Name:	Klausur-ID:
Vorname:	
Matrikelnummer:	

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Jun.-Prof. D. Hofheinz, Jun.-Prof. H. Meyerhenke 28.09.2015

Klausur Algorithmen I

Aufgabe 1.	Kleinaufgaben	16 Punkte
Aufgabe 2.	Bellman-Ford- und Kruskal-Algorithmus	9 Punkte
Aufgabe 3.	Multiple Choice	7 Punkte
Aufgabe 4.	Optimierung	12 Punkte
Aufgabe 5.	Algorithmenentwurf (Graphen)	10 Punkte
Aufgabe 6.	Duplikate eliminieren	6 Punkte

Bitte beachten Sie:

- Merken Sie sich Ihre **Klausur-ID**.
- **Schreiben** Sie auf **alle Blätter** der Klausur und Zusatzblätter Ihre **Klausur-ID**.
- Die Klausur enthält 16 Blätter.
- Die durch Übungsblätter gewonnenen Bonuspunkte werden erst nach Erreichen der Bestehensgrenze hinzugezählt. Die Anzahl der Bonuspunkte entscheidet nicht über das Bestehen der Klausur.

Aufgabe	1	2	3	4	5	6	Summe
max. Punkte	16	9	7	12	10	6	60
Punkte							

Bonuspunkte:	Summe:	**Note:**

Aufgabe 1. Kleinaufgaben [16 Punkte]

a. Beweisen oder widerlegen Sie:

i) $n^{\log_{10} n} \in \Omega(n^{1000})$,

ii) $n^{n+1} \in O(n^n)$,

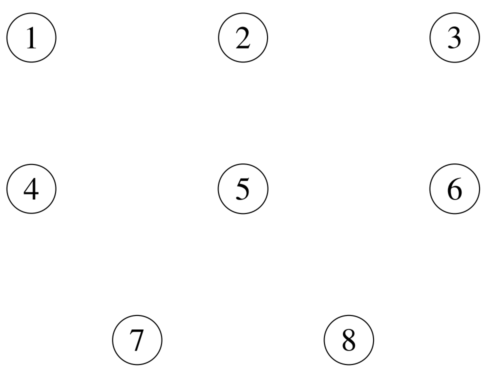
für $n \in \mathbb{N}$. [3 Punkte]

b. Sei das folgende Adjazenzfeld eines gerichteten Graphen $G = (V, E)$ gegeben:

	1	2	3	4	5	6	7	8	9
V	1	3	6	7	7	9	11	12	12

	1	2	3	4	5	6	7	8	9	10	11
E	6	7	1	5	6	2	6	8	3	8	8

Zeichnen Sie G. Benutzen Sie dafür die unten stehende Vorlage, wobei die Kreise die Knoten darstellen. [2 Punkte]



(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 1

c. Nennen Sie einen Vorteil und einen Nachteil des dummy header bei verketteten Listen aus der Vorlesung. [1 Punkt]

d. Nennen Sie einen Vorteil von ganzzahligem gegenüber vergleichsbasiertem Sortieren und einen Vorteil von vergleichsbasiertem gegenüber ganzzahligem Sortieren. [1 Punkt]

e. Zeigen Sie ohne Verwendung des Master-Theorems (etwa mittels Variablenwechsels), dass für

$$T(n) = \begin{cases} 3 & \text{falls } n = 3, \\ T(n^{1/3}) + 1 & \text{falls } n \geq 27, \end{cases}$$

$T(n) \in \mathrm{O}(\log \log n)$ gilt. Dabei sei $n = 3^{3^i}$, für $i \in \mathbb{N} \cup \{0\}$. [4 Punkte]

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 1

f. Fügen Sie die Hashwerte der Elemente $4, 5, 6, 12, 22, 33$ in dieser Reihenfolge mittels linearer Suche (= lineares Sondieren) in die unten gegebene Hashtabelle ein. Dabei sei $h(x) = (2 \cdot x^2 + 3) \bmod 11$ die zu benutzende Hashfunktion. [3 Punkte]

