## Aufgabe 1: (Paralleles Sortieren)

Sortieren Sie die Zahlenfolge 5, 8, 1, 6, 3, 4, 9, 10, 4, 5, 4, 7, 12, 2, 13, 4. Benutzen Sie dazu

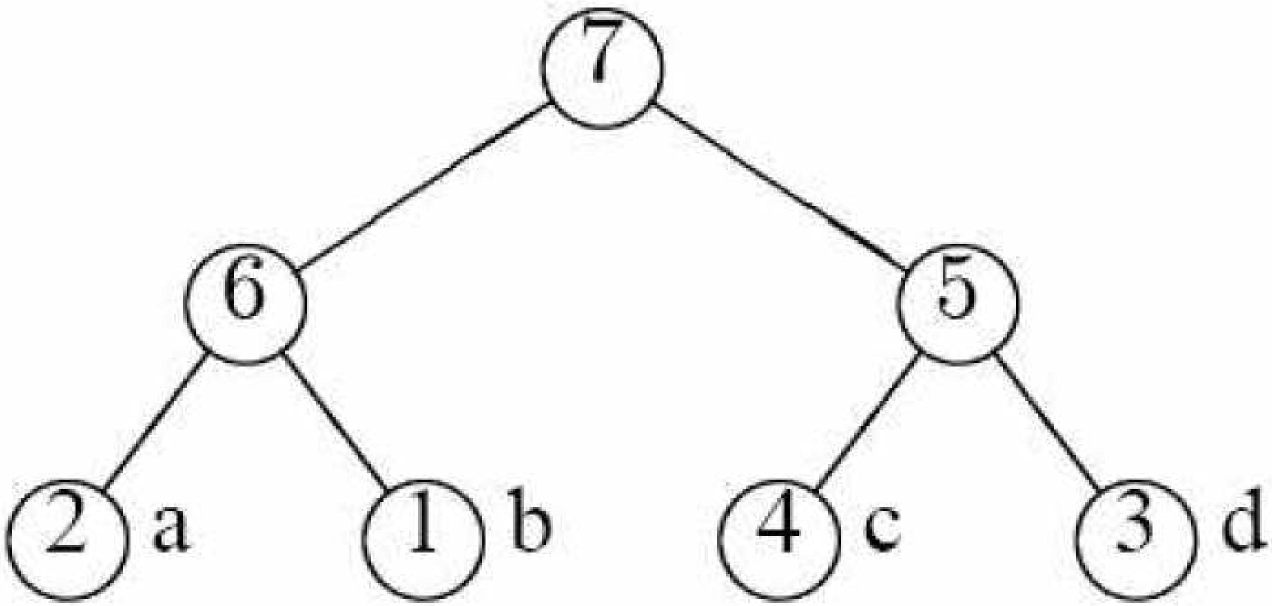
a. das Verfahren der "Linearen Kette",

b. den Divide and Conquer Ansatz (vgl. Folien im Netz).

Wie groß, wie breit und wie tief sind diese parallelen Algorithmen? D.h., wie wieviele Bausteine sind für eine hardwaremäßig Realisierung nötig, wieviel für eine softwaremäßige und wie ist die Laufzeit der Algorithmen.
Geben Sie die Zahlenfolgen bei allen Zwischenschritten an.
Sind diese Verfahren stabil?

## Aufgabe 2

Entfernen Sie das Maximum des Max-Heaps (in Baum-Darstellung)




und stellen Sie die Heapbedingung nach dem Verfahren *Bottom-up-Heapsort* her. Welches Blatt des Baumes wird während des Versickerns besucht? Welche Schlüssel werden beim Einsinken miteinander verglichen?

