

Algorithmen und Datenstrukturen

Aufgabe 1 Hashing – Divisionsmethode (Beispiellösung)

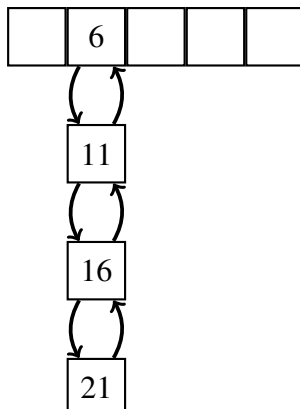
- a) Da der *modulo*-Operator Hashwerte im Intervall $[0, \dots, m - 1]$ erzeugt, reicht es eine Hash-tabelle der Grösse m zu wählen.
- b) Für die Hashwerte erhalten wir:
- $key(4) = 4 \bmod 4 = 0$
 - $key(9) = 9 \bmod 4 = 1$
 - $key(14) = 14 \bmod 4 = 2$
 - $key(19) = 19 \bmod 4 = 3$

Damit ergibt sich folgende Hashtabelle:

| | | | |
|---|---|----|----|
| 4 | 9 | 14 | 19 |
|---|---|----|----|

- c) Für die Hashwerte erhalten wir:
- $key(6) = 6 \bmod 5 = 1$
 - $key(11) = 11 \bmod 5 = 1$
 - $key(16) = 16 \bmod 5 = 1$
 - $key(21) = 21 \bmod 5 = 1$

Es ergeben sich also Kollisionen bezüglich des Hashwertes 1, welche wir wie folgt mittels einer verketteten Liste auflösen. Damit ergibt sich folgende Hashtabelle:



Aufgabe 2 KMP-Präfix-Tabelle (Beispiellösung)

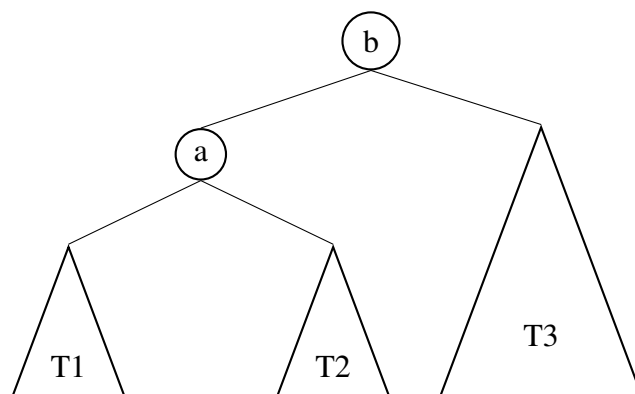
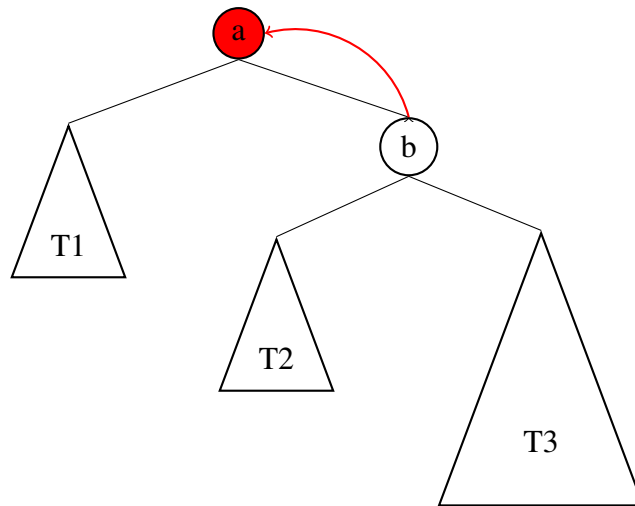
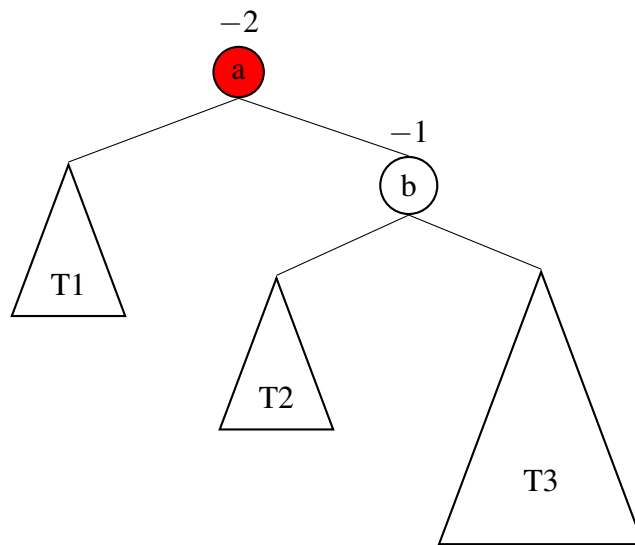
a) `std::string`
-1 0 0 0 0 0 1 2 0 0 0

b) `int sort(int* data, int n)`
-1 0 0 0 0 0 0 0 0 0 1 2 3 0 0 0 0 0 0 0 0 1 2 3 4 0

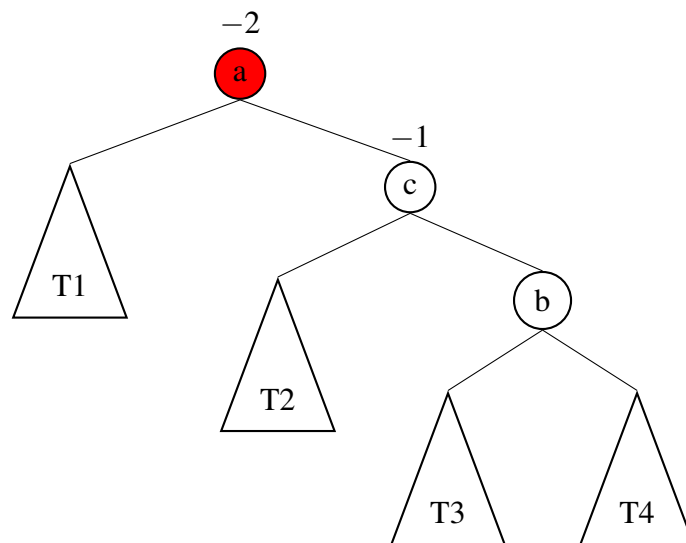
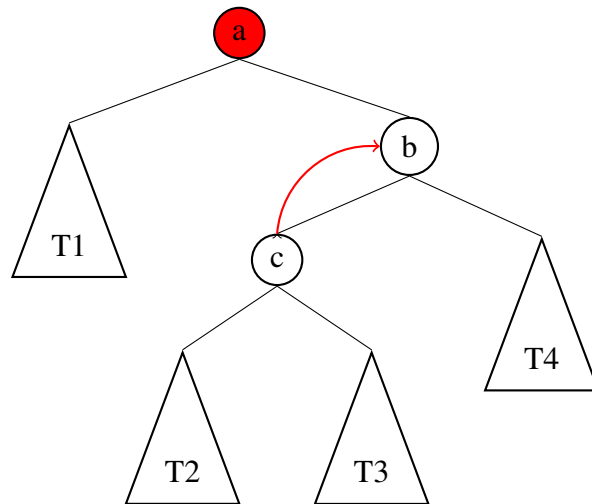
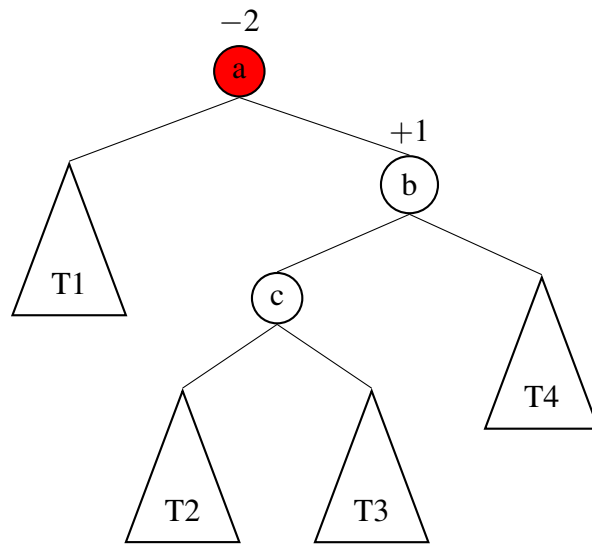
c) `TU München`
-1 0 0 0 0 0 0 0 0 0

Aufgabe 3 AVL-Baum – Rebalacierung (Beispiellösung)

a) Es handelt sich um eine (einfache) Rotation (selbes Vorzeichen in den Differenzen bei den Knoten a und b):

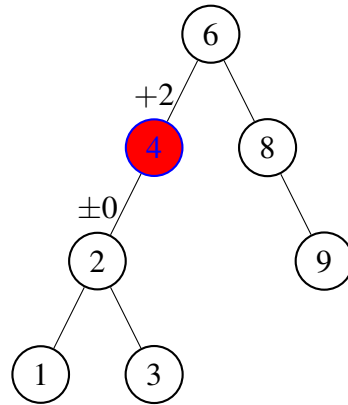
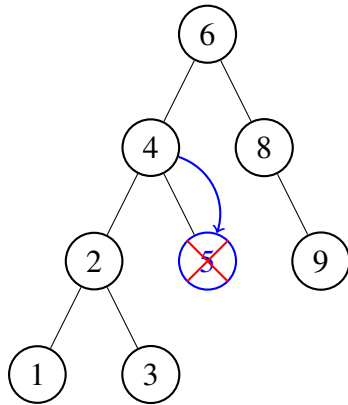
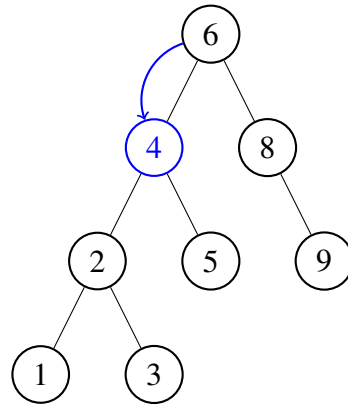
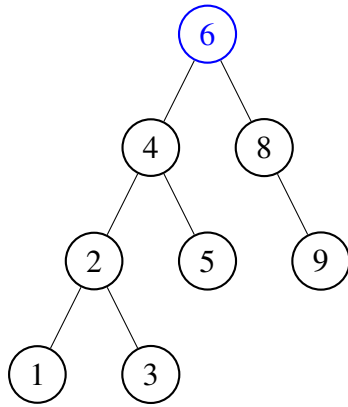


