



# Informatik II

## Hauptklausur SS 07

Vorname:

Name:

Matrikelnummer:

Zur Klausur sind keine Hilfsmittel zugelassen. Die Bearbeitungszeit beträgt 60 Minuten. Bitte tragen Sie Ihren Namen und Ihre Matrikelnummer auf dieser und allen folgenden Seiten ein.

Schreiben Sie Ihre Lösungen in den jeweils für die entsprechende Aufgabe vorgesehenen Kasten. Die Größe des Kastens steht nicht zwangsweise in Zusammenhang mit dem Umfang der für volle Punktzahl erforderlichen Antwort. Sollte der Platz im Kasten nicht ausreichen, können Sie die Rückseite des jeweiligen Blattes verwenden. Vermerken Sie im Kasten, wo der Rest Ihrer Antwort zu finden ist.

Für alle Programmieraufgaben gilt, dass Sie Variablenamen, Kommentare, etc. wahlweise in deutscher oder in englischer Sprache angeben dürfen. Ansonsten gelten die aus der Vorlesung bekannten Programmierrichtlinien.

Die Klausur ist komplett und geheftet abzugeben. Sie dürfen die Rückseiten der Aufgabenblätter als Konzeptpapier benutzen. Verwenden Sie kein eigenes Papier. Sollten Sie zusätzliches Papier benötigen, so fordern Sie dieses bei der Klausuraufsicht an.

Die unten stehende Tabelle sowie das Summenfeld unten auf jeder Seite werden von uns bei der Korrektur ausgefüllt.

Aufgabe	1	2	3	4	5	6	7	Summe
Maximal	7	9	9	10	9	9	7	60
Erreicht								

Note:

## 1 Verständnis- und Wissensfragen (7 Punkte)

a) Kreuzen Sie jeweils an, ob die Aussage wahr oder falsch ist. (4 Punkte)

**Hinweis:** Korrekte Antworten werden mit 0,5 Punkten, falsche Antworten mit einem Abzug von 0,5 Punkten und nicht beantwortete Fragen mit 0 Punkten bewertet. Sollte daraus in der Summe ein negatives Ergebnis für die Teilaufgabe a) resultieren, wird die Teilaufgabe mit 0 Punkten bewertet.

Aussage	wahr	falsch
A sei ein Supertyp von B. Y sei ein Subtyp von X. Ferner seien die Methoden $X.foo(B)$ und $Y.foo(B)$ definiert. Die angegebenen Typenbeziehungen und Methoden verletzen die Kontravarianzregel.		X
Bei einem einfachen Zweigüberdeckungstest kann nie eine Kante des Kontrollflussgraphen doppelt durchlaufen werden.		X
Die Klasse <code>java.util.List</code> implementiert das Interface <code>java.util.Collection</code> .		X
Ein Las Vegas Algorithmus liefert immer das korrekte Ergebnis.	X	
Das Löschen in einer randomisierten Skip-Liste geschieht durch einen Zufallsalgorithmus.		X
Das Auswahlproblem ist nicht schneller als $O(n \log n)$ lösbar.		X
In einem binären Suchbaum, der die Rot-Schwarz-Eigenschaften erfüllt (Rot/Schwarz-Baum), gilt: Der Vater eines roten Knotens ist stets schwarz.	X	
Die Entwurfsmethode <i>Dynamische Programmierung</i> ist nur dann effizient, wenn die Teilprobleme disjunkt sind.		X

b) Wie lautet die Definition für die obere asymptotische Schranke? (1 Punkt)

$$O(g(n)) = \{f(n) \mid \exists c > 0 \exists n_0 > 0 \forall n > n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$

c) Bezeichne  $k$  einen Knoten, der in einen Rot/Schwarz-Baum eingefügt wurde. Bezeichne  $parent(k)$  den Vater von  $k$ . Sei  $parent(k)$  ein linkes Kind von  $parent(parent(k))$ . Ein Teil der Einfügeoperation ist die Wiederherstellung der Eigenschaften des Rot/Schwarz-Baumes. Welche drei Fälle sind dabei zu unterscheiden? (2 Punkte)

- (E1) Der Onkel von  $k$  ist rot.
- (E2) Der Onkel von  $k$  ist schwarz, und  $k$  ist rechtes Kind.
- (E3) Der Onkel von  $k$  ist schwarz, und  $k$  ist linkes Kind.