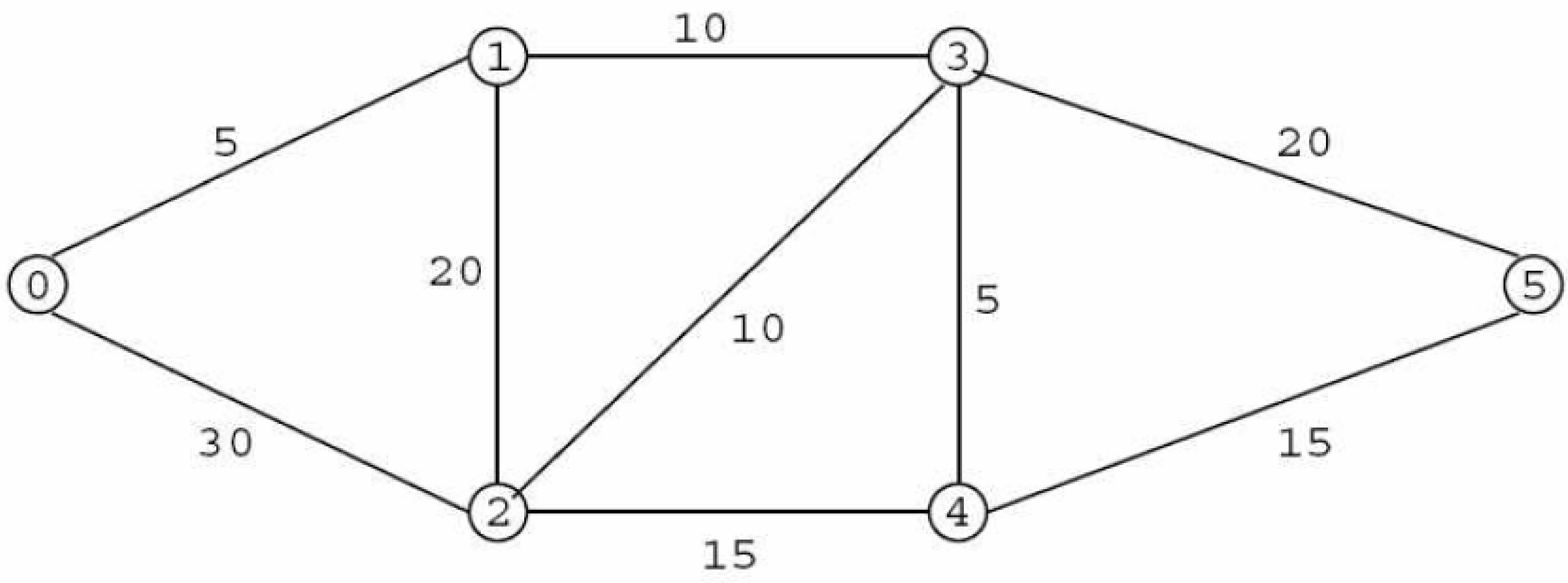
**Aufgabe 3** (Eindeutige Kantengewichte und minimale Spannbäume)

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph mit nicht negativen Kantengewichten $c(e) \geq 0$ für $e \in E$. Ferner gelte $c(e) \neq c(e')$ falls $e \neq e'$, d.h. die Kantengewichte sind paarweise verschieden. Zeigen Sie, dass es genau einen minimalen Spannbaum in $G$ gibt.

## Aufgabe 4

Erstellen Sie aus der Folge 1, 3, 5, 6, 2, 7, 4 einen Heap in Array-Darstellung. Sortieren Sie die Folge mittels *Bottom-up-Heapsort*.

**Aufgabe 5:** (Der Prim-Algorithmus)
Bestimmen Sie einen minimalen spannenden Baum mit dem Primalgorithmus beginnend am Knoten 0.