0	1	2	3	4	5	6	7	8	9	10	11

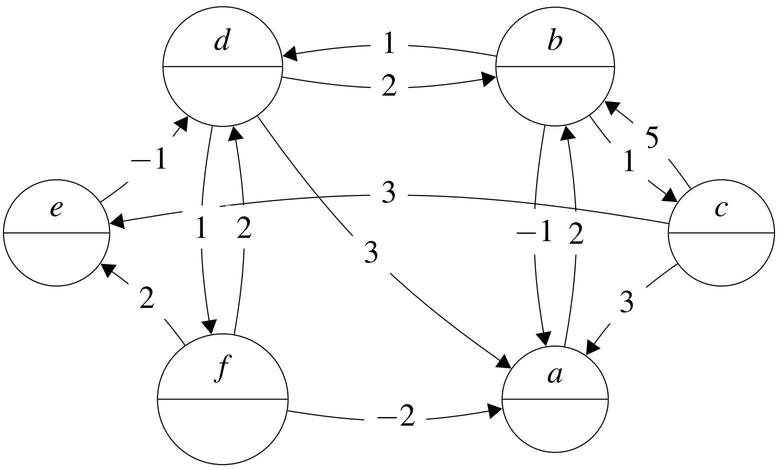
g. Gegeben sei das Feld $A = [9, 2, 3, 1, 4, 6, 4]$. Nutzen Sie Insertionsort aus der Vorlesung, um A aufsteigend zu sortieren. Geben Sie den Zustand von A nach jedem Einfüge-Schritt in der unten gegebenen Tabelle an. [2 Punkte]

9	2	3	1	4	6	4

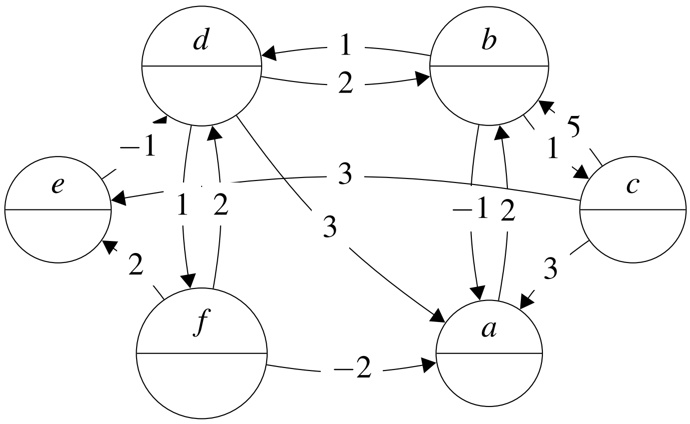
Aufgabe 2. Bellman-Ford- und Kruskal-Algorithmus [9 Punkte]

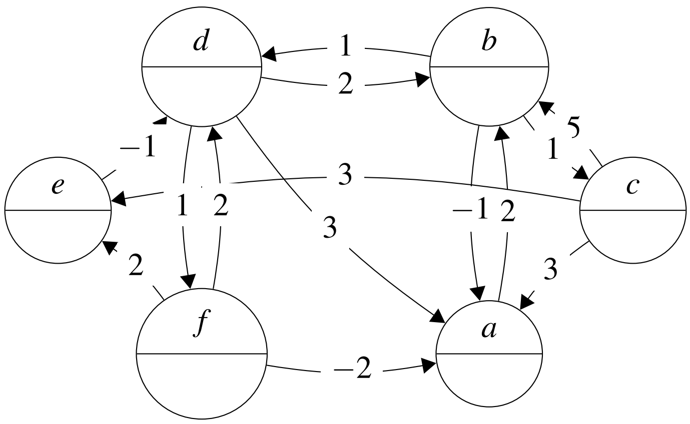
a. Wir betrachten den unten gegebenen Graphen G mit Kantengewichten. Führen Sie den Bellman-Ford-Algorithmus auf G mit Startknoten c aus und tragen Sie dabei die kürzeste Distanz zwischen c und jedem Knoten in G ein. Tragen Sie nur das endgültige Ergebnis ein. In jedem Knoten wurde dafür Platz gelassen. Zeichnen Sie zudem den vom Bellman-Ford-Algorithmus berechneten Baum kürzester Wege in G ein. [4 Punkte]

