Name:
Vorname:
Matrikelnummer:

Klausur-ID:

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders

28.7.2014

Klausur Algorithmen I

Aufgabe 1.	Kleinaufgaben	15 Punkte
Aufgabe 2.	Alle Wege	9 Punkte
Aufgabe 3.	Pseudocode-Analyse	5 Punkte
Aufgabe 4.	4-äre Heaps	12 Punkte
Aufgabe 5.	Shuffle	8 Punkte
Aufgabe 6.	Minimale Spannbäume	11 Punkte

Bitte beachten Sie:

- Als Hilfsmittel ist nur **ein** DIN-A4-Blatt mit Ihren **handschriftlichen** Notizen zugelassen.
- Merken Sie sich Ihre **Klausur-ID** auf dem Aufkleber für den Notenaushang.
- **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 Bonuspunkte entscheidet nicht über das Bestehen.

Aufgabe		1	2	3	4	5	6	Summe
max. Punkte		15	9	5	12	8	11	60
Punkte	EK							
	ZK							

Bonuspunkte:	Summe:	**Note:**

Aufgabe 1. Kleinaufgaben [15 Punkte]

a. Sie sind Software Engineer beim sozialen Netzwerk Giigle Minus. Ihr Chef möchte von Ihnen einen Algorithmus, der zu zwei gegebenen Personen p, p' des Netzwerks die gemeinsamen Freunde im Netzwerk ausgibt. Sie können davon ausgehen, dass das Netzwerk als ungerichter Graph $G = (\{p_1, p_2, p_3, \ldots, \}, E)$ in Adjazenzarray-Darstellung gegeben ist. Es gilt $\{p, p'\} \in E$ genau dann, wenn Person p mit Person p' in dem Netzwerk befreundet ist.

Skizzieren Sie einen Algorithmus, der das Problem für zwei gegebene Personen p, p' in erwartet $\mathrm{O}(\deg(p) + \deg(p'))$ löst. Dabei bezeichnet $\deg(v)$ den Knotengrad eines Knotens v. Begründen Sie kurz die Laufzeit Ihres Algorithmus.
Hinweis: Beachten Sie, dass das Adjazenzarray nicht notwendigerweise sortiert ist und alle zusätzlichen Speicherzellen vor Verwendung initialisiert werden müssen. [3 Punkte]

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 1

b. Nennen Sie die asymptotische Anzahl von *Swaps* und *Vergleichen* im *best-case* und *worst-case* für Insertion Sort auf einem Array mit n verschiedenen Elementen, wenn die Einfügestelle mit *binärer Suche* bestimmt wird. [4 Punkte]

	best-case	worst-case
Swaps:		
Vergleiche:		

c. Notieren Sie die folgenden Funktionen aufsteigend sortiert nach asymptotischen Wachstum, und begründen Sie Ihre Behauptung.

$$f_1(n) = \frac{n^2}{\ln n^{2/3}} \qquad f_2(n) = n \cdot \ln n^{3/2} \qquad f_3(n) = \frac{3}{2} \cdot n^{3/2}$$

[3 Punkte]

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 1

d. Beschreiben Sie kurz den Begriff *erwartete* Laufzeit. [1 Punkte]

e. Tragen Sie die Namen der folgenden Algorithmen und Datenstruktur-Operationen aus der Vorlesung in die entsprechenden Felder nach ihrer asymptotischer *worst-case Zeitkomplexität*. [4 Punkte]

- *quickSelect()* auf einem Array der Länge n.
- *LSDRadixSort()* auf einem Array der Länge n, das 32-bit Ganzzahlen enthält.
- *splice()* auf einer doppelt verketteten Liste der Länge n.
- *buildAddressableHeap()* auf einem Array der Länge n.
- *isDAG()* auf einem gerichteten Graphen mit n Knoten und $4n$ Kanten.
- *bellmanFord()* auf einem gerichteten Graphen mit n Knoten und $2n$ Kanten.