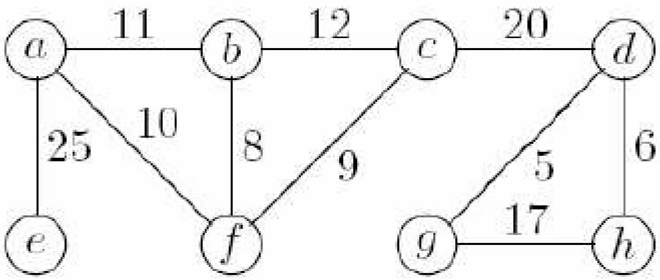


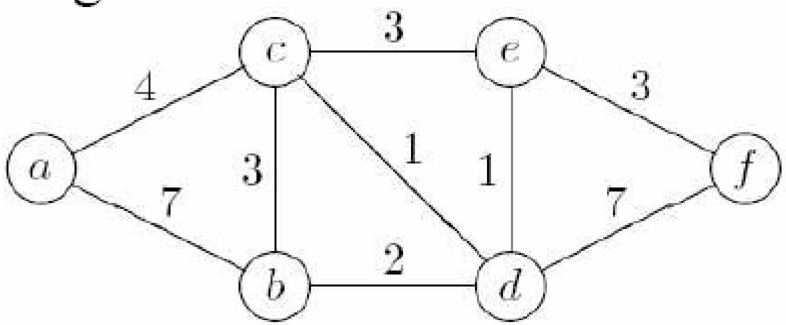
**Aufgabe 6:** (Vergleich Prim und Dijkstra Algorithmus)

a) Bestimmen Sie anhand des Algorithmus von Prim einen MST im gegebenen Graphen.

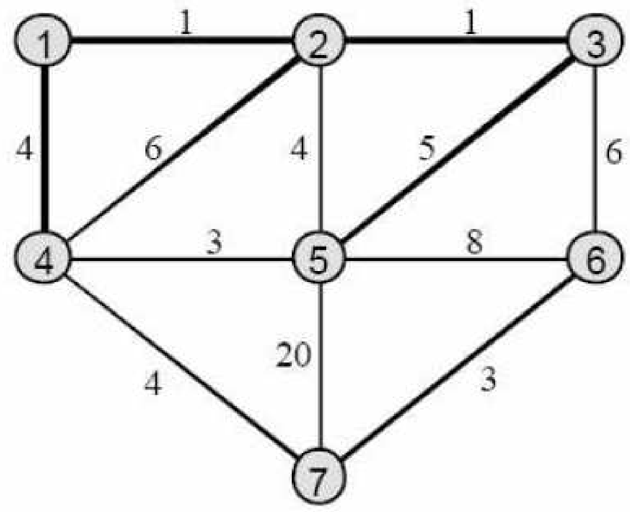
b) Ermitteln Sie mit dem Algorithmus von Dijkstra im gegebenen Graphen die Abstände aller Knoten von a.

Vergleichen Sie die Algorithmen und deren Ergebnisse



**Aufgabe 7:** (Der Algorithmus von Kruskal)

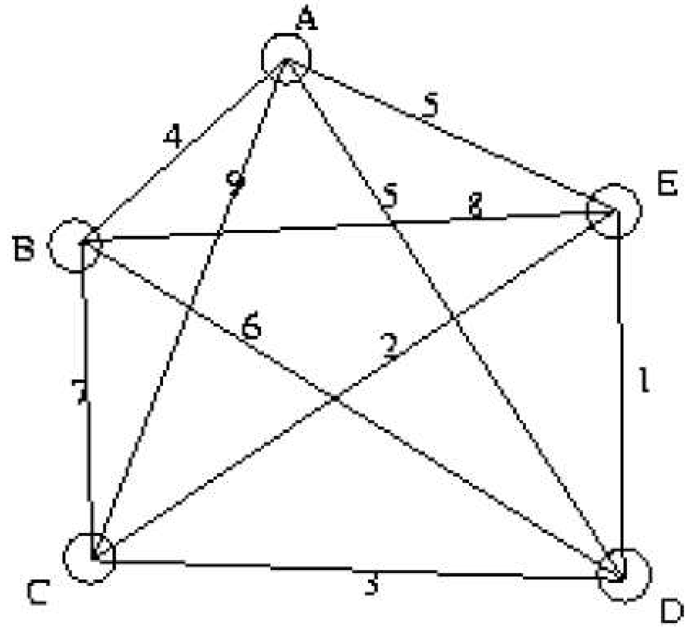



a) Wenden Sie den Algorithmus von *Kruskal* auf den gegebenen Graphen an. Geben Sie dabei in jedem Schritt die gewählte Kante und die Teilgraphen an.

b) Ein Graph kann mehrere verschiedene minimale Spannbäume haben. Ist dies in dem gegebene Beispiel der Fall, und wenn ja, wo zeigt sich diese Möglichkeit im Algorithmus?

