Algorithmen und Datenstrukturen
Klausur WS 2017/18

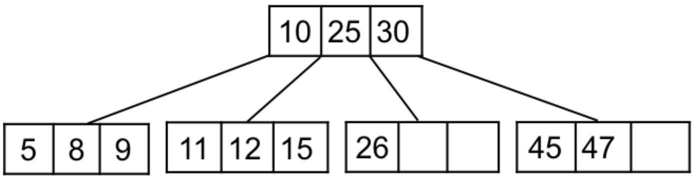
Angewandte Informatik Bachelor

Name	
Matrikelnummer	

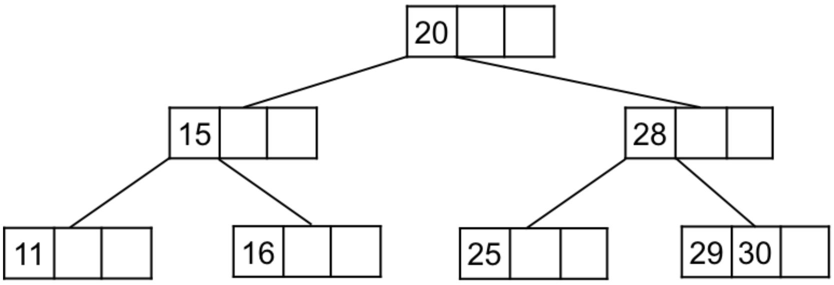
Aufgabe 1	B-Baum	12	
Aufgabe 2	Tiefensuche	18	
Aufgabe 3	Algorithmus von Dijkstra	14	
Aufgabe 4	Algorithmus von Kruskal	16	
Summe		**60**	

Aufgabe 1 B-Bäume (12 Punkte)

a) Fügen Sie in folgendem B-Baum (der Ordnung 4) die Schlüssel 7 und dann 21 ein.



b) Löschen Sie in folgendem B-Baum (der Ordnung 4) die Schlüssel 20 und dann 25.



Aufgabe 2 Tiefensuche (18 Punkte)

Gegeben ist der aus der Vorlesung bekannte rekursive Tiefensuchalgorithmus, der alle Knoten in einem gerichteten Graphen besucht.

Der Algorithmus ist ergänzt um eine Datenstruktur tiefenSucheBeendet (im Algorithmus unterstrichen), in der jeder Knoten v gespeichert wird, nachdem v und alle seine Nachfolger besucht worden sind.

```
Set<Vertex> besucht;
Set<Vertex> tiefenSucheBeendet;

void visitAllNodes(DirectedGraph g) {
        besucht = ∅;
        tiefenSucheBeendet = ∅;
        for (jeden Knoten v)
                if (! besucht.contains(v) )
                        visit(v, g, besucht);
}

void visit(Vertex v, DirectedGraph g) {
        besucht.add(v);    // besuche v
        for ( jeden Nachfolger w von v )
                if ( ! besucht.contains(w) )
                        visit(w, g);
        tiefenSucheBeendet.add(v);
}
```

a) Geben Sie den Tiefensuchwald (TSW) für den gerichteten Graphen auf der folgenden Seite an. Es soll immer dann eine gerichtete Kante (v,w) eingetragen werden, wenn Knoten w von Knoten v aus besucht wird. In den for-Schleifen des Algorithmus werden die Knoten in numerischer Reihenfolge durchlaufen.

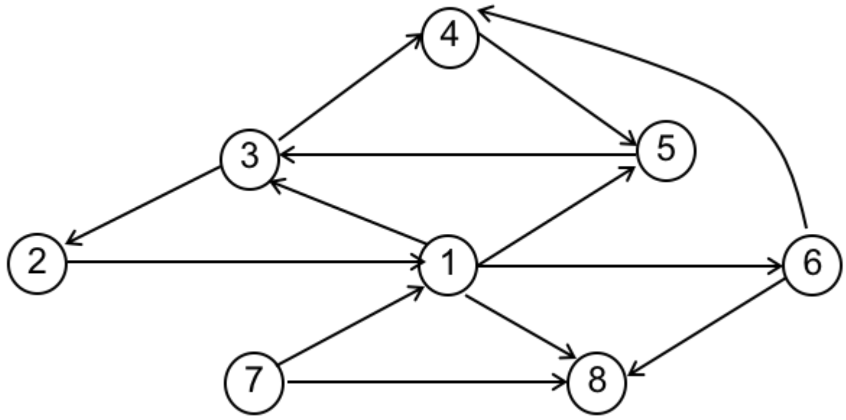
b) Tragen Sie im TSW für jeden Knoten v den Zeitpunkt ein (1, 2, 3, ...) ein, an dem die Tiefensuche für v beendet wurde (d.h. Aufruf von tiefenSucheBeendet.add(v)).