$\Theta(1)$	
$\Theta(\log n)$	
$\Theta(n)$	
$\Theta(n \log n)$	
$\Theta(n^2)$	

Aufgabe 2. Alle Wege [9 Punkte]

a. Gegeben sei ein gerichteter Graph $G = (V, E)$ mit ungewichteten Kanten. Entwerfen Sie einen Algorithmus in Pseudocode, der in $O(|V| + |E|)$ für zwei gegebene Knoten $s, t \in V$ die *Anzahl der kürzesten Wege* von s nach t bestimmt. Sie können hierfür das folgende Breitensuchschema verwenden und an den gekennzeichneten Stellen Pseudocode hinzufügen, oder auf extra Blättern einen eigenen Algorithmus entwickeln. [4 Punkte]

Breitensuchschema für $G = (V, E)$ **mit Startknoten** $s \in V$

$d = \langle\infty, \ldots, \infty\rangle$: **Array of** $\mathbb{N}_0 \cup \{\infty\}$; $d[s] := 0$

parent$= \langle\perp, \ldots, \perp\rangle$: **Array of** NodeId; parent$[s] := s$

$Q = \langle s\rangle, Q' = \langle\rangle$: **Set of** NodeId

⓪

for ($\ell := 0$; $Q \neq \langle\rangle$; ℓ++)

①

foreach $u \in Q$ **do**

②

foreach $(u, v) \in E$ **do**

③

if parent$[v] = \perp$ **then**

④

$Q' := Q' \cup \{v\}$; $d[v] := \ell + 1$; parent$[v] := u$

⑤

⑥

⑦

$(Q, Q') := (Q', \langle\rangle)$

⑧

⑨

return $\boxed{}$

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 2

b. Beweisen Sie die *Korrektheit* und *Laufzeit* Ihres Algorithmus aus Teilaufgabe b). [5 Punkte]

Aufgabe 3. Pseudocode-Analyse [5 Punkte]

Gegeben sei der nachfolgende Pseudocode.

```
Function calculate(A : Array [1..n] of BigNumber) : BigNumber
    calculateRec(A, 1, n)

Function calculateRec(A : Array [1..n] of BigNumber, l : ℤ, r : ℤ) : BigNumber
    s := 0
    if l ≤ r then
        i := l
        while i ≤ r do
            if A[i] ≥ 0 then
                s := s ⊕ A[i]
            else
                s := s ⊖ A[i] ⊕ 2014728
            i := i + 1
        s := s ⊕ calculateRec(A, l + 1, r)
    return s
```

In den folgenden Teilaufgaben analysieren wir die *exakte Anzahl* der arithmetischen Operationen $\oplus$ (Addition) und $\ominus$ (Subtraktion) von `BigNumbers`, die ein Aufruf von calculate(A) für ein Array A von `BigNumber` mit n Elementen benötigt.

a. Charakterisieren Sie sowohl die *best-* als auch die *worst-case* Eingaben der Größe n für die Funktion calculate(). [1 Punkt]

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 3

b. Geben Sie für *best-case* Eingaben eine Rekurrenz $T_{bc}(n)$ und für *worst-case*-Eingaben eine Rekurrenz $T_{wc}(n)$ an. *Hinweis:* Denken Sie auch an die Anfangswerte. [2 Punkte]

c. Bestimmen und beweisen Sie für $T_{wc}(n)$ eine geschlossene Form. [2 Punkte]

Aufgabe 4. 4-äre Heaps [12 Punkte]

In der Vorlesung wurden binäre Heaps als Implementierung von Prioritätswarteschlangen behandelt. Im Folgenden betrachten wir 4-äre Heaps. Hier hat jeder Knoten maximal vier anstatt maximal zwei Kinder. Wie binäre Heaps werden auch 4-äre Heaps implizit repräsentiert, indem die Einträge ebenenweise in ein Array geschrieben werden. Sei ein solches Array $h[1..n]$ gegeben. Dann gibt die Funktion $\text{parent}(i)$ den Elternknoten des Knotens i zurück, und $\text{child}(i, j)$, $(1 \leq j \leq 4)$ gibt das j-te Kind des Knotens i. Daraus ergibt sich folgende
Invariante: $\forall i > 1 : h[\text{parent}(i)] \leq h[i]$

a. Für folgenden 4-ären Heap ist die Invariante verletzt. Markieren Sie die betroffene(n) Position(en) i, indem Sie ein **X** unter den Eintrag schreiben. [1 Punkte]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	11	34	17	47	15	16	23	31	32	70	81	76	62	57	83

b. In einen anfangs leeren 4-ären Heap werden nacheinander folgende Elemente eingefügt:
92, 98, 48, 70, 22, 95, 76, 36, 17, 55, 94.
Anschließend wird eine deleteMin-Operation durchgeführt. Geben Sie den Zustand des Feldes h an, nach Ausführen

1. der ersten fünf insert-Operationen,
2. aller insert-Operationen sowie
3. der anschließenden deleteMin-Operation.