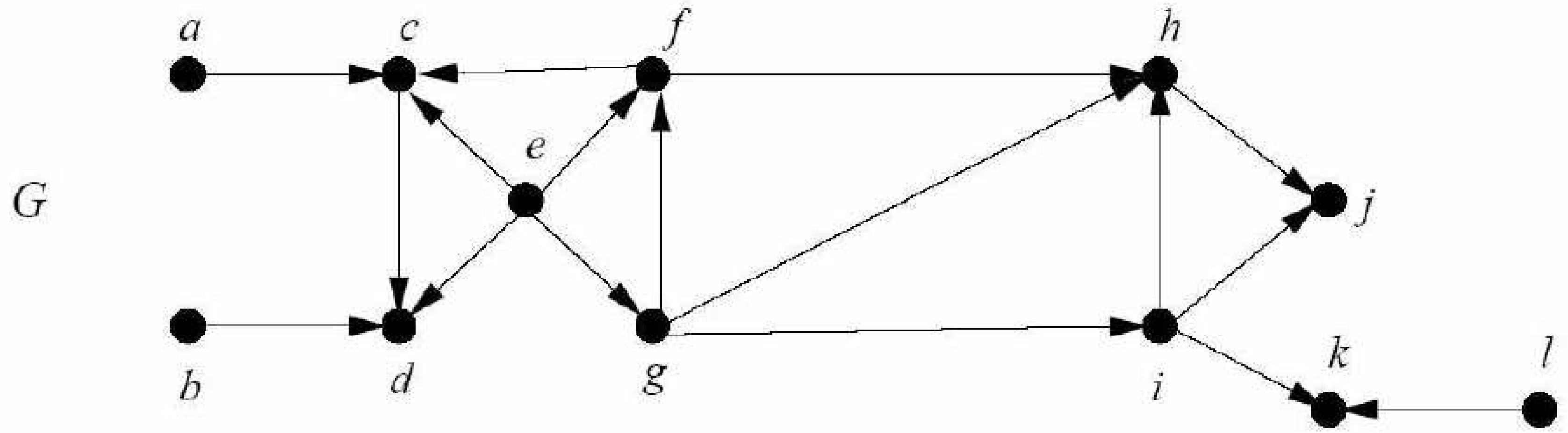
**Aufgabe 8:** (Der Algorithmus von Kruskal)

Verwenden Sie den Algorithmus von Kruskal um einen minimalen spannenden Baum zu folgendem Graph zu finden (Beschreiben Sie die einzelnen Schritte genau).