Der Graph G:



Zwei Kopien von G zum Rechnen:



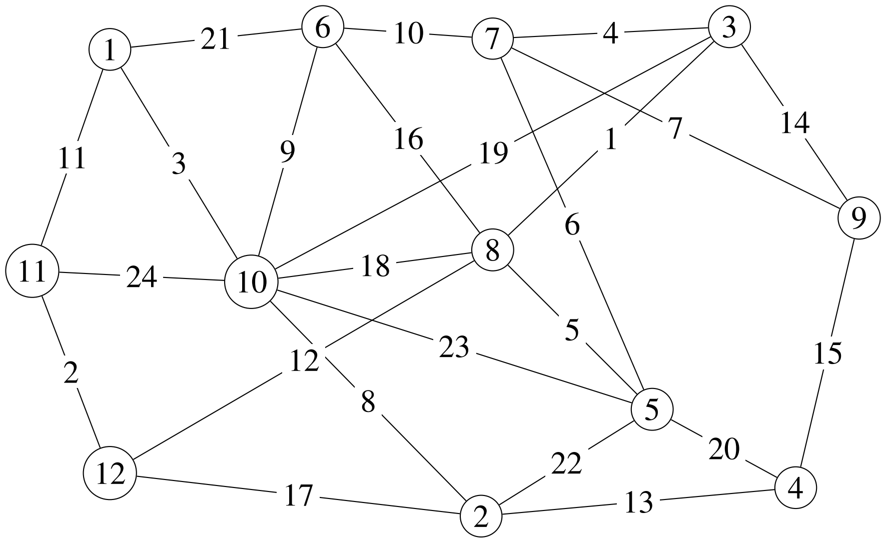


(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 2

b. Wir betrachten den unten gegebenen Graphen $G' = (V, E)$, mit $V = \{1, 2, \ldots, 12\}$. (Die natürlichen Zahlen $1, 2, \ldots, 12$ repräsentieren hier die Knotenlabel. Die Kantengewichte sind eindeutig mit $1, \ldots, 24$ belegt.) Berechnen Sie einen Minimum Spanning Tree (MST) von G' mit dem Algorithmus von Kruskal. Geben Sie jeweils die Kanten des MST in der Reihenfolge an, in der sie vom Algorithmus auswählt werden. Nutzen Sie als Schreibweise für Kanten die folgende Form: (u, v), für Knoten $u, v \in V$. [5 Punkte]

Der Graph G':



1. __________________
2. __________________
3. __________________
4. __________________
5. __________________
6. __________________
7. __________________
8. __________________
9. __________________
10. __________________
11. __________________

Aufgabe 3. Multiple Choice [7 Punkte]

Markieren Sie für jede der folgenden Aussagen, ob sie zutreffend ist. Korrekt markierte Aussagen geben entweder $0,5$ Punkte oder 1 Punkt (wie angegeben), inkorrekt markierte Aussagen geben $0,5$ bzw. 1 Punkt Abzug. Aussagen, die nicht markiert sind, ergeben weder Punkte noch Abzug. Eine negative Gesamtpunktzahl in dieser Aufgabe wird als 0 Punkte gewertet.

