

Prof. Johannes Buchmann  
Fateme Shirazi  
Johannes Braun  
Dr. Albrecht Petzoldt

## WS 2014/15



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

[illegible]

---

## Allgemeines

---

- Halten Sie Ihren Studenausweis und einen Lichtbildausweis zur Kontrolle bereit.
- Füllen Sie das Deckblatt vollständig aus.
- Prüfen Sie, ob die Klausur 12 Aufgaben enthält.
- Verwenden Sie für jede Aufgabe ein neues Blatt.
- Leerblätter werden von der Aufsicht gestellt. Verwenden Sie kein eigenes Papier.
- Geben Sie die verwendeten Formeln, Sachverhalte und Zwischenergebnisse an.

---

## Bewertung

---

- Unleserlichkeit führt zu Punktabzug.
- Diese Klausur wird nur gewertet, falls die Semesterleistung ordnungsgemäß erbracht wurde.

---

## Dauer der Klausur und zugelassene Hilfsmittel

---

- Ihnen stehen 120 Minuten zum Bearbeiten der Aufgaben zur Verfügung.
- Einzige zugelassene Hilfsmittel sind ein nicht programmierbarer Taschenrechner ohne Formelspeicher und ein beidseitig handschriftlich beschriebenes DIN-A4 Blatt (keine Kopien etc.).
- Bitte geben Sie andere elektronische Geräte (Handys, PDAs, Laptops, programmierbare Taschenrechner) der Klausuraufsicht zur Verwahrung.
- Studierende, deren Muttersprache nicht Deutsch ist, können zusätzlich ein zweisprachiges Wörterbuch verwenden.
- Die Klausuraufsicht überprüft die Hilfsmittel.

a) Ergänzen Sie den folgenden Text.

Ein azyklischer Graph mit  $n$  Knoten hat höchstens \_\_\_\_\_ Kanten.

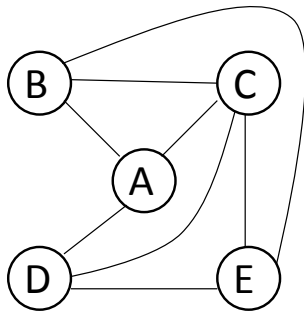
Ein bipartiter Graph mit  $n$  Knoten hat höchstens \_\_\_\_\_ Kanten.

Ein vollständiger trinärer Baum der Höhe  $h$  hat genau \_\_\_\_\_ Knoten.

b) Gegeben seien zwei Polynome  $A$  und  $B$  vom Grad  $n$ . Geben Sie die Laufzeit folgender Algorithmen in Abhängigkeit von  $n$  an (Groß O-Notation).

Polynomaddition $A + B$	
Evaluation von $A$ (mittels Hornerschema)	
Polynommultiplikation $A \cdot B$ (mittels FFT)	

c) Ist auf untenstehendem Graphen eine Eulertour möglich? Anfangs- und Endpunkt müssen dabei nicht notwendigerweise übereinstimmen. Begründung?



Betrachten Sie den folgenden Algorithmus Selection-Sort. Der Algorithmus erhält als Input ein Array  $A$  und überschreibt dieses mit dem sortierten Array.

Selection-Sort( $A$ )

```
1   $n = \text{length}(A)$ 
2  for  $j = 1$  to  $n - 1$  do
3     $\text{smallest} = j$ 
4    for  $i = j + 1$  to  $n$  do
5      if  $A[i] < A[\text{smallest}]$  then
6         $\text{smallest} = i$ 
7    exchange  $A[j]$  and  $A[\text{smallest}]$ 
```

Beweisen Sie die Korrektheit des Algorithmus Selection-Sort. Geben Sie dazu zunächst eine geeignete Schleifeninvariante für die äußere for-Schleife (Zeile 2 -7) an.