c) Für den Fall, dass ein Knoten v zu einem bereits besuchten Knoten w führt, soll der TSW aus a) durch eine weitere Kante (v,w) ergänzt werden. Unterscheiden Sie drei Fälle:

- Vorwärtskante (v,w): es gibt im TSW bereits einen Weg von v nach w. Zeichnen Sie Vorwärtskanten in blau ein (alternativ Pfeil mit Beschriftung V)
- Rückwärtskante (v,w): es gibt im TSW bereits umgekehrt einen Weg von w nach v. Zeichnen Sie Rückwärtskanten in rot ein (alternativ Pfeil mit Beschriftung R).
- Querkante (v,w): Kanten, die weder Vorwärts- noch Rückwärtskanten sind, sind Querkanten und sollen grün eingezeichnet werden (alternativ Pfeil mit Beschriftung Q)

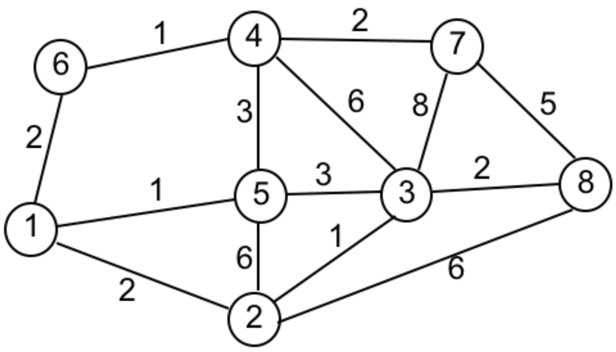
d) Wo enthält der Graph einen Zyklus und durch welche Kante aus c) kann er erkannt werden? Es genügt die Angabe eines Zyklus.

e) Wie kann im Algorithmus mit Hilfe der Datenstruktur tiefenSucheBeendet einen Zyklus erkannt werden? Ergänzen Sie den oben angegebenen Algorithmus.


4
3
5
2
1
6
7
8

Aufgabe 3 Algorithmus von Dijkstra (14 Punkte)

Gegeben ist ein ungerichteter Graph mit Kosten als Gewichte. Bestimmen Sie mit dem Algorithmus von Dijkstra vom Startknoten s = 2 zu allen anderen Knoten jeweils einen günstigsten Weg.



a) Tragen Sie in folgende Tabelle nach jedem Besuchsschritt folgendes ein:

- der besuchte Knoten b
- die Kosten d[v] für den günstigsten Weg von Startknoten s nach v
- den Vorgängerknoten p[v] für den günstigsten Weg von Startknoten s nach v.

Hinweis: Es brauchen nur die d- und p-Werte eingetragen werden, die sich geändert haben. Die endgültigen p- und d-Werte können durch Umrandung besonders gekennzeichnet werden.

b	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]	d[8]		p[1]	p[2]	p[3]	p[4]	p[5]	p[6]	p[7]	p[8]

b) Geben Sie den gefundenen günstigsten Weg von 2 nach 7 an.

c) Welche Kosten hat der günstigste Weg von 2 nach 7?

Aufgabe 4 Algorithmus von Kruskal (16 Punkte)

Ein gewichteter, ungerichteter Graph mit der Knotenmenge V = {1, 2, 3, 4, 5, 6, 7, 8, 9} und 25 Kanten ist durch folgende Adjazenzmatrix gegeben. Zur besseren Lesbarkeit sind die Gewichte nur für eine Kantenrichtung eingetragen.

	1	2	3	4	5	6	7	8	9
1		10	3			6	6		12
2			12	13	8	11	13	14	7
3				15	11	8	4		
4								9	
5						11			2
6							5	16	
7								14	
8									17
9									

a) Bestimmen Sie einen minimal aufspannenden Baum mit dem <u>Algorithmus von Kruskal</u>. Geben Sie dazu für jeden Schritt die ausgewählte Kante und das Gewicht an und skizzieren Sie die dazu-gehörende Union-Find-Struktur für die Knotenmenge V. Verwenden Sie den Union-By-Height-Algorithmus.

Schritt	Kante	Gewicht	Union-Find-Struktur
1			
2			
3			
4			
5			
6			
7			
8			

b) Geben Sie die Datenstruktur (Elternfeld p) für die im letzten Schritt erhaltene Union-Find-Struktur an.

c) Geben Sie den berechneten minimal aufspannenden Baum zeichnerisch an. Wählen Sie den Knoten 1 als Wurzel.