Geben Sie auch den Rückgabewert der deleteMin-Operation an. Nutzen Sie den freien Platz bzw. Konzeptpapier zum Berechnen und die drei vorbereiteten Felder zum Angeben Ihrer Lösungen. Beachten Sie die *Reihenfolge* dieser insert-Operationen. [3 Punkte]

1	2	3	4	5	6	7	8	9	10	11

1	2	3	4	5	6	7	8	9	10	11

1	2	3	4	5	6	7	8	9	10	11

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 4

c. Die Wurzel des Heaps sei in Ebene 0. Wieviele Elemente e_k sind in der k-ten vollständig gefüllten Ebene? Wieviele Elemente d_k sind in den Ebenen davor?
Hinweis: $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$ für alle $a \in \mathbb{R}, n \in \mathbb{N}$. [1 Punkt]

d. Implementieren Sie die Funktionen parent(i) und child(i, j), indem Sie geschlossene Formeln angeben und herleiten.
Hinweis: Stellen Sie zunächst eine Formel auf, welche die Ebene eines Indizes zurückgibt. [5 Punkte]

e. Implementieren Sie insert(e). Nutzen Sie Funktionen parent(i) und child(i, j) statt expliziter Indexberechnung. [2 Punkte]

Aufgabe 5. Shuffle [8 Punkte]

Gegeben seien drei Wörter A, B, und C der Länge $|A| = n$, $|B| = m$ und $|C| = n+m$.
Das Wort C heißt *Shuffle* der Wörter A und B genau dann, wenn C durch eine Verzahnung der Buchstaben von A und B gebildet werden kann, wobei die Reihenfolge der Buchstaben der beiden Wörter erhalten bleiben muss.

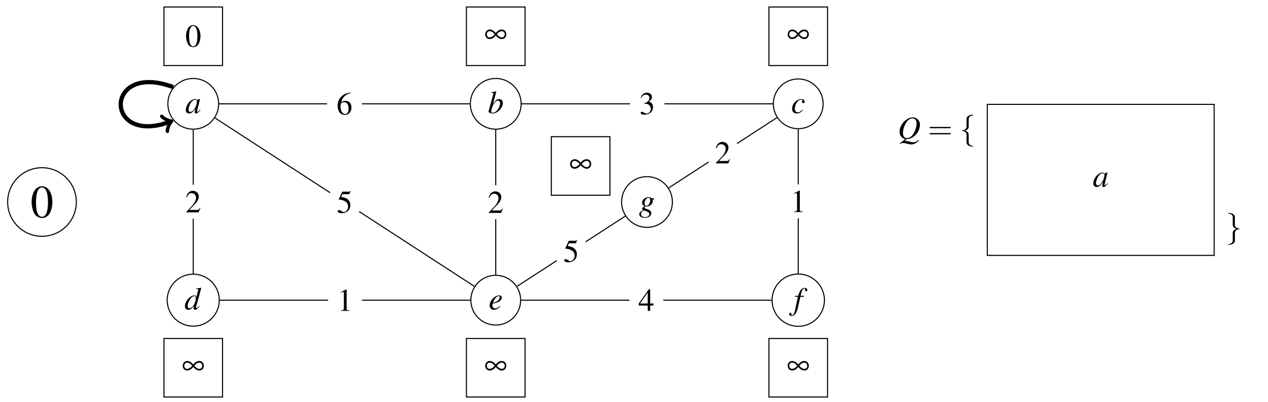
a. Gegeben seien die Wörter $A =$*fußball* und $B =$*weltmeister*. Zeigen Sie oder widerlegen Sie, dass die folgenden beiden Wörter C_1 und C_2 jeweils ein Shuffle der Wörter A und B sind:
$C_1 =$*fuwelßtmebailstler*
$C_2 =$*fuwßeltbamelitsler* [2 Punkte]