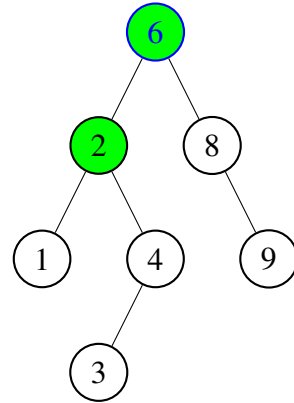
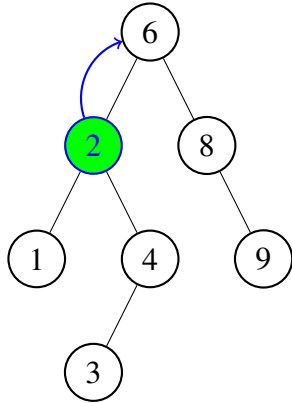
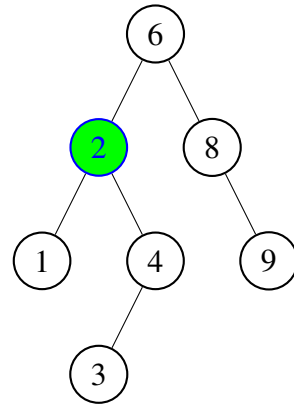
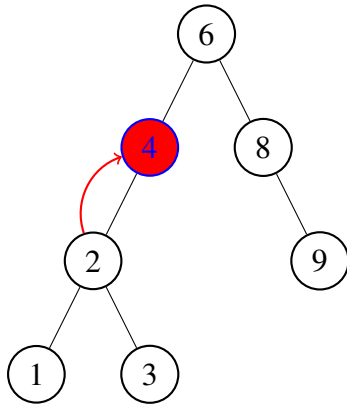
b) In diesem Fall wird eine Doppelrotation benötigt (unterschiedliches Vorzeichen in den Differenzen bei den Knoten a und b):



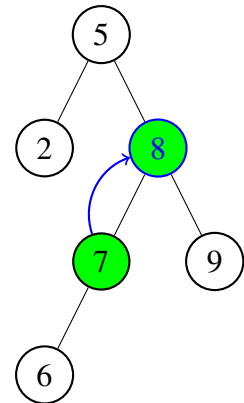
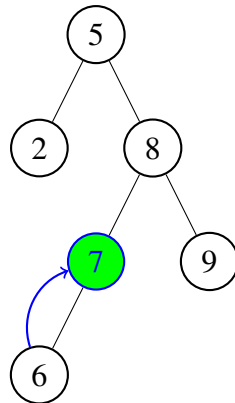
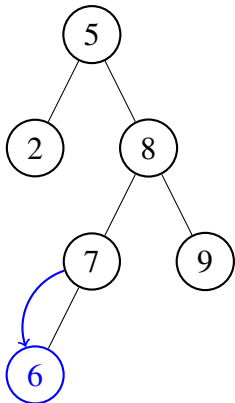
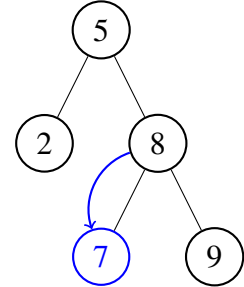
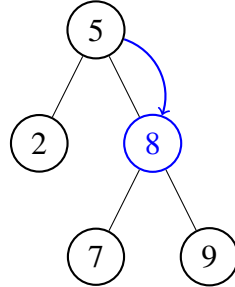
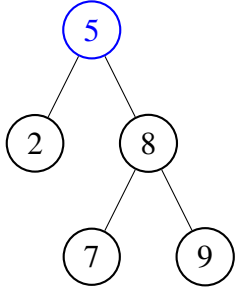
Nun muss lediglich eine weitere einfache Rotation entsprechend der Teilaufgabe (a) durchgeführt werden.

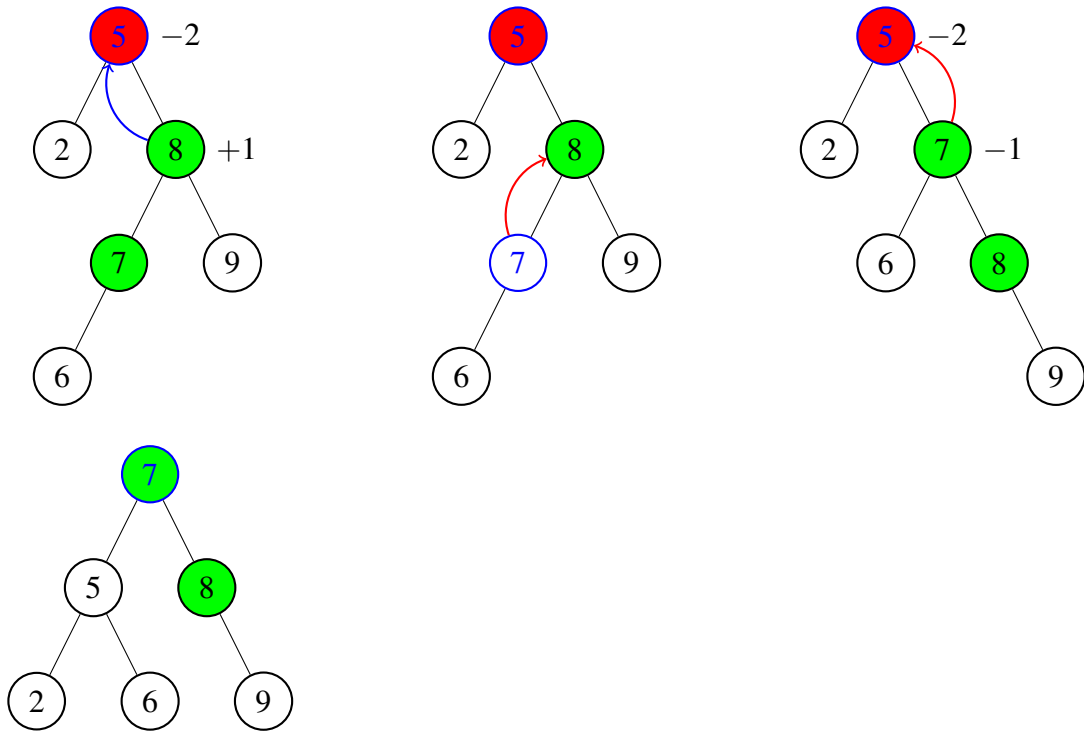
c) Zuerst suchen wir das Element „5“ indem wir entsprechend seiner Eigenschaften durch den Suchbaum traversieren. Danach stellen wir die AVL-Bedingung wieder her, indem wir, ausgehend von dem gelöschten Knoten in Richtung Wurzel traversieren und entsprechend Rotationen durchführen. Im folgenden stellt der blaue Knoten den aktuell betrachteten Knoten dar. Die Traversierung wird durch blaue Pfeile dargestellt. Rote Knoten stellen Knoten dar, bei welchen die AVL-Bedingung nicht erfüllt ist, während in grüne Knoten diese Bedingung gilt.





d) Wir gehen erneut wie in der vorherigen Teilaufgabe vor:



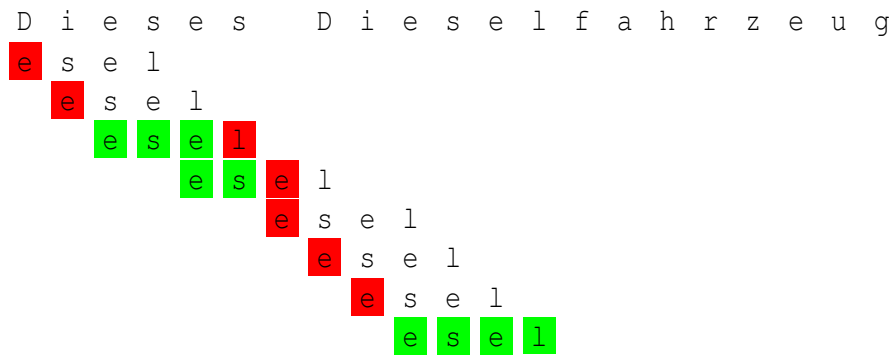


Aufgabe 4 KMP-Suche (Beispiellösung)

Zuerst stellen wir die Präfix-Tabelle auf:

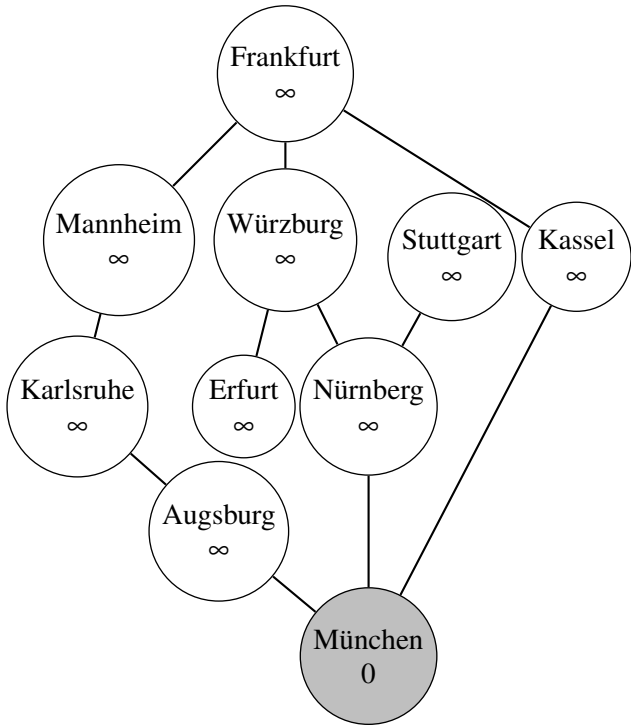
```
e s e l  
-1 0 0 1
```

Nun wenden wir den KMP-Algorithmus an:

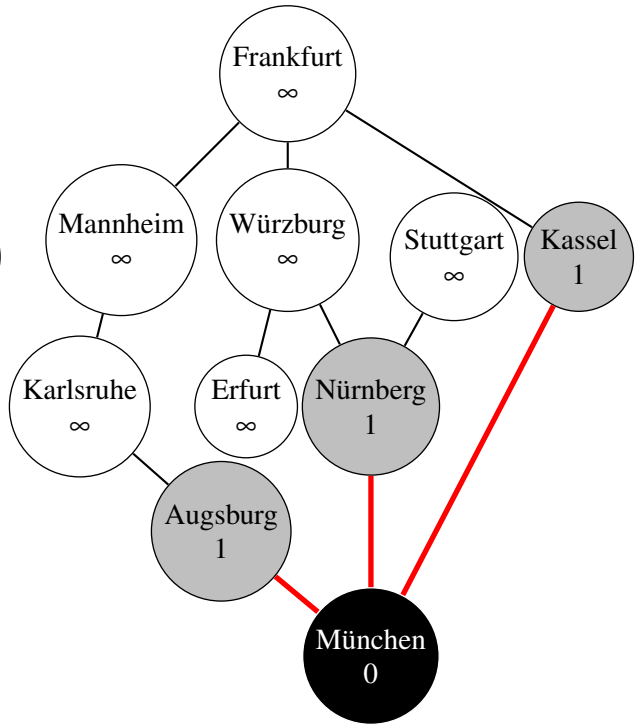


Aufgabe 5 DFS und BFS (Beispiellösung)

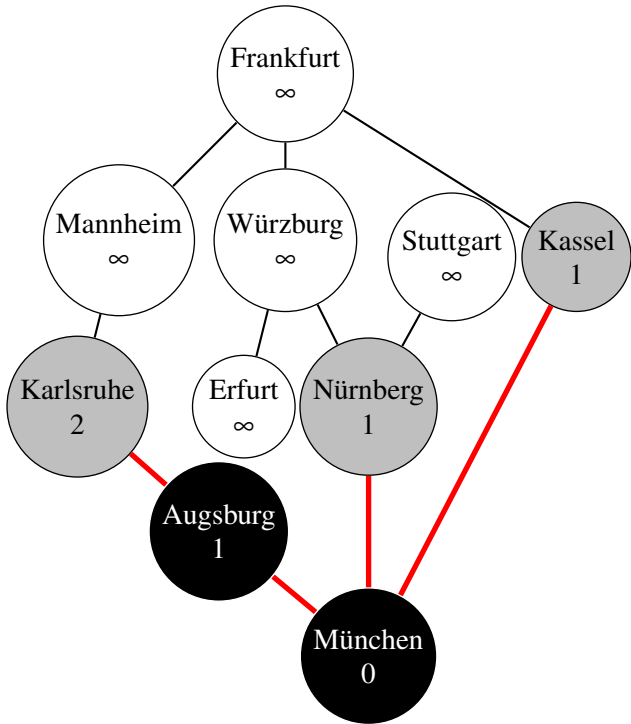
Wir beginnen mit der Anwendung des BFS:



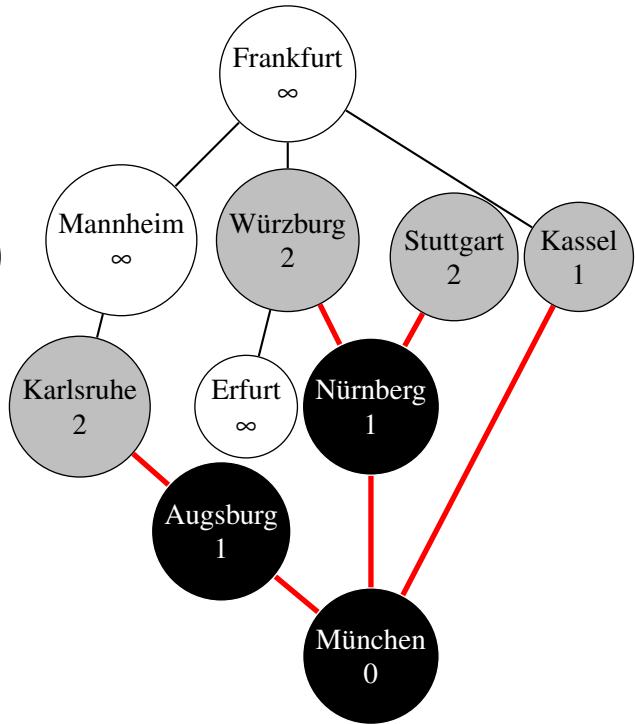
Q: München



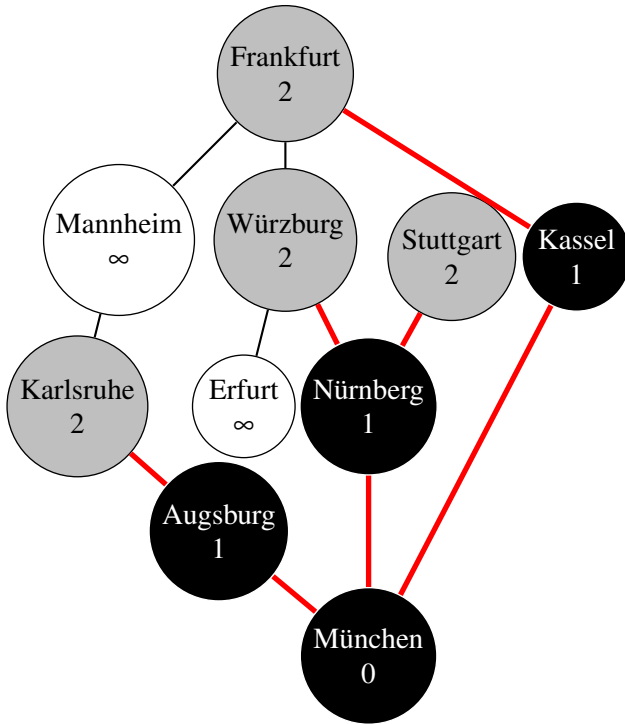
Q: Augsburg, Nürnberg, Kassel



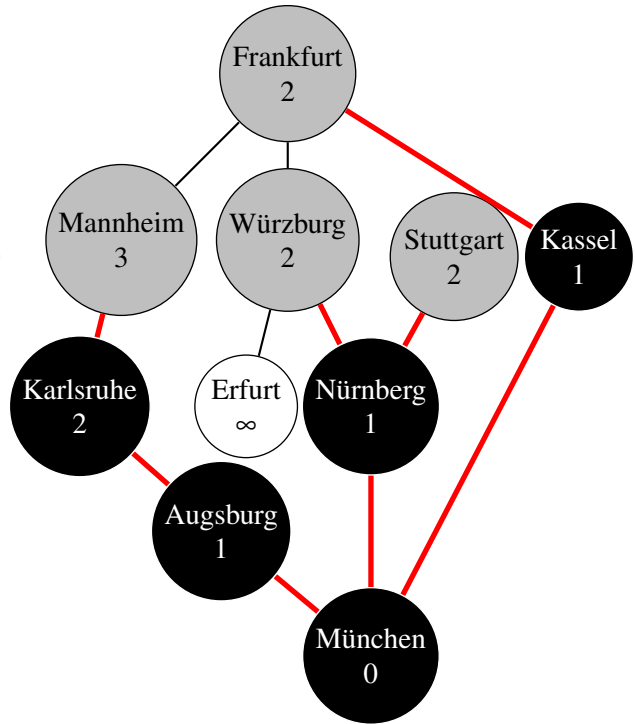
Q: Nürnberg, Kassel, Karlsruhe



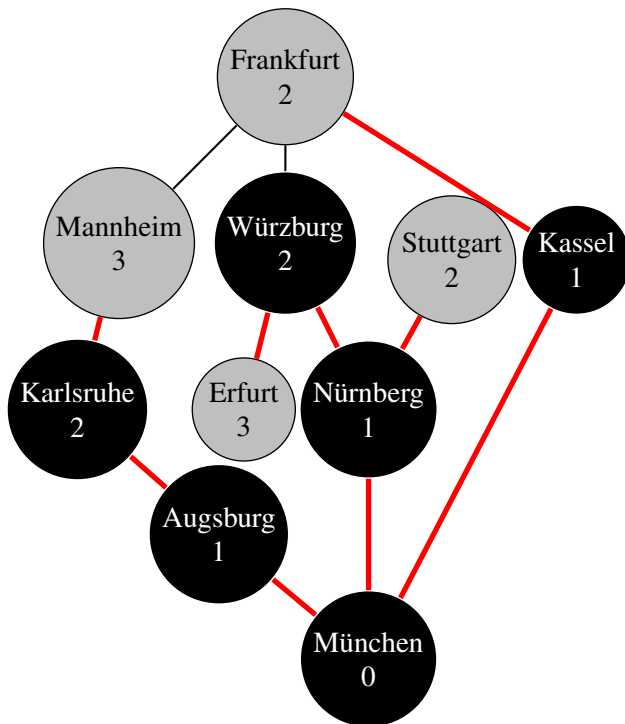
Q: Kassel, Karlsruhe, Würzburg, Stuttgart