Bei Ihrer Argumentation können Sie der Einfachheit halber annehmen, dass alle Elemente des Arrays  $A$  verschieden sind.

a) Vervollständigen Sie die folgende Definition.

$$f(n) = \Omega(g(n)) \Leftrightarrow \text{_____ } c \in \mathbb{R}_+ \text{ _____ } n_0 \in \mathbb{N} \text{ so dass _____ } \forall n \geq n_0.$$

b) Vervollständigen Sie folgende Tabelle. Kreuzen Sie dabei an, ob  $f(n) = \omega(g(n))$ ,  $f(n) = \Theta(g(n))$  oder  $f(n) = o(g(n))$  gilt. In jeder Halbzeile ist nur eine Lösung richtig.

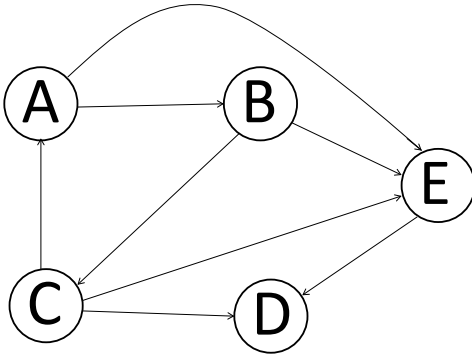
$f(n)$	$g(n)$	$\omega$	$\Theta$	$o$	$f(n)$	$g(n)$	$\omega$	$\Theta$	$o$
$3 \cdot \log(n^2)$	$0.5 \cdot (\log n)^2$				$2 \cdot n^2$	$e^{3 \log n}$			
$e^{n^2}$	$e^{5 \cdot n}$				$5 \cdot \log(3 \cdot n)$	$4 \cdot (\log n^2)$			
$2 \cdot e^{2 \cdot \log n}$	$3^{\sqrt{n}}$				$n^4$	$e^{2 \cdot \log \log n}$			

Illustrieren Sie die Operation von Partition (Subroutine von Quicksort) auf dem Array  $A[p \dots r] = \langle 17, 11, 3, 9, 22, 15 \rangle$ . Der Algorithmus durchläuft in einer for-Schleife nacheinander die Elemente des Arrays. Geben Sie den Zustand des Arrays  $A$  nach jedem Durchlauf der for-Schleife an. Als Pivotelement wird dabei  $A[r]$  gewählt. Der Zähler  $i$  wird immer dann erhöht, wenn das gerade betrachtete Element kleiner als das Pivotelement ist. Geben Sie den Wert von  $i$  nach jeder Iteration der for-Schleife an (inklusive des Wertes von  $i$  vor dem ersten Schleifendurchgang). Geben Sie darüber hinaus den Rückgabewert der Funktion an.

$\langle 17, 11, 3, 9, 22, 15 \rangle$	$i =$
$\langle \quad , \quad , \quad , \quad , \quad \rangle$	$i =$
$\langle \quad , \quad , \quad , \quad , \quad \rangle$	$i =$
$\langle \quad , \quad , \quad , \quad , \quad \rangle$	$i =$
$\langle \quad , \quad , \quad , \quad , \quad \rangle$	$i =$
$\langle \quad , \quad , \quad , \quad , \quad \rangle$	$i =$
$\langle \quad , \quad , \quad , \quad , \quad \rangle$	$i =$

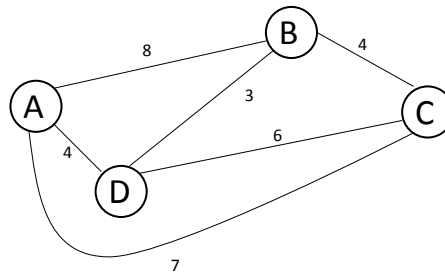
Rückgabewert:

Führen Sie auf untenstehendem Graphen eine Tiefensuche durch. Benutzen Sie dabei Knoten A als Startknoten. Geben Sie für jeden Knoten discovery und finishing time an. Führen Sie darüber hinaus eine Kategorisierung der Kanten durch, indem Sie zu jeder Kante den jeweiligen Kantentyp (Baumkante (T), Vorwärtskante (F), Rückkante (B), Querkante (Q)) angeben.