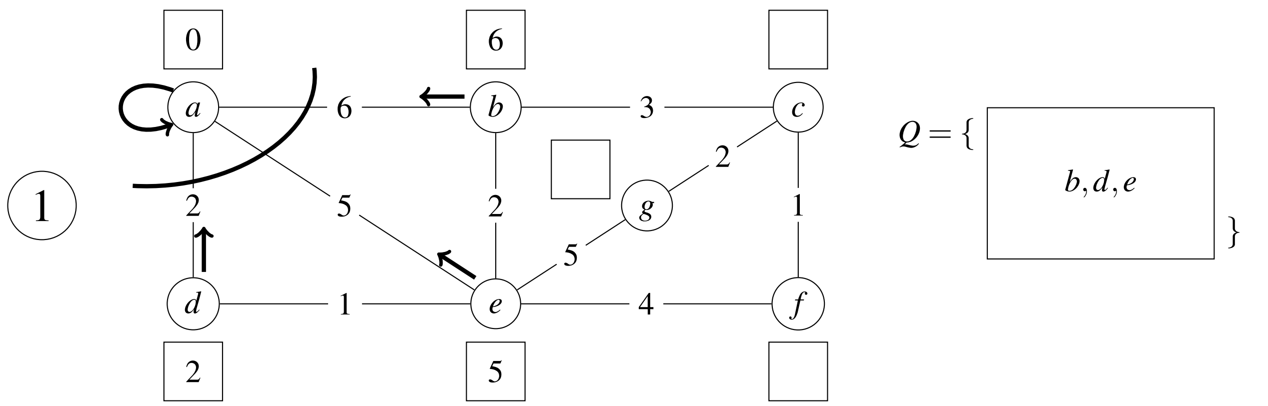
b. Entwerfen Sie einen Algorithmus, der für ein Wort C in Zeit $O(nm)$ bestimmt, ob es ein Shuffle der Wörter A und B ist. Wie in Teilaufgabe a) sei $|A| = n$, $|B| = m$ und $|C| = n+m$. [6 Punkte]

Aufgabe 6. Minimale Spannbäume [11 Punkte]

a. Nennen und erläutern Sie die Eigenschaft von minimalen Spannbäumen (MSTs) auf der der Algorithmus von Jarník-Prim basiert. [1 Punkt]

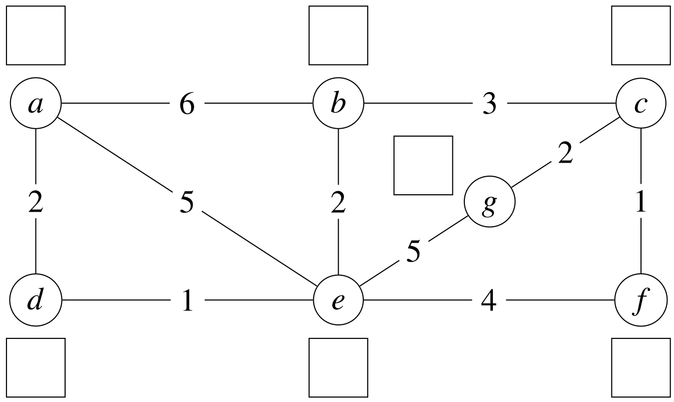
b. Führen Sie auf dem folgenden ungerichteten Graphen mit Kantengewichten den Algorithmus von Jarník-Prim mit Startknoten a durch. Verwenden Sie für jeden Schleifendurchlauf eine Kopie des Graphen, wobei die ersten zwei Iterationen bereits vorgegeben sind. Markieren Sie nach jeder Iteration den *aktuellen Schnitt* und tragen Sie für jeden Knoten v den Vorgänger $pred[v]$ als kurzen Pfeil und die aktuelle Distanz $d[v]$ in den Kasten ein, wobei Sie nur geänderte Werte eintragen müssen. Notieren Sie zwischen den Iterationen den Inhalt der Queue, und markieren sie nach Beenden den berechneten minimalen Spannbaum. Beschriften Sie deutlich die Graphen die gewertet werden sollen mit den Schleifendurchlaufsnummern! [5 Punkte]



