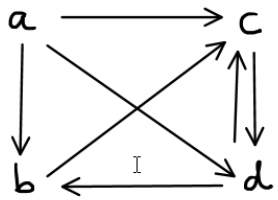
1. (4 Punkte) a. Gib für den abgebildeten Graphen die im Unterricht vorgeschlagenen Implementierung in Python an.
   b Mit welcher Schleife können wir alle Nachbarn des Knoten 'd' durchlaufen?
   c. Welcher boolesche Ausdruck überprüft, ob es eine Kante von 'c' nach 'a' gibt?
   Die Antworten für b. und c. sollen auch für andere Ausprägungen des Graphen gelten.



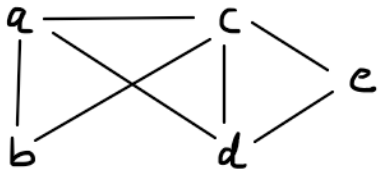
Lösung:

```
a. G = {
'a': set('bcd'),
'b': set('c'),
'c': set('d'),
'd': set('bc'),
}

b. for v in G['d']:
    # do something with v

c. 'a' in G['c']
```

2. (4 Punkte) a. Gib für den abgebildeten Graphen die im Unterricht vorgeschlagene Implementierung in Python an.
   b Mit welcher Schleife können wir alle Nachbarn des Knoten 'c' durchlaufen?
   c. Welcher boolesche Ausdruck überprüft, ob es eine Kante von 'a' nach 'd' gibt?
   Die Antworten für b. und c. sollen auch für andere Ausprägungen des Graphen gelten.



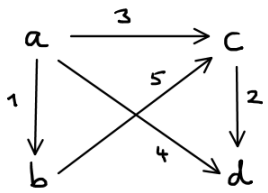
Lösung:

```
a. G = {
'a': set('bcd'),
'b': set('ca'),
'c': set('ade'),
'd': set('ace'),
'e': set('cd')
}

b. for v in G['c']:
     # do something with v

c. 'd' in G['a']
```

3. (5 Punkte) a. Gib für den abgebildeten Graphen die im Unterricht vorgeschlagene Implementierung in Python an.
   b Mit welcher Schleife können wir alle Nachbarn des Knoten 'a' durchlaufen?
   c. Welcher boolesche Ausdruck überprüft, ob es eine Kante von 'b' nach 'a' gibt?
   d. Durch welchen Ausdruck kommen wir an das Gewicht der Kante von 'b' nach 'c'?
   Die Antworten für b. c. und d. sollen auch für andere Ausprägungen des Graphen gelten.



Lösung:

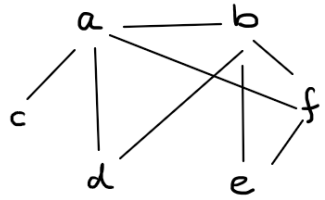
```
a.  G = {
        'a': {'b':1, 'c':3, 'd':4},
        'b': {'c':5},
        'c': {'d':2},
        'd': {}
    }

b.  for v in G['a']:
      # do something with v

c. 'a' in G['b']

d. G['b']['c']
```

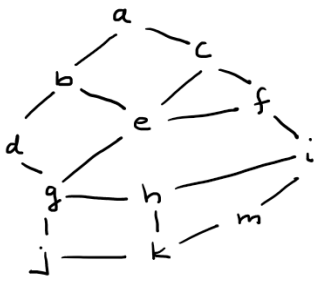
4. (2 Punkte) Der Graph wird von Startknoten a aus mit Tiefensuche traversiert. Die Nachbarn werden in alphabetischer Reihenfolge besucht. Gib die Reihenfolge der besuchten Knoten an.



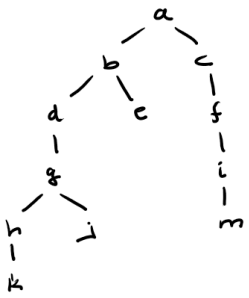
Lösung:

Reihenfolge: a b d e f c

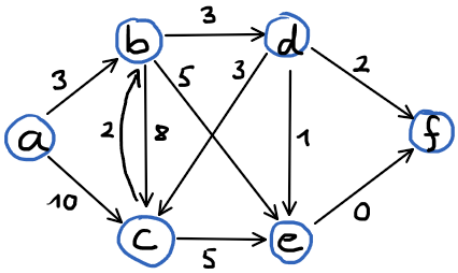
5. (3 Punkte) Zeichne den shortest-path Baum der Breitensuche. Wir beginnen bei a und nehmen an, dass Knoten auf der gleichen Ebene in alphabetischer Reihenfolge besucht werden.



Lösung:



6. (6 Punkte) Ermittle mit dem Algorithmus von Dijkstra die kürzesten Wege von Knoten a zu allen anderen Knoten.
   a. Notiere die Reihenfolge der endgültig markierten Knoten.
   b. Notiere für jeden Knoten die Reihenfolge der Werte, mit denen er markiert wird.



Lösung:

```
Reihenfolge der endgültigen Markierungen: a b d e f c
a : 0
b : inf 3
c : inf 10 9
d : inf 6
e : inf 8 7
f : inf 8 7
```