Finden Sie für den folgenden Graphen einen minimalen aufspannenden Baum. Benutzen Sie den Algorithmus von Prim. Benutzen Sie den Knoten  $A$  als Startknoten. Der Algorithmus durchläuft in einer Schleife alle Knoten des Graphen. Tragen Sie nach jedem Schleifendurchlauf die Werte von  $key[v]$  (Abstand von  $v$  zur bisherigen Zusammenhangskomponente),  $\pi[v]$  (Vorgänger) (für alle Knoten  $v$ ),  $Q$  (noch nicht abschließend betrachtete Knoten) und  $u$  (gerade abgeschlossener Knoten) in die folgende Tabelle ein. Jede Tabellenzeile entspricht dabei einem Iterationsschritt.

Geben Sie den minimalen aufspannenden Baum an, indem Sie die entsprechenden Kanten im untenstehenden Graphen markieren.



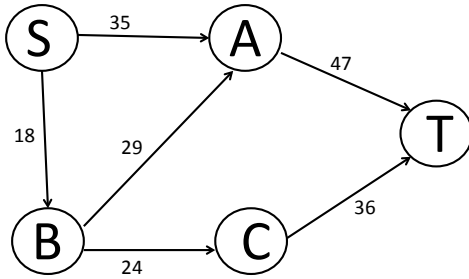
$key[A]$	$key[B]$	$key[C]$	$key[D]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	$Q$	$u$
0	$\infty$	$\infty$	$\infty$	$nil$	$nil$	$nil$	$nil$	$\{A, B, C, D, E\}$	—



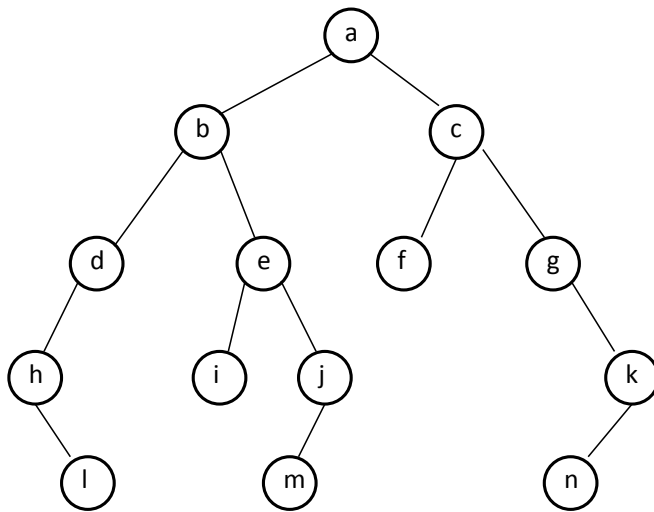
Bestimmen Sie mit dem Algorithmus von Ford-Fulkerson den maximalen Fluss in untenstehendem Flussnetzwerk mit Quelle S und Senke T. Der Algorithmus fügt in jedem Schritt einen ergänzenden Pfad zum aktuellen Flussnetzwerk hinzu. Dies wird so lange wiederholt, bis es keinen ergänzenden Pfad mehr gibt.

Geben Sie nach jedem Schritt den gerade hinzugefügten ergänzenden Pfad, seine Kapazität und den aktuellen Gesamtfluss an. Skizzieren Sie außerdem das entstehende Restnetzwerk. Fügen Sie dabei stets den ergänzenden Pfad mit maximaler Kapazität hinzu.

Beweisen Sie die Maximalität des gefundenen Flusses, indem Sie einen minimalen Cut in das in der Aufgabenstellung angegebene Netzwerk einzeichnen.



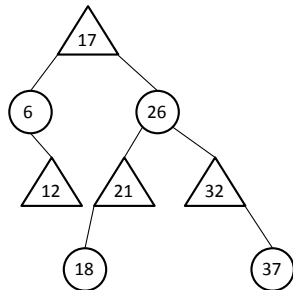
Betrachten Sie den folgenden binären Baum. Die im Baum enthaltenen Schlüssel  $a, \dots, n$  erfüllen die binäre Suchbaumeigenschaft.