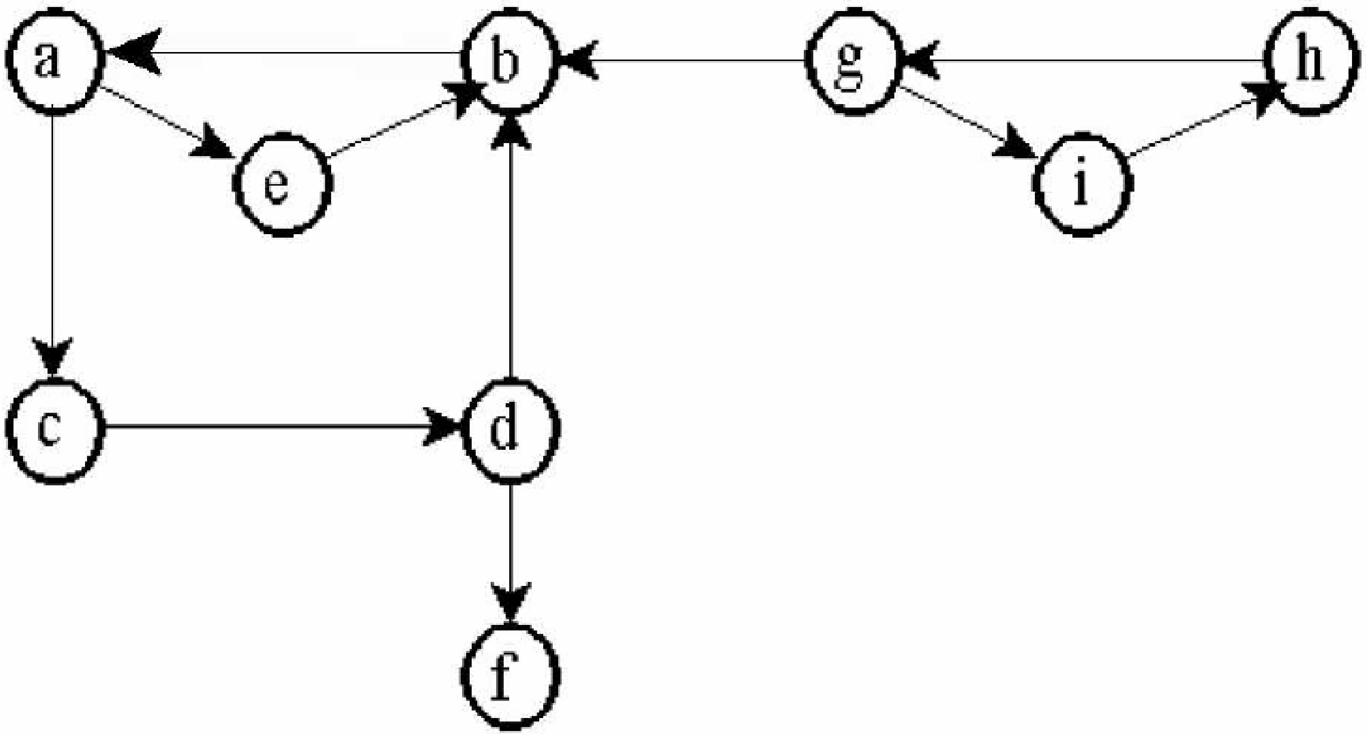


**Aufgabe 9:** (topologisches Sortieren)

Geben Sie für den folgenden azyklischen Graphen eine topologische Sortierung an.

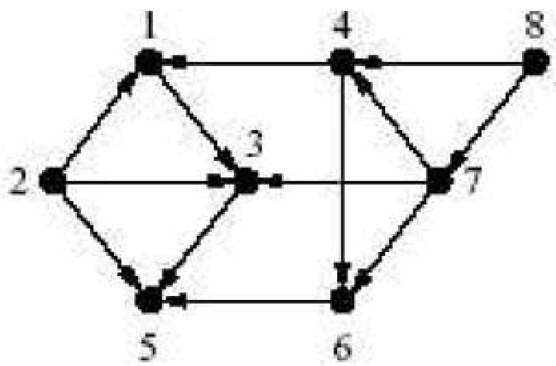


**Aufgabe 10:** (Zusammenhangskomponenten)
Finden Sie alle starken und schwachen Zusammenhangskomponenten des folgenden Graphen:



## **Aufgabe 11** (topologische Sortierung)

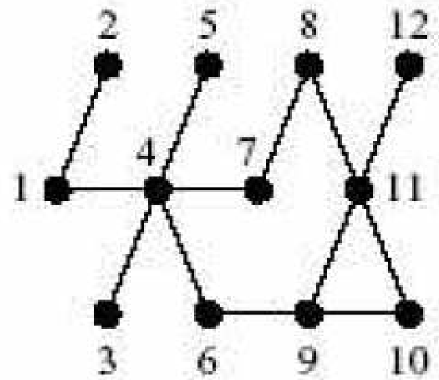
**Definition:** Eine *topologische Sortierung* eines gerichteten Graphen $G = (V, E)$ ist eine Anordnung $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ seiner Knoten $V = \{v_1, v_2, \ldots, v_n\}$, so dass für alle Kanten $(v_{i_j}, v_{i_k})$ gilt $j < k$.




Geben Sie eine topologische Sortierung der Knoten des obigen Graphen an. Welchen Typ (bzgl. Eingangsgrad) besitzt der Knoten, der in der topologischen Sortierung der vorigen Teilaufgabe am Anfang steht? Warum muss das so sein? Zeigen Sie, dass ein gerichteter Graph G genau dann eine topologische Sortierung besitzt, wenn er azyklisch, d. h. kreisfrei, ist.