## 2 Modellierung mit UML (9 Punkte)

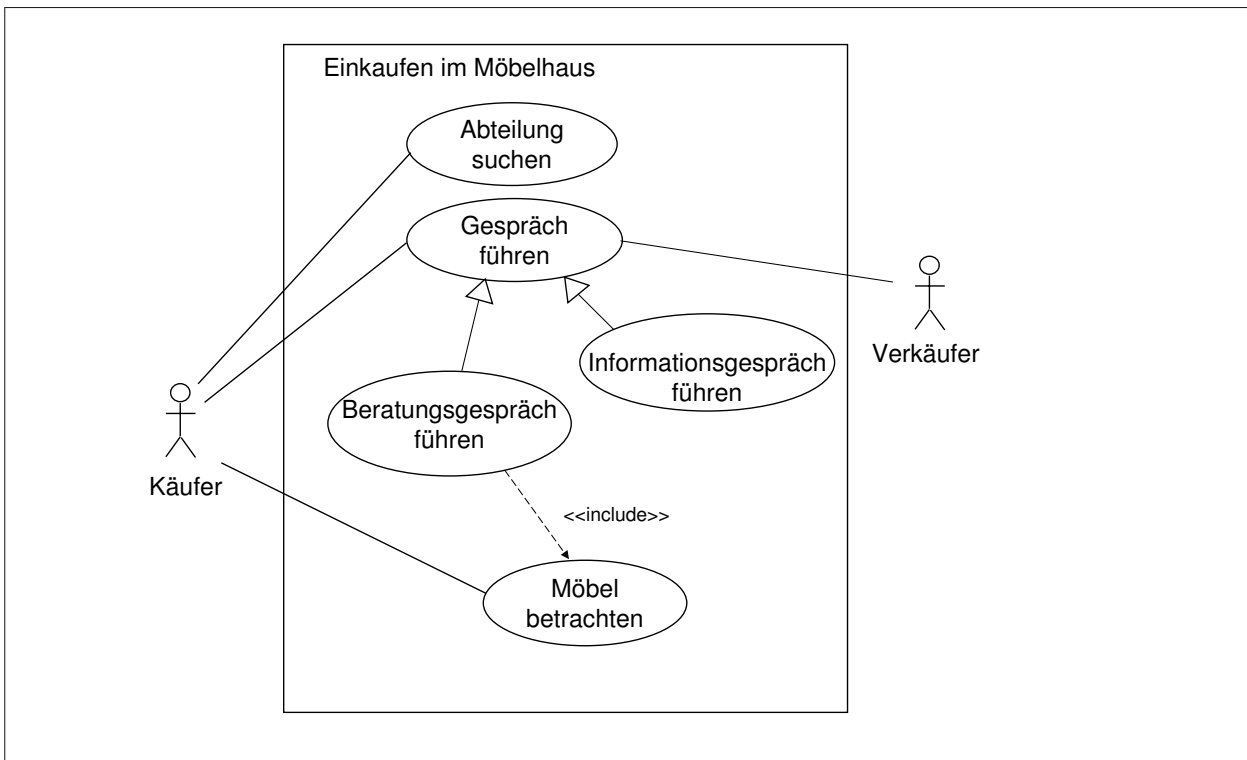
Setzen Sie folgende Beschreibungen für das Szenario Möbelkauf in UML um. Benutzen Sie dafür die jeweils angegebene Diagrammart.

**Hinweis:** Lesen Sie die jeweilige Beschreibung vollständig durch, bevor Sie mit der Modellierung beginnen.

a) Modellieren Sie die folgenden Zusammenhänge als **Anwendungsfalldiagramm**.