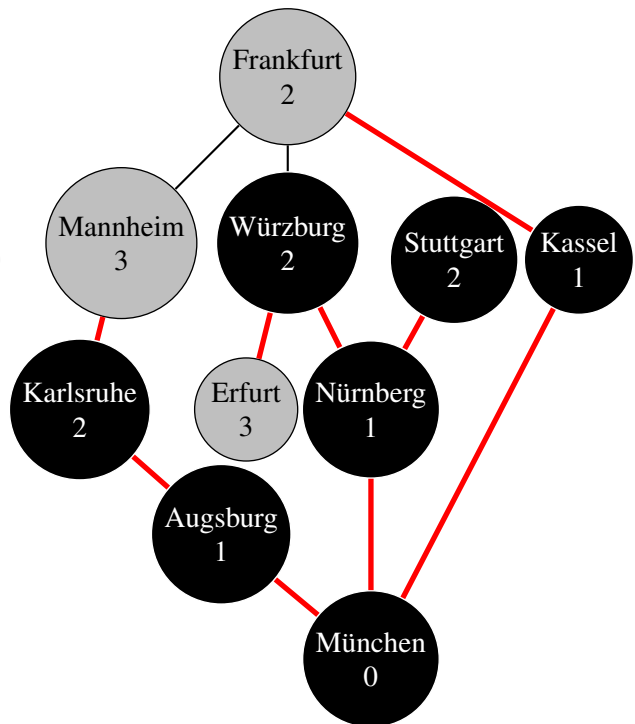
Q: Karlsruhe, Würzburg, Stuttgart, Frankfurt



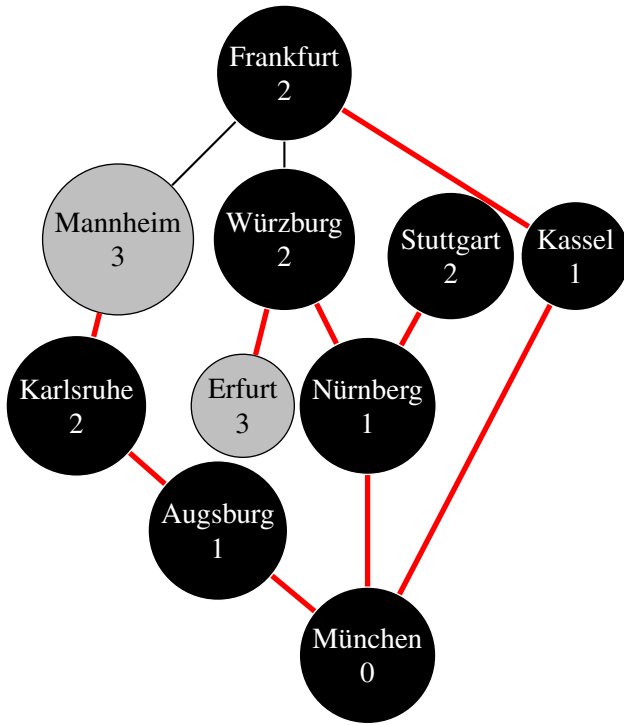
Q: Würzburg, Stuttgart, Frankfurt, Mannheim



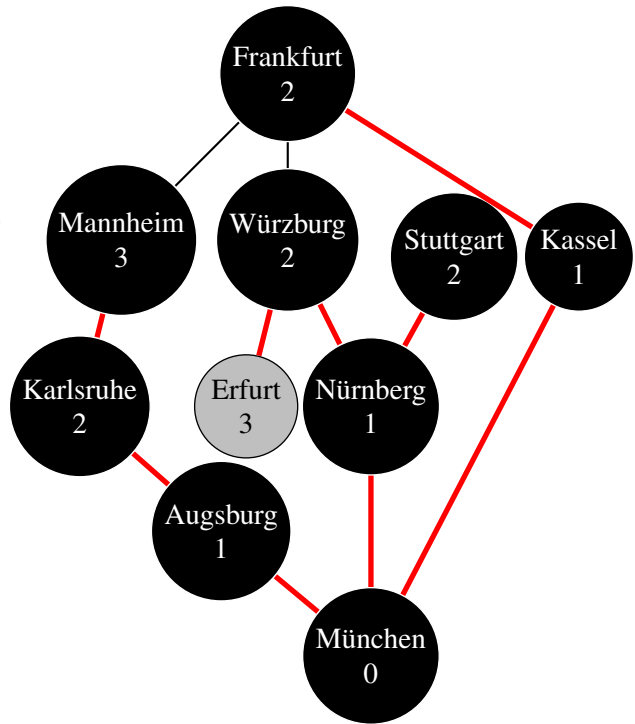
Q: Stuttgart, Frankfurt, Mannheim, Erfurt



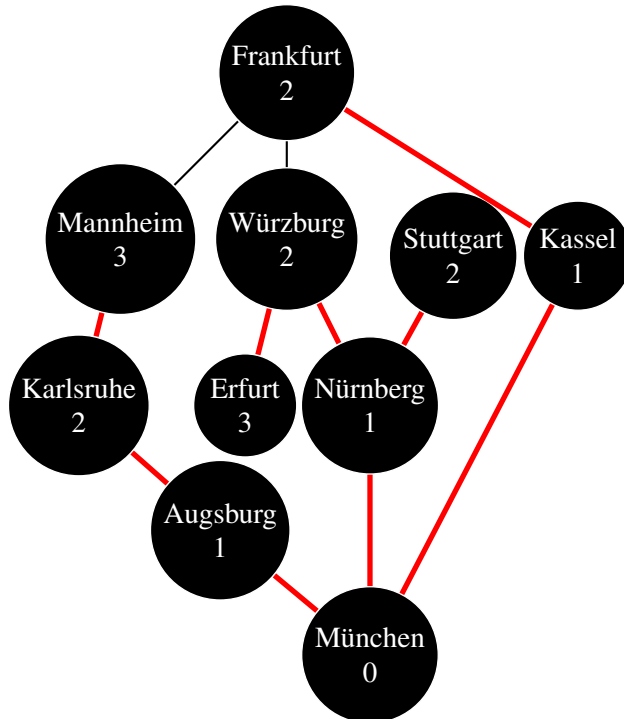
Q: Frankfurt, Mannheim, Erfurt



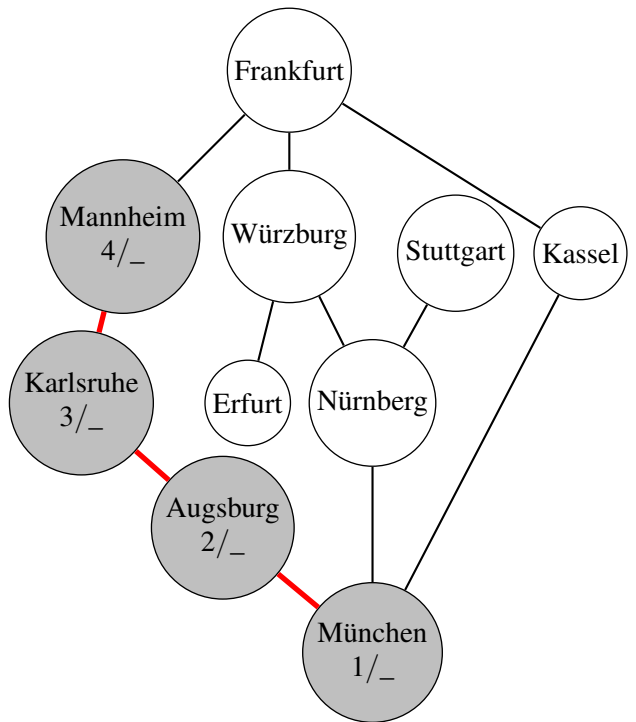
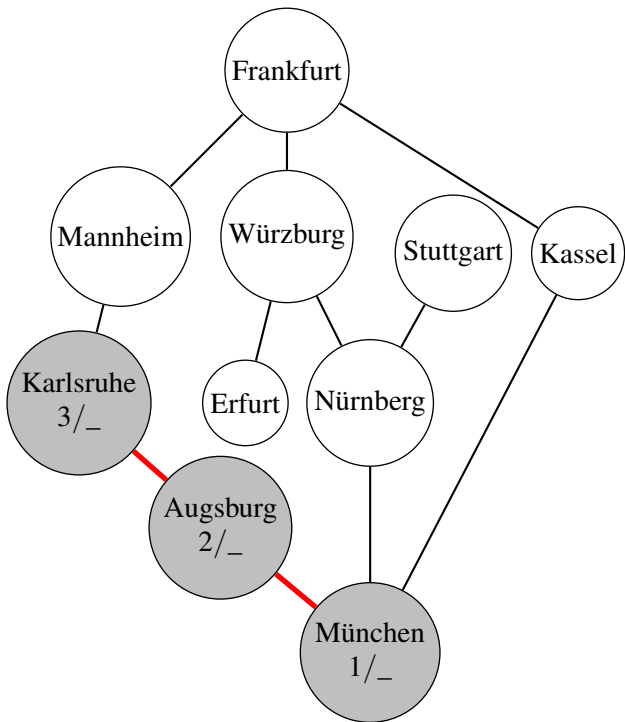
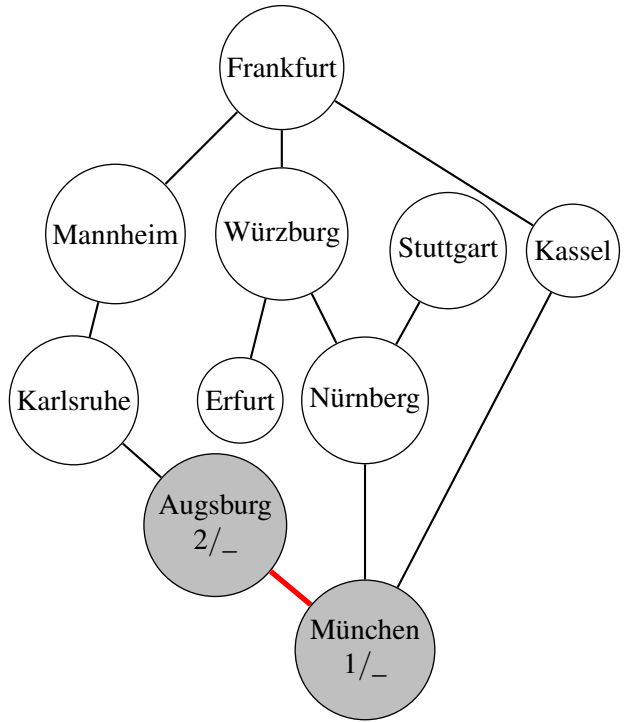
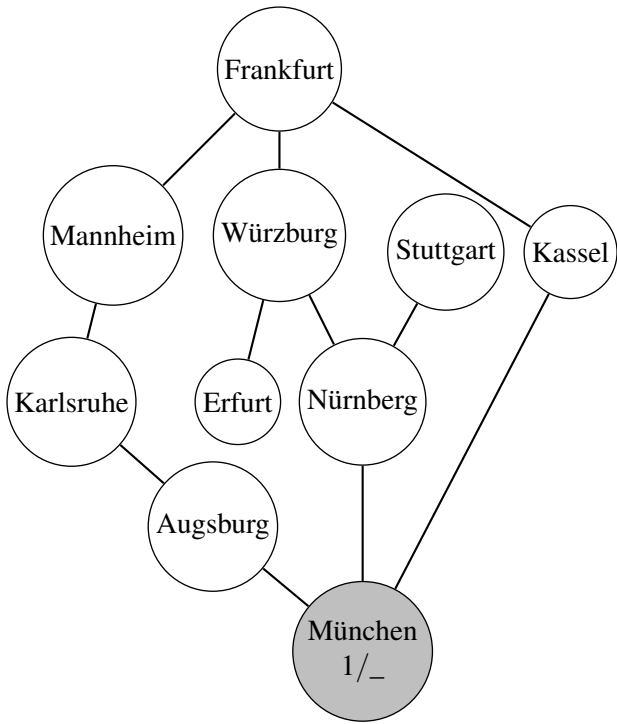
Q: Mannheim, Erfurt