## **Aufgabe 12** (Tiefensuche)

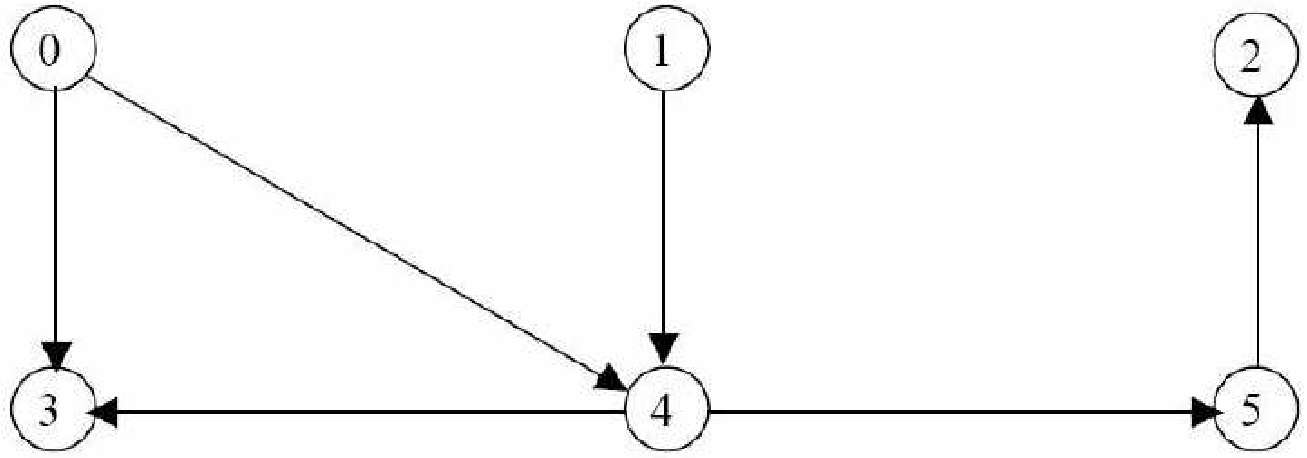
Geben Sie für den gegebenen Graphen die Reihenfolge an, in der die Knoten (und Kanten) bei einer Tiefensuche beginnend bei Knoten 1 durchlaufen werden. Bei Wahlmöglichkeit soll zunächst der Nachbarknoten mit der kleinsten Nummer gewählt werden. Woran kann man bei der Tiefensuche erkennen, dass er einen Kreis enthält?



**Aufgabe 13** (bipartite Graphen)
Zeigen Sie, dass ein ungerichteter Graph genau dann bipartit ist, wenn er keine Kreise ungerader Länge enthält.

**Aufgabe 14** (Zusammenhang von Graphen)
Schreiben Sie (im Pseudocode) einen Algorithmus, der in $O(n + m)$ Laufzeit feststellt, ob ein ungerichteter Graph zusammenhängend ist. Schreiben Sie (im Pseudocode) einen Algorithmus, der in $O(n + m)$ Laufzeit einen aufspannenden Baum für einen Graphen berechnet, wenn es einen solchen gibt und andernfalls ausgibt, dass der Graph nicht zusammenhängend ist.

**Aufgabe 15**
Führen Sie den Algorithmus von Floyd für den Beispielgraphen durch (die Kantengewichte seien überall 1).



**Aufgabe 16** (Transitive Hülle)

Ändern Sie den Algorithmus von Floyd so ab, dass er die transitive Hülle eines Graphen berechnet (Algorithmus von Warshall).

**Aufgabe 17** (Transitive Hülle Teil 2)

Es sei der folgende gerichtete Graph G=(V,E) gegeben:

V = {A,B,C,D,E,F,G,H,I,J,K}
E = {(A,B),(A,D) ,(A,K) ,(B,C) ,(C,H) ,(D,C) ,(E,D) ,(E,F) ,(G,F) ,(H,F) ,(I,H), (J,I) ,(K,J)}

Geben Sie die Adjazenzmatrix dieses Graphen an. Berechnen Sie die transitive Hülle des Graphen mit dem Warshall Algorithmus und stellen Sie das Ergebnis in einer Adjazenzmatrix dar. Wie kann anhand der Adjazenzmatrix der transitiven Hülle einfach entschieden werden, welcher Knoten von allen anderen erreichbar ist und von welchem Knoten aus alle anderen erreichbar sind?