- Geben Sie den kleinsten der im Baum enthaltenen Schlüssel an.
- Geben Sie den Nachfolger von  $b$  bzw.  $l$  an.
- Geben Sie den Vorgänger von  $n$  bzw.  $a$  an.
- Geben Sie den fünftgrößten der im Baum enthaltenen Schlüssel an.

Führen Sie auf untenstehendem RB-Baum die Operationen Rb-Delete(26) und RB-Delete(37) in dieser Reihenfolge durch. Skizzieren Sie den RB-Baum nach jedem Löschvorgang und geben Sie jeweils an, ob und welche Fälle der Methode RB-Delete-Fixup aufgerufen werden.

Stellen Sie dabei rote und schwarze Knoten wie in der Abbildung gezeigt dar.



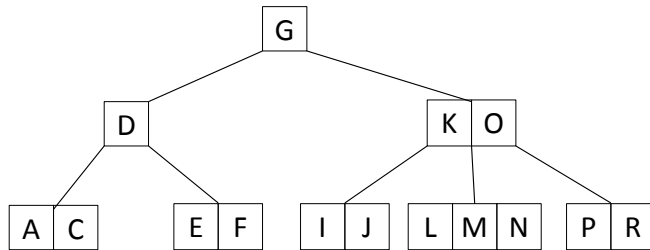
schwarzer Knoten



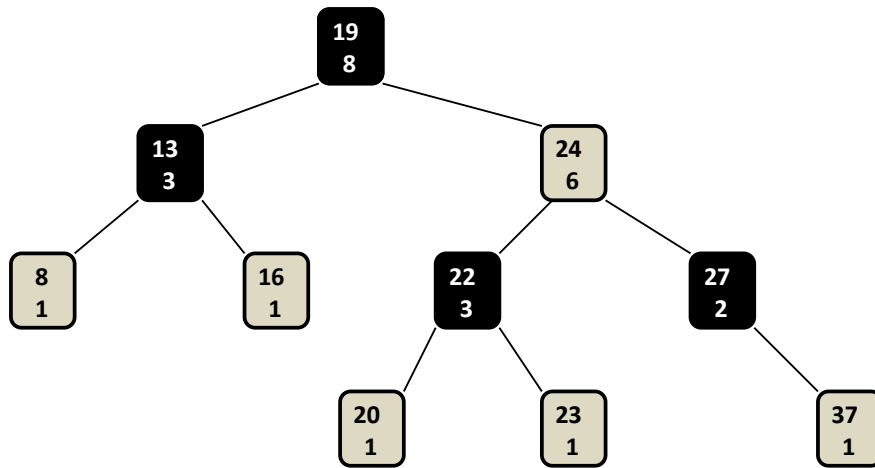
roter Knoten

Gegeben sei der untenstehende B-Baum vom Minimalgrad 2. Löschen Sie die Schlüssel E und O in dieser Reihenfolge aus dem Baum.

Skizzieren Sie den entstehenden B-Baum nach jedem Löschvorgang und geben Sie an, welche Fälle beim Löschen auftreten.



Gegeben sei der untenstehende OS-Baum  $T$  mit 10 Knoten. Bestimmen Sie den Rang des Schlüssels 22 sowie den zweitgrößten der im Baum enthaltenen Schlüssel mit Hilfe der Funktionen OS-Rank und OS-Select. Geben Sie an, welcher Wert der Funktion OS-Select übergeben wird und füllen Sie die untenstehenden Tabellen aus. Geben Sie ferner die Rückgabewerte der Funktionen an.



OS-Rank( $T, 22$ )

Iteration	$y$	$r$

Rückgabewert:

OS-Select( $\text{root}(T), \underline{\hspace{2cm}}$ )

Iteration	$x$	$i$	$r$

Rückgabewert:

Berechnen Sie die diskrete Fouriertransformation des Vektors  $a = (1, 2, 1, 3)$ .