Aussage	**ja**	**nein**
Eine doppelt verkettete Liste verhält sich in allen Anwendungen zeiteffizienter als eine einfach verkettete Liste. *(0,5 Punkte)*		
Eine einfach verkettete Liste verhält sich in allen Anwendungen zeiteffizienter als eine doppelt verkettete Liste. *(0,5 Punkte)*		
Ein Sentinelelement (oder Wächterelement) hilft, Fallunterscheidungen in Suchalgorithmen zu vermeiden. *(0,5 Punkte)*		
Amortisiert haben insert- und remove-Operationen bei (a,b)-Bäumen konstante Laufzeit. *(0,5 Punkte)*		
Der Heapsort-Sortieralgorithmus aus der Vorlesung funktioniert vollständig *inplace*. *(0,5 Punkte)*		
Dijkstras Algorithmus arbeitet nur auf Graphen, die gerichtet und azyklisch sind und positive Kantengewichte haben. *(0,5 Punkte)*		
Mit einem Adjazenzfeld lässt sich für ein gegebenes Knotenpaar (u,v) in worst-case-Laufzeit $O(1)$ entscheiden, ob u und v benachbart sind. *(1 Punkt)*		
In einem DAG („Directed Acyclic Graph“, also „gerichteter azyklischer Graph“) ist die Summe der Eingangsgrade aller Knoten immer doppelt so groß wie die Summe der Ausgangsgrade aller Knoten. *(1 Punkt)*		
Die Ackermann-Funktion ist algorithmisch berechenbar. *(1 Punkt)*		
Es ist ein Polynomialzeitalgorithmus für das Lösen ganzzahliger linearer Programme (ILPs) bekannt. *(1 Punkt)*		

Aufgabe 4. Optimierung [12 Punkte]

Nehmen Sie an, Sie programmieren einen Parkhaus-Automaten, der Wechselgeld als Münzen zurückgibt. Der Automat soll dabei für jeden Rückgabebetrag **möglichst wenige** Münzen zurückgeben.

a. Die Rückgabewährung habe die Münzen $\{1,2,5,10,20,50\}$. Geben Sie für dieses Münzsystem M_1 für die Eingaben (= Rückgabebeträge) $W_1 = 34$ bzw. $W_2 = 96$ an, in welchem Schritt der unten stehende Greedy-Algorithmus welche Münze in die Multimenge A einfügt! [2 Punkte]

Greedy-Algorithmus für Eingabe W
`while` $(W > 0)$ `do`
  Wähle Münze mit größtmöglichem Betrag $b \leq W$ und füge sie in die Multimenge A ein
  $W \leftarrow W - b$
return A

Lösungen bitte in die Tabelle eintragen
(ggf. müssen für eine korrekte Lösung nicht alle Positionen ausgefüllt werden):

Rückgabebetrag	1. Münze	2. Münze	3. Münze	4. Münze	5. Münze	6. Münze
34						
96						

b. Für das Münzsystem M_1 in a) liefert der Greedy-Algorithmus immer das optimale Ergebnis, also die Lösung mit den wenigsten Münzen.
Geben Sie für das veränderte (und fiktive) Münzsystem $M_2 = \{1,4,8,14,25\}$ eine Instanz bestehend aus

- Rückgabebetrag,
- Lösung des Greedy-Algorithmus und
- optimaler Lösung

an, bei der die Lösung des Greedy-Algorithmus nicht optimal ist. Ihre Lösung darf jeweils aus weniger als 6 Münzen bestehen. [2 Punkte]

Algo	1. Münze	2. Münze	3. Münze	4. Münze	5. Münze	6. Münze
Greedy						
Optimal						

Zur Lösung in der Tabelle passender Rückgabebetrag:

c. Sie sollen in Teilaufgabe d.) einen Algorithmus entwerfen, der das Problem für beliebige ganzzahlige Münzsysteme optimal löst. Als Lösung soll der Algorithmus dabei nun die **minimale Zahl der benötigten Münzen** berechnen, keine Multimenge bestehend aus den zugehörigen Münzen!