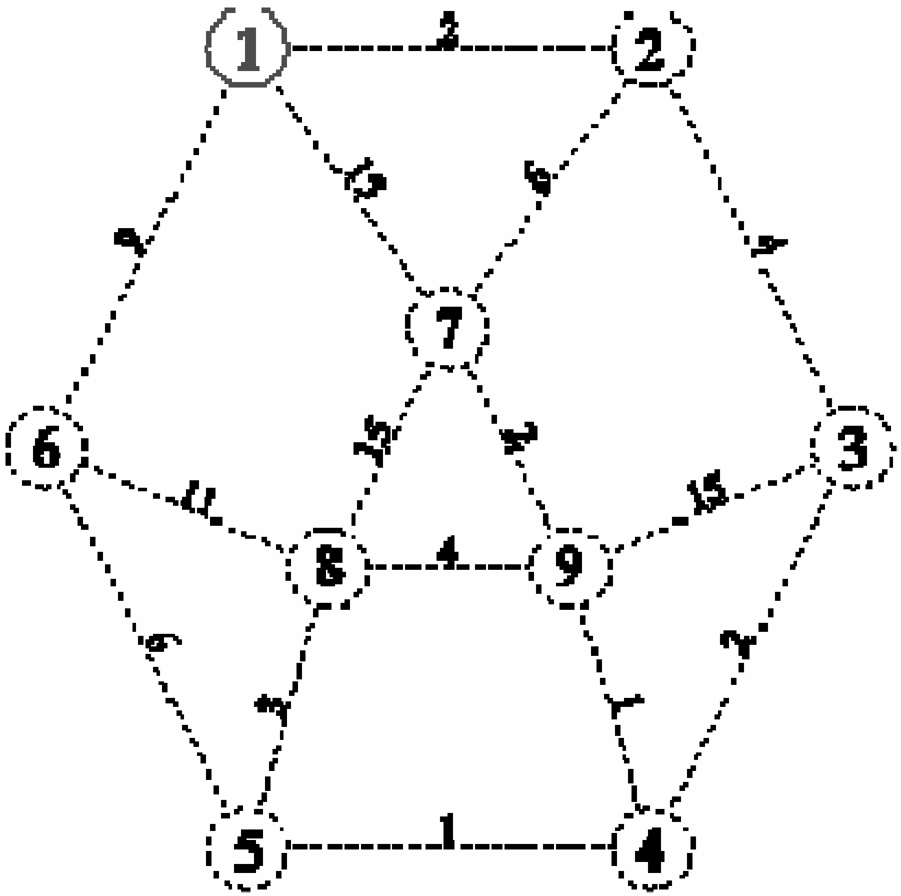
**Aufgabe 18 (**Minimaler Spannbaum (MST), kürzeste Wege**)**

