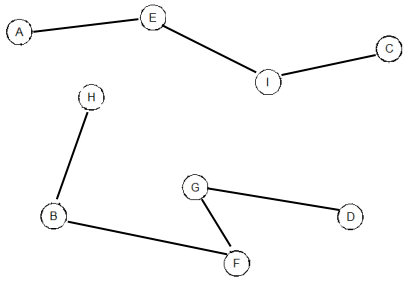
Kante {G,H} kann nicht eingefügt werden, da G und H in der Union-Find-Datenstruktur dieselbe Wurzel B besitzen und daher ein Zyklus entstehen würde.

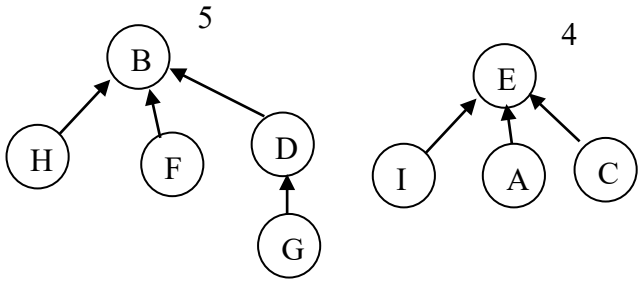
Kante {A,E}: Der Baum zu E besitzt 2 Knoten. Der Baum zu A besitzt nur einen Knoten. Daher wird der Baum zu A unter der Wurzel des Baumes von E eingefügt.



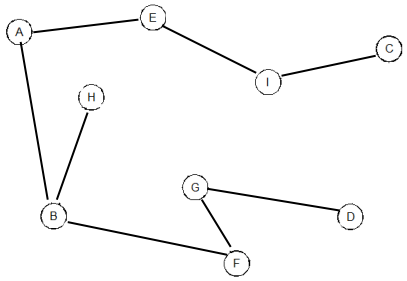


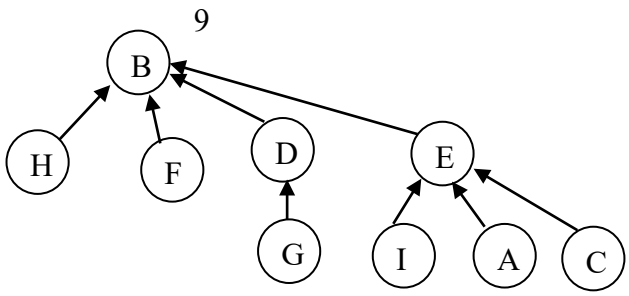
Kante {C,I}: Der Baum zu I besitzt 3 Knoten. Der Baum zu C besitzt nur einen Knoten. Daher wird der Baum zu C unter der Wurzel des Baumes von I eingefügt.





Kante {A,B}: Der Baum zu A besitzt 4 Knoten, der Baum zu B besitzt 5 Knoten. Daher wird der Baum zu A unter der Wurzel des Baumes von B eingefügt.





Kante {D,F} kann nicht eingefügt werden, da D und F in der Union-Find-Datenstruktur dieselbe Wurzel B besitzen und daher ein Zyklus entstehen würde.

Kante {C,D} kann nicht eingefügt werden, da C und D in der Union-Find-Datenstruktur dieselbe Wurzel B besitzen und daher ein Zyklus entstehen würde.

Kante {C,E} kann nicht eingefügt werden, da C und E in der Union-Find-Datenstruktur dieselbe Wurzel B besitzen und daher ein Zyklus entstehen würde.

VL13, Lösung 3

a)

```
public Graph(Knoten[] knoten)
{
	// Aufgabe 3
	// ToDo: Aus der übergebenen Listenstruktur ein Feld
	//       der Knotennamen sowie die Adjazenzmatrix aufbauen
	namen = new String[knoten.length];
	matrix = new int[knoten.length][knoten.length];

	for (int i = 0; i < knoten.length; i++)
	{
		namen[i] = knoten[i].getName();
		KnotenLink k = knoten[i].getErsterNachfolger();
		while (k !=null)
		{
			matrix[i][k.getZiel()] = k.getLaenge();
			k = k.getNaechsterNachfolger();
		}
	}

}
```

b)

```
public boolean istUngerichtet()
{
	// Aufgabe 3:
	// ToDo: prüfen, ob die Adjazenzmatrix symmetrisch ist
	for (int a = 0; a < matrix.length; a++)
		for (int b = 0; b < a; b++)
			if (matrix[a][b] != matrix[b][a])
				return false;
		return true;
}
```

Ein Java-Programme zur Lösung der Aufgabe 3 finden Sie zusätzlich in der Datei ML13-Aufgabe_3.zip.