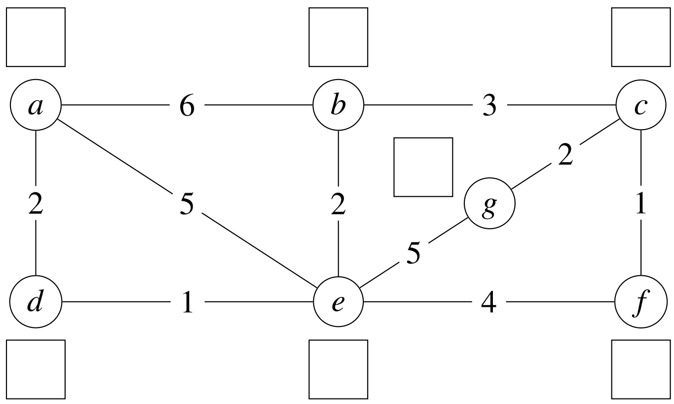


(weitere Graphen auf dem nächsten Blatt)

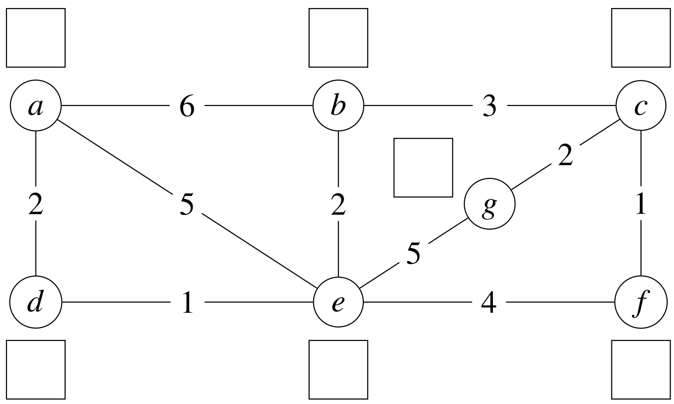
Fortsetzung von Aufgabe 6



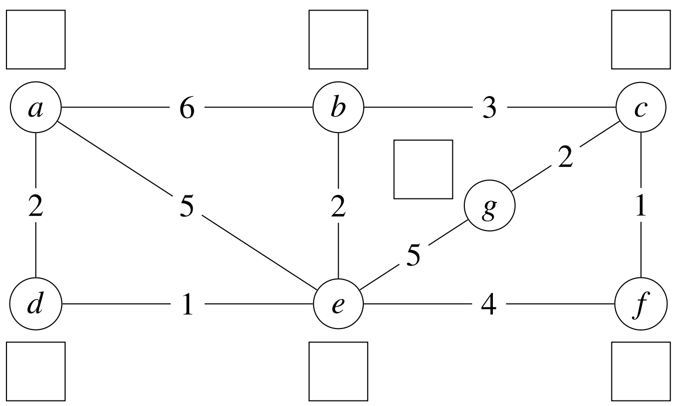
$Q = \{ \quad \}$



$Q = \{ \quad \}$



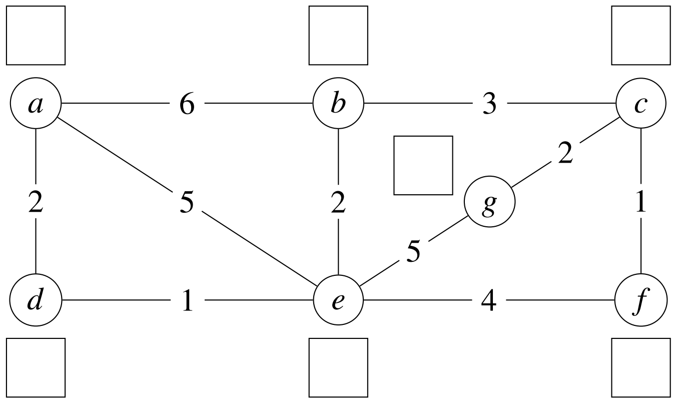
$Q = \{ \quad \}$



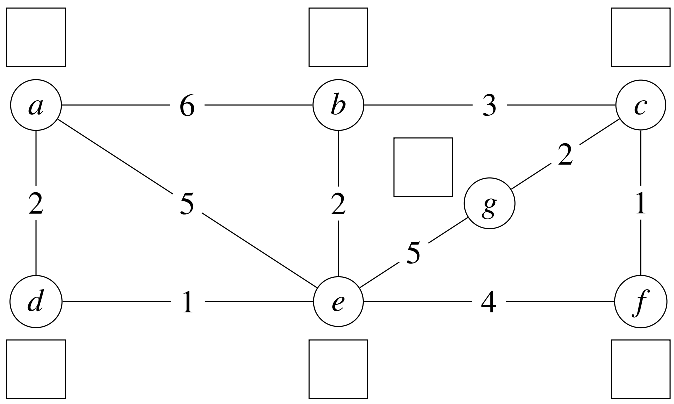
$Q = \{ \quad \}$

(weitere Graphen auf dem nächsten Blatt)

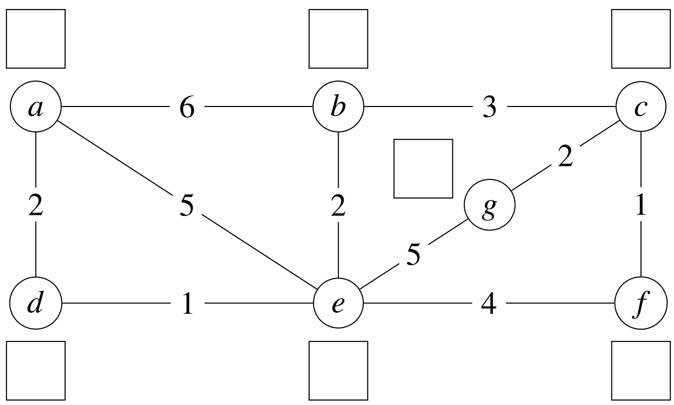
Fortsetzung von Aufgabe 6



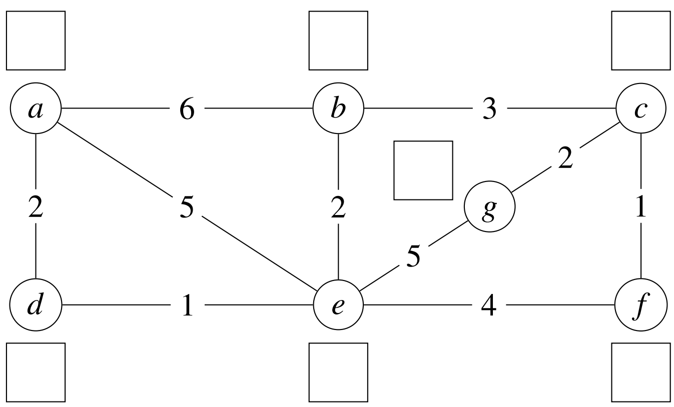
$Q = \{ \quad \}$



$Q = \{ \quad \}$



$Q = \{ \quad \}$



$Q = \{ \quad \}$

(Teilaufgabe **c.** auf dem nächsten Blatt.
Sie erhalten weitere Blätter mit Graphen bei Bedarf von der Aufsicht.)

Fortsetzung von Aufgabe 6

c. Professor Euler hat bereits viele Brücken untersucht und schlägt folgenden Algorithmus zur Berechnung eines minimalen Spannbaums vor:
In einem ungerichteten, gewichteten, zusammenhängenden Graphen untersucht man die Kanten nach Gewicht in nicht-steigender Reihenfolge, und entfernt jede Kante, die zu diesem Zeitpunkt keine Brücke ist. Eine Kante ist eine Brücke, wenn der Graph beim Entfernen der Kante in zwei Komponente verfällt.
Berechnet der Algorithmus von Professor Euler in jedem Graphen einen minimalen Spannbaum? Beweisen Sie dies, oder geben Sie ein Gegenbeispiel an. [5 Punkte]

Konzeptpapier (Abgabe freiwillig)