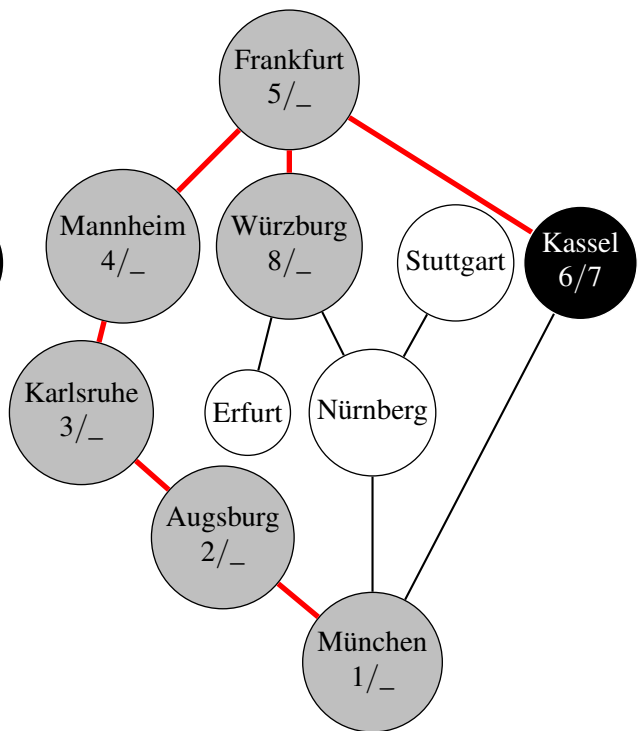
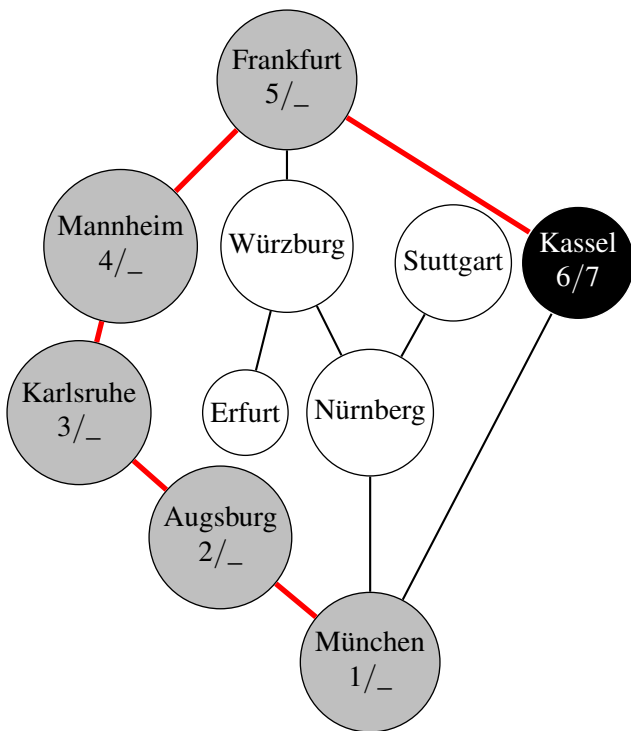
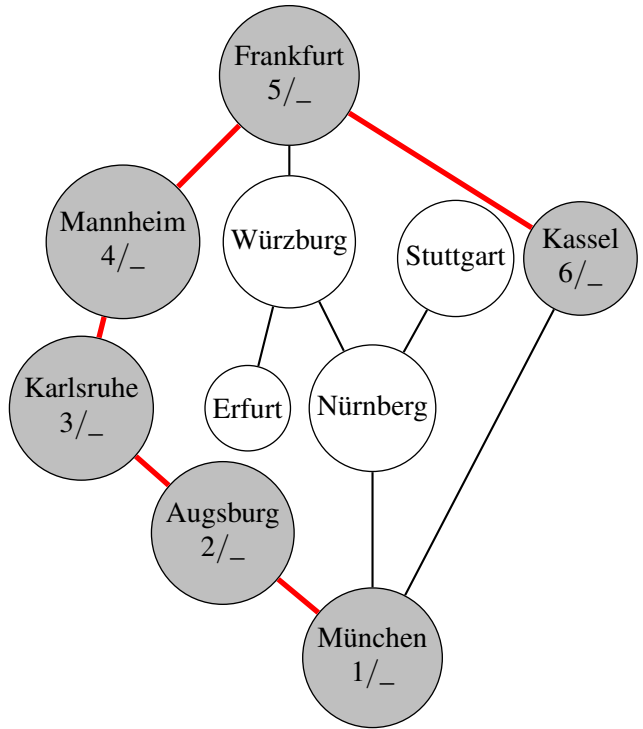
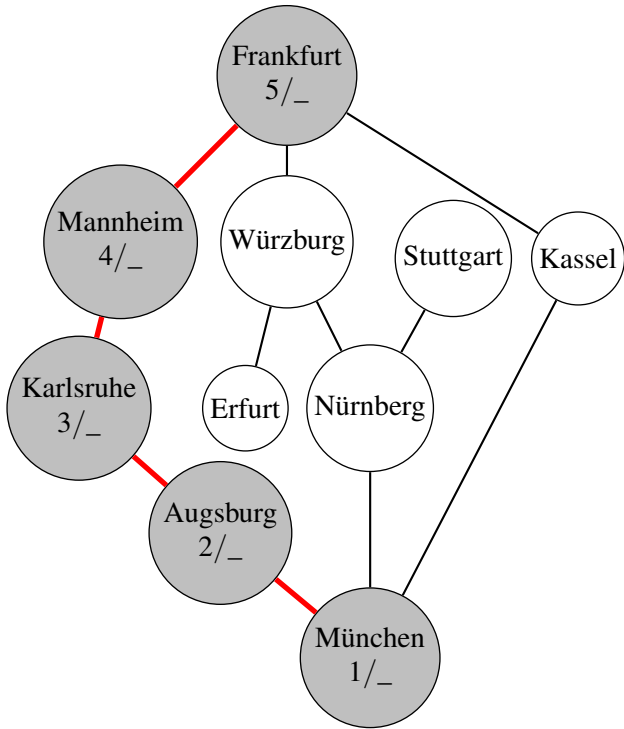


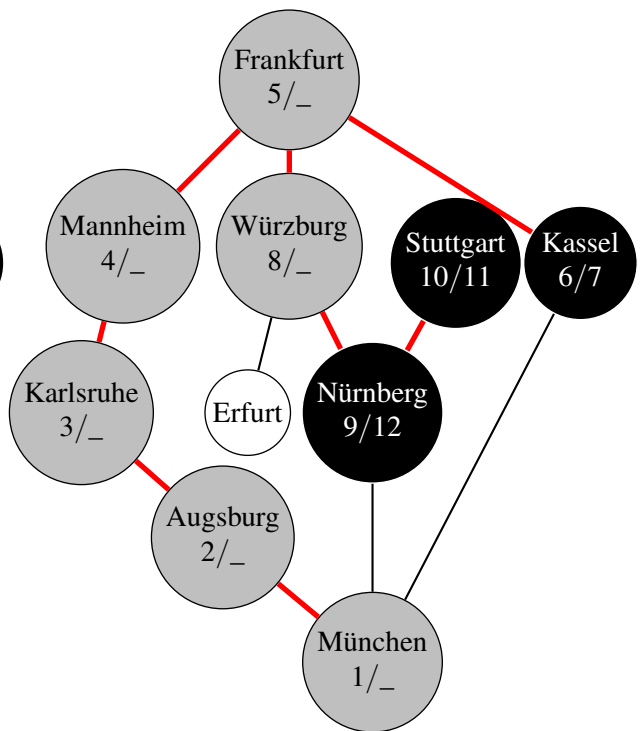
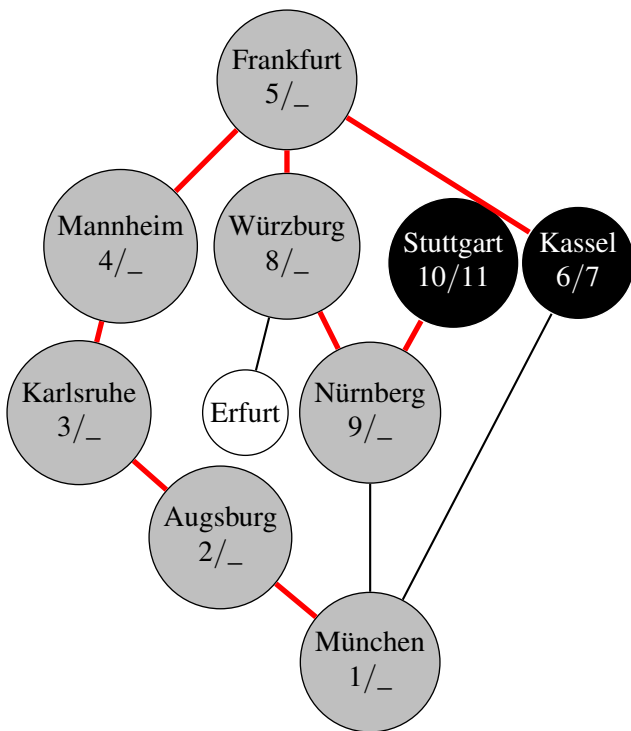
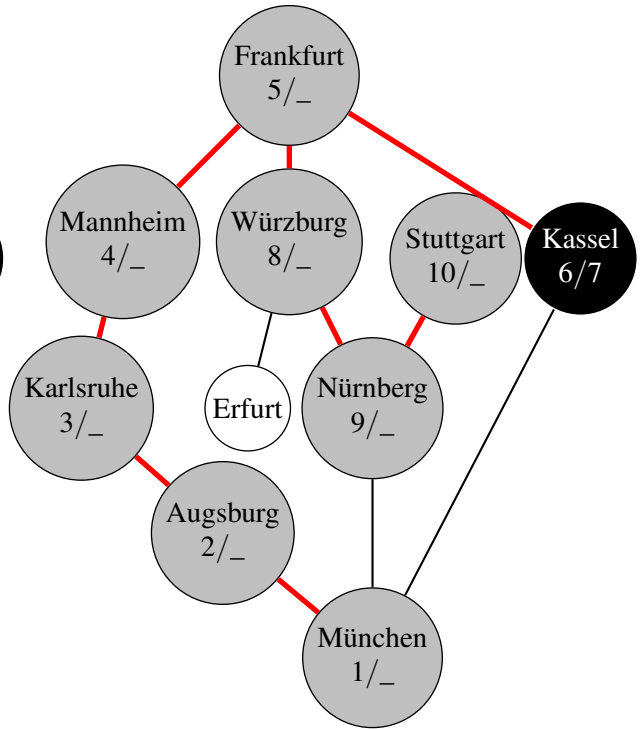
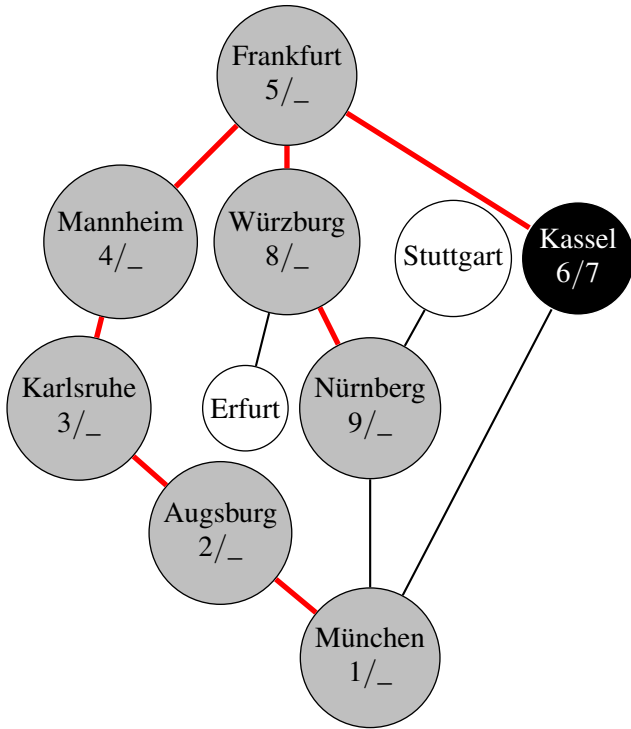
Q: Erfurt

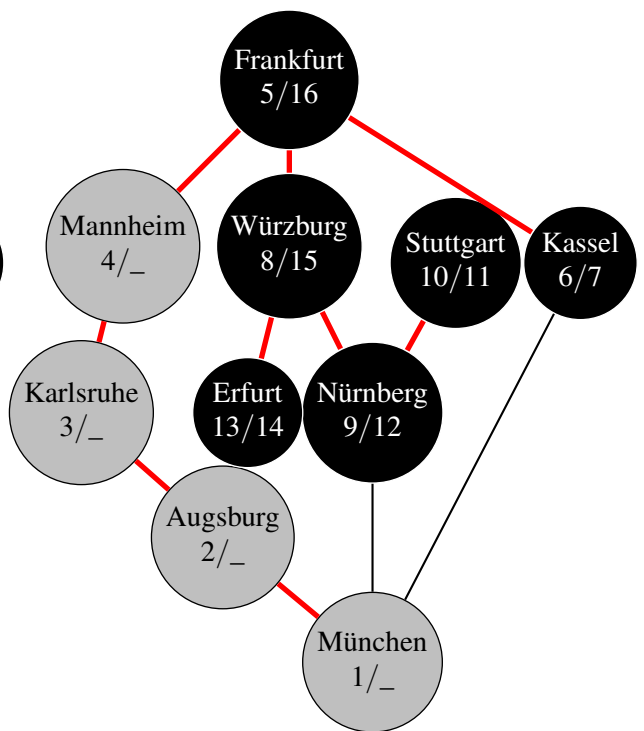
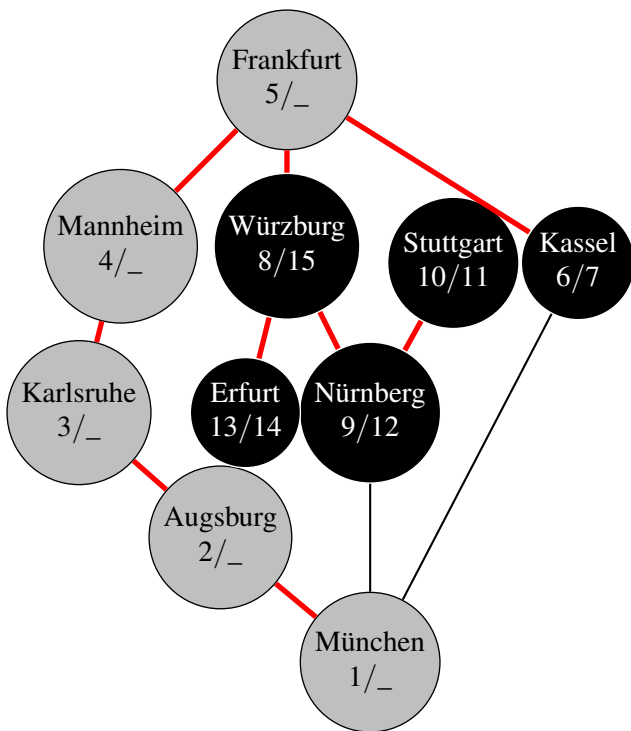
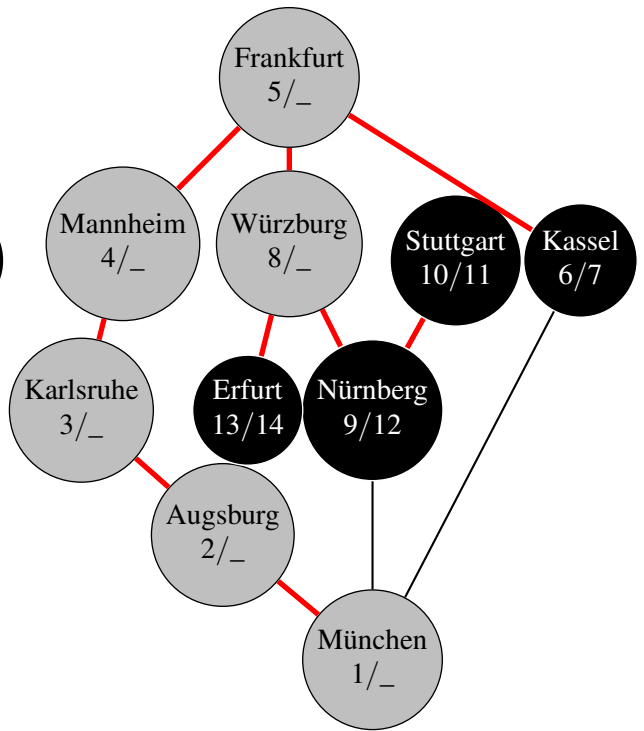
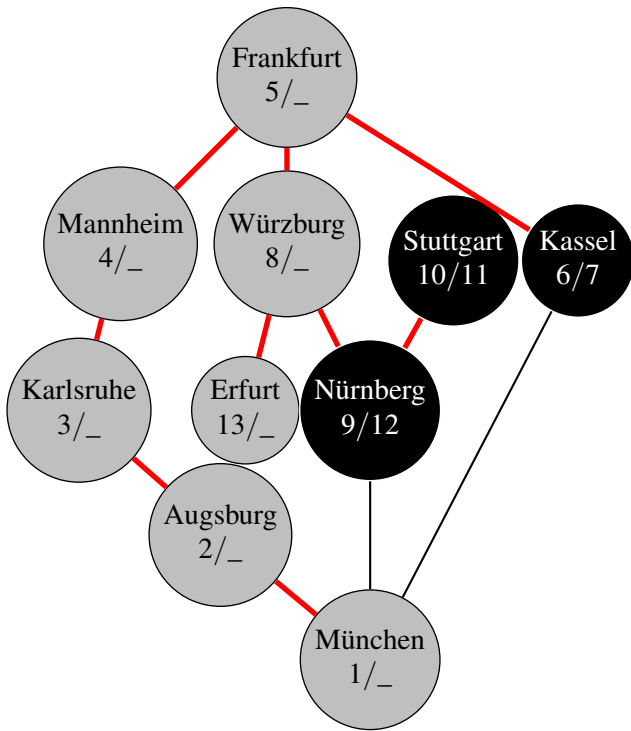


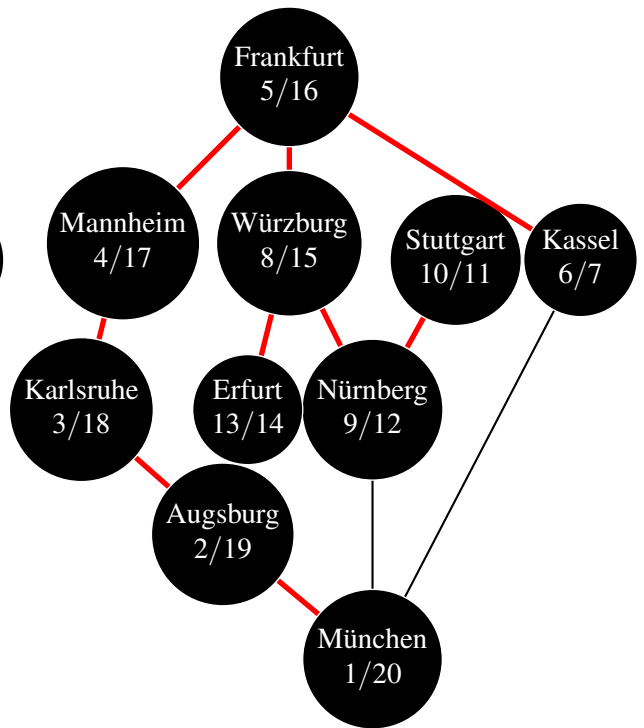
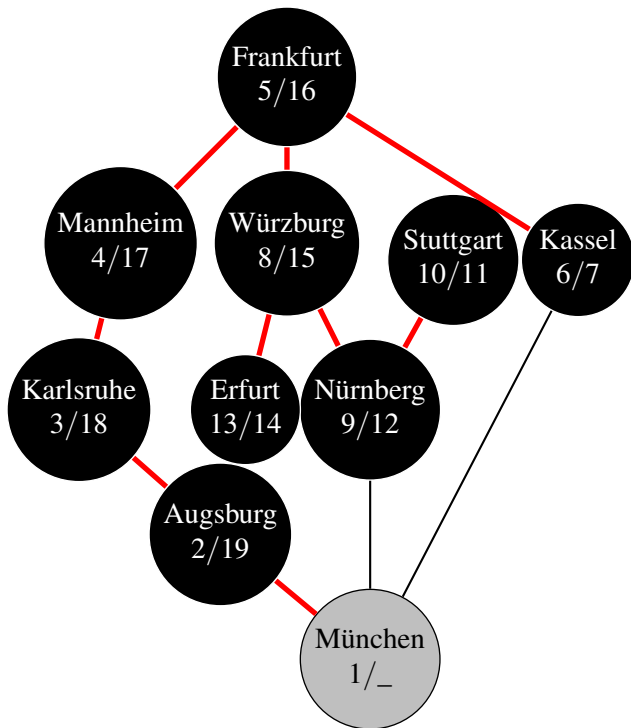
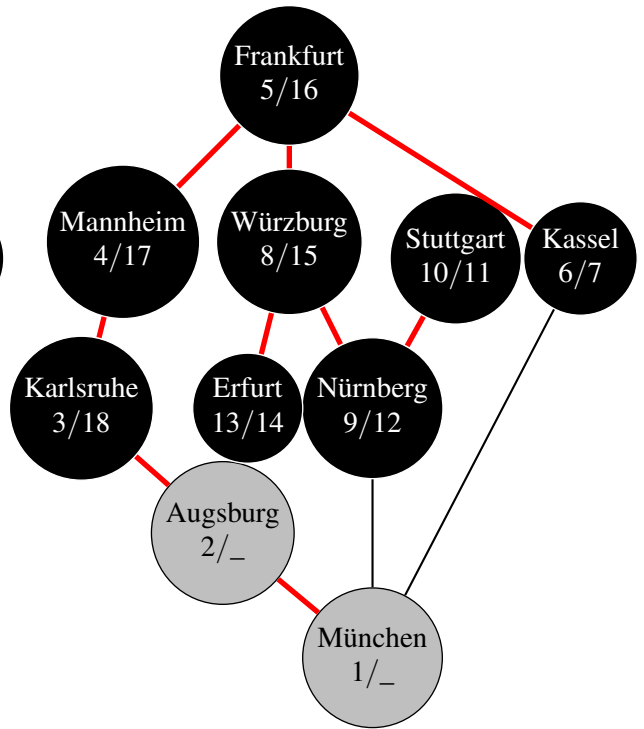
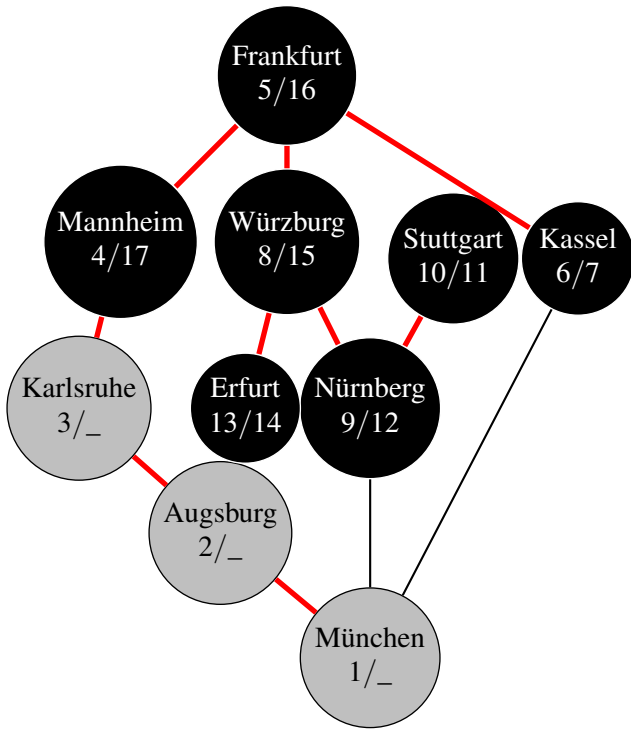
Nun wenden wir den DFS auf den gegebenen Graphen an:







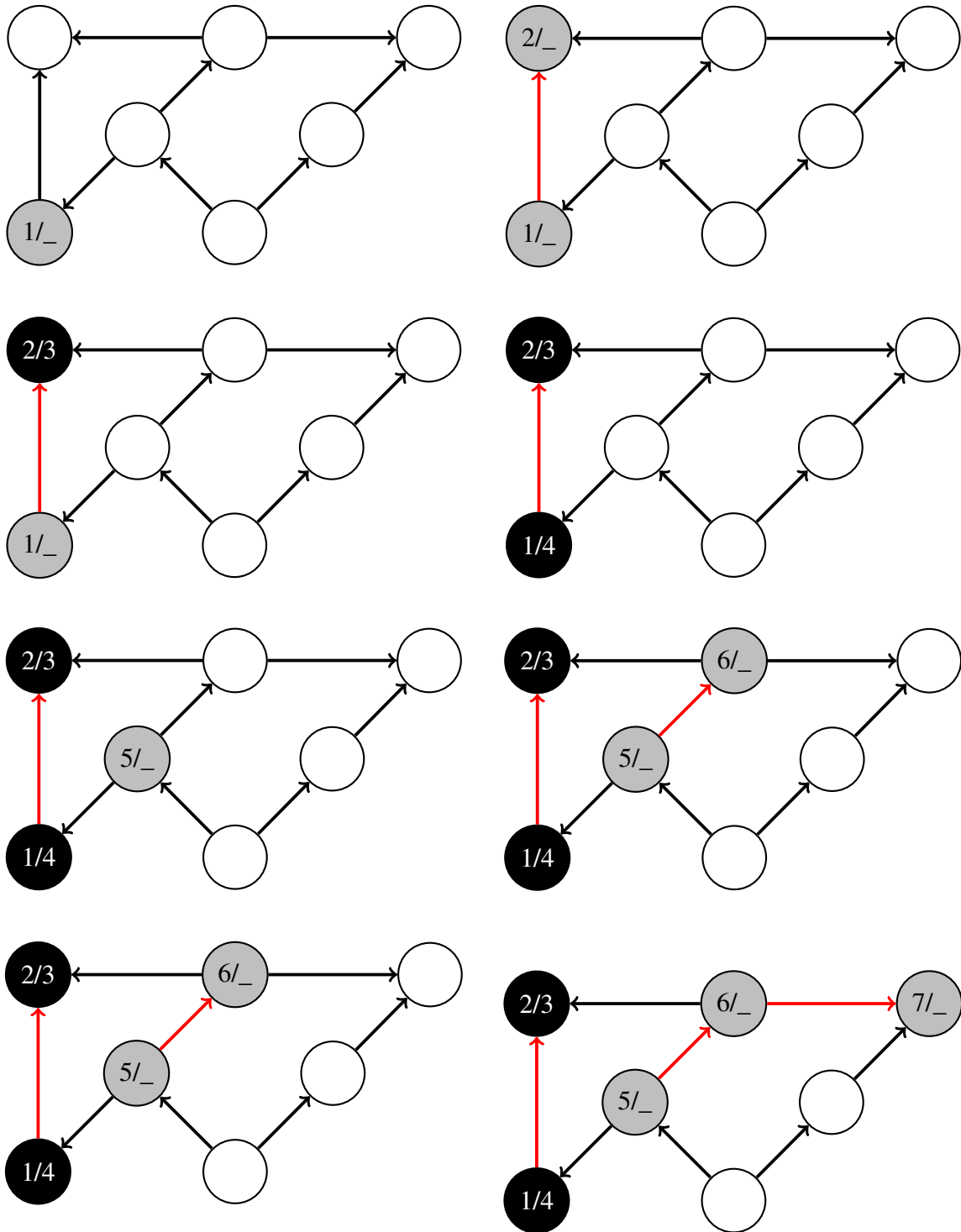


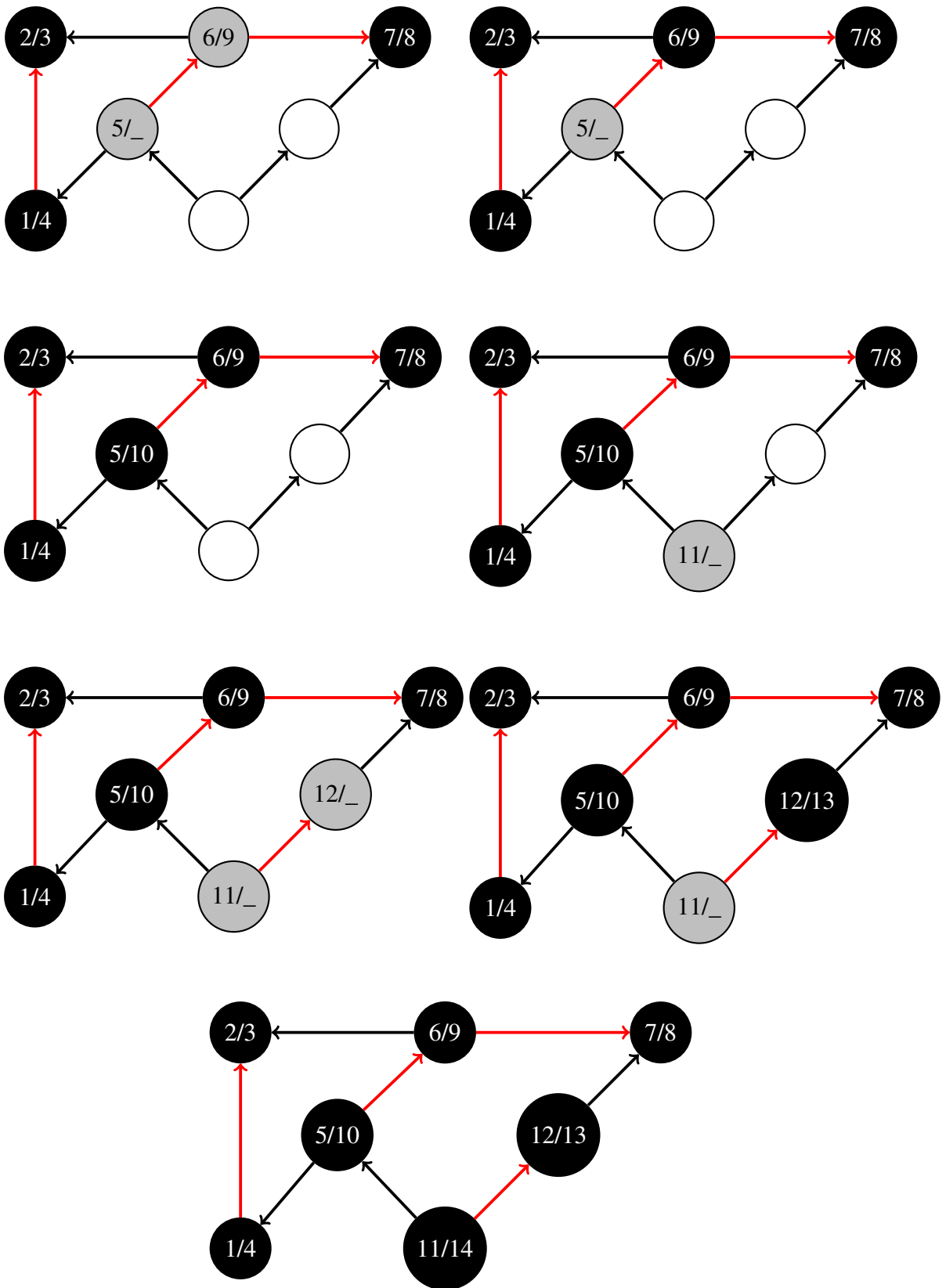


Aufgabe 6 Zyklenfreiheit – DFS (Beispiellösung)

Wir wenden den DFS an. Falls wir eine Kante zu einem bereits grau markierten Knoten finden, enthält der Graph einen Zyklus.

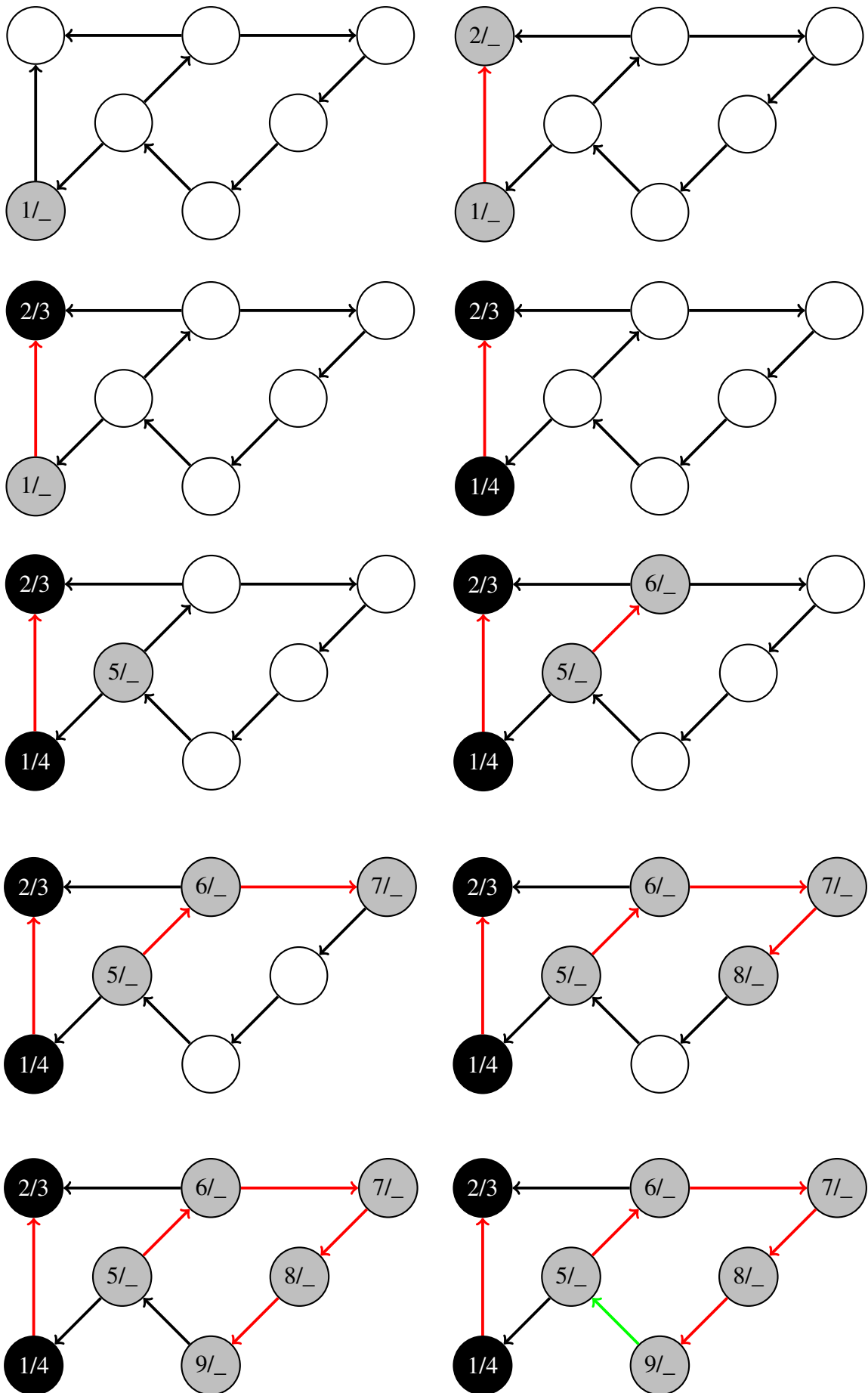
a) Anwendung des DFS ergibt:





Wir haben keine Kante auf einen bereits grau markierten Knoten gefunden. Daher ist der Graph zyklensfrei.

b) Anwendung des DFS ergibt:



Der Graph enthält daher einen Zyklus.