Geben Sie in dieser Teilaufgabe zunächst eine Rekurrenzgleichung an, mit der sich die Lösung rekursiv darstellen lässt! Schreiben Sie dazu, mit welchem Buchstaben Sie was modellieren (**Beispiel:** W: Rückgabebetrag)! [3 Punkte]

d. Vervollständigen Sie nun die durch Linien gekennzeichneten Lücken im Pseudocode derart, dass der Algorithmus die optimale Lösung (genauer: die Zahl der benötigten Münzen) für jedes Münzsystem M in der Zeit $O(W \cdot |M|)$ und mit Platzverbrauch $O(W)$ berechnet! Verwenden Sie im Pseudocode keine O-Notation! [5 Punkte]

MinimumChange(W)

numCoins $\leftarrow$ neues Array der Länge __________

numCoins[0] $\leftarrow$ __________

```
for w ← 1 to __________ do
    numCoins[w] ← ∞
    for m ∈ M do
        if (_____________) and (________________________________________) then
            numCoins[__________] ← _____________________________________________
return numCoins[__________]
```

Aufgabe 5. Algorithmenentwurf (Graphen) [10 Punkte]

a. Gegeben sei ein ungerichteter, zusammenhängender Graph G. Mittels einer modifizierten Tiefensuche kann geprüft werden, ob G einen Kreis enthält. Der folgende Algorithmus verwendet die Farben *white* und *grey*, um die Knoten während der Tiefensuche geeignet einzufärben. Ergänzen Sie den Pseudocode in den markierten Lücken, sodass der Algorithmus das gegebene Problem löst.

function findCycle($G = (V, E)$):

  color all nodes white

  node $s \leftarrow$ any node $\in V$

  return dfs(s,s)

function dfs(u,w): // starte dfs bei u, kommend von Vorgänger w

  bool found $\leftarrow$ false

  color u grey

  `foreach` $\{u, v\} \in E$ `do`

    `if` ____________________ **and** ____________________ `then`

      ____

    `else`

      found $\leftarrow$ ____________________ **or** found

  ____

[5 Punkte]

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 5

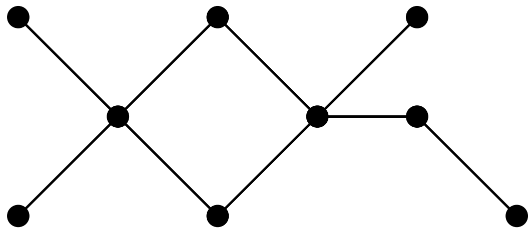


Abbildung 1: Dieser Graph ist 2∃-zusammenhängend.

b. Ein zusammenhängender, ungerichteter Graph $G = (V, E)$ heißt 2∃-zusammenhängend, falls eine Kante existiert, die entfernt werden kann, ohne den Zusammenhang zu zerstören. Beschreiben Sie präzise, wie sich G in $O(|E|)$ Zeit auf diese Eigenschaft prüfen lässt **und** beweisen Sie, dass Ihr Algorithmus das korrekte Ergebnis liefert. [5 Punkte]

Aufgabe 6. Duplikate eliminieren [6 Punkte]

Gegeben sei ein Array A von n Fließkommazahlen. In A sollen Duplikate gefunden und eliminiert werden. Die Ausgabe soll ein neues Array A' sein, das jedes Element aus A nur einmal enthält. Geben Sie in Teilaufgabe a) und b) jeweils einen Algorithmus für diese Problemstellung in Pseudocode an und verwenden Sie die Funktionalität von aus der Vorlesung bekannten Algorithmen und Datenstrukturen.

a. Geben Sie einen Algorithmus mit Laufzeit $O(n \log n)$ an. [3 Punkte]

b. Geben Sie einen Algorithmus mit erwarteter Laufzeit $O(n)$ an. [3 Punkte]

Konzeptpapier für Nebenrechnungen.