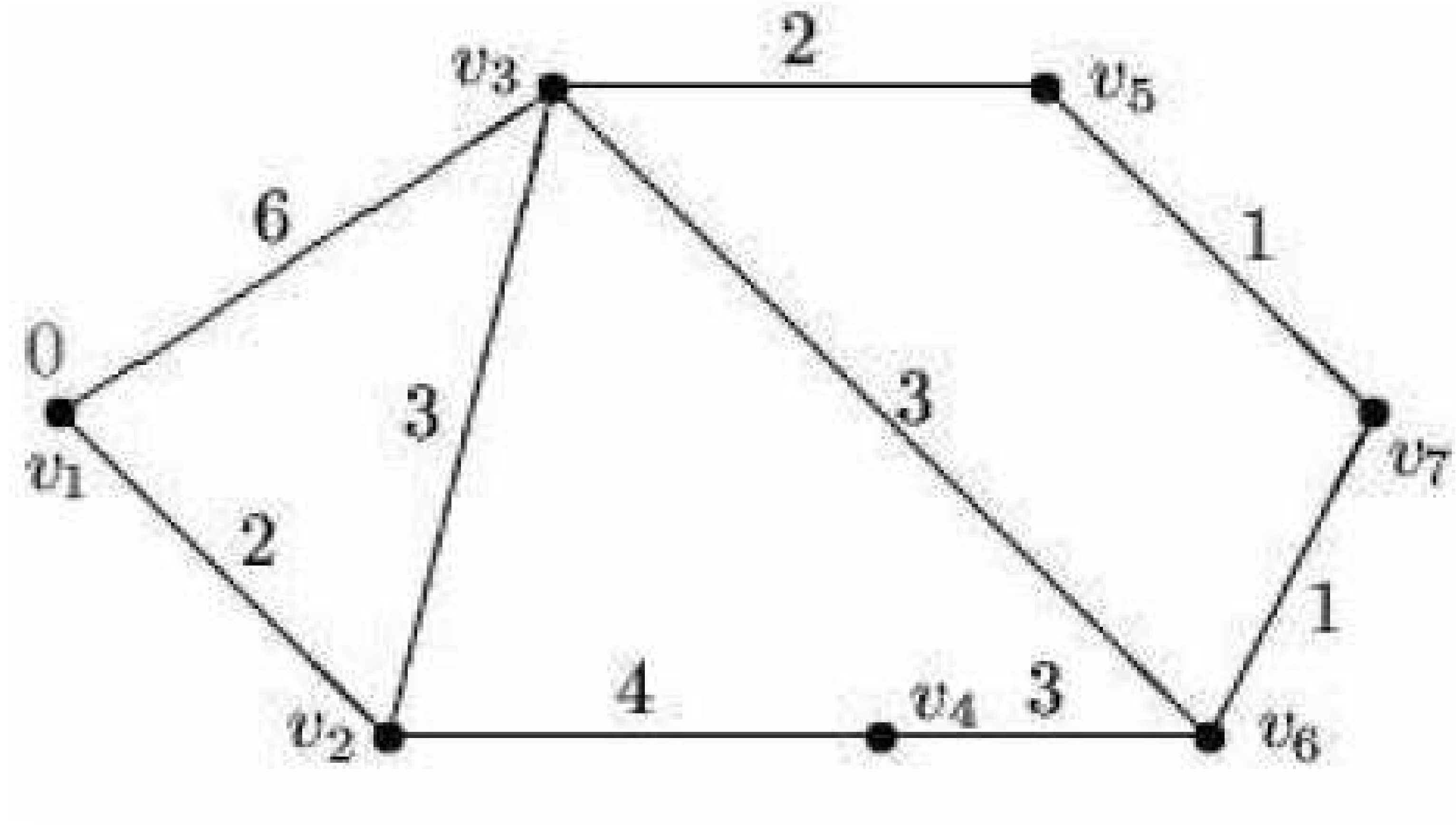
Es sei folgender gewichteter ungerichteter Graph G=(V,E) gegeben:
V = {A,B,C,D,E,F}
E = {(A,B,10),(A,C,4),(A,E,3),(C,D,5),(C,E,2),(E,D,2),(E,F,4),(D,F,3) ,(D,B,1) (F,B,5)}
Berechnen Sie einen minimalen Spannbaum nach dem Kruskal **-** und nach dem Prim Algorithmus. Bestimmen Sie den kürzesten Weg von B nach C mit dem Dijkstra Algorithmus.

**Aufgabe 19 (**kürzeste Wege**)**
Es sei folgender gewichteter **gerichteter** Graph G=(V,E) gegeben:
V = {A,B,C,D,E,F}
E={(A,B,10),(A,C,4),(A,E,3),(C,D,4),(C,E,2),(E,C,0),(E,D,2),(D,E,4),(E,F,4),(D,F,3), (D,B,1) (F,B,5),(B,F,4)}
Bestimmen Sie alle kürzesten Wege (APSP) mit dem Algorithmus von Dijkstra und mit dem Algorithmus von Floyd.

# Aufgabe 20 Anwendung des Algorithmus von Dijkstra

Bestimmen Sie mit dem Algorithmus von Dijkstra einen Kürzeste-Wege-Baum vom Knoten 1 aus




$v_3$
2
$v_5$
6
1
0
3
3
$v_1$
$v_7$
2
1
4
$v_4$
3
$v_2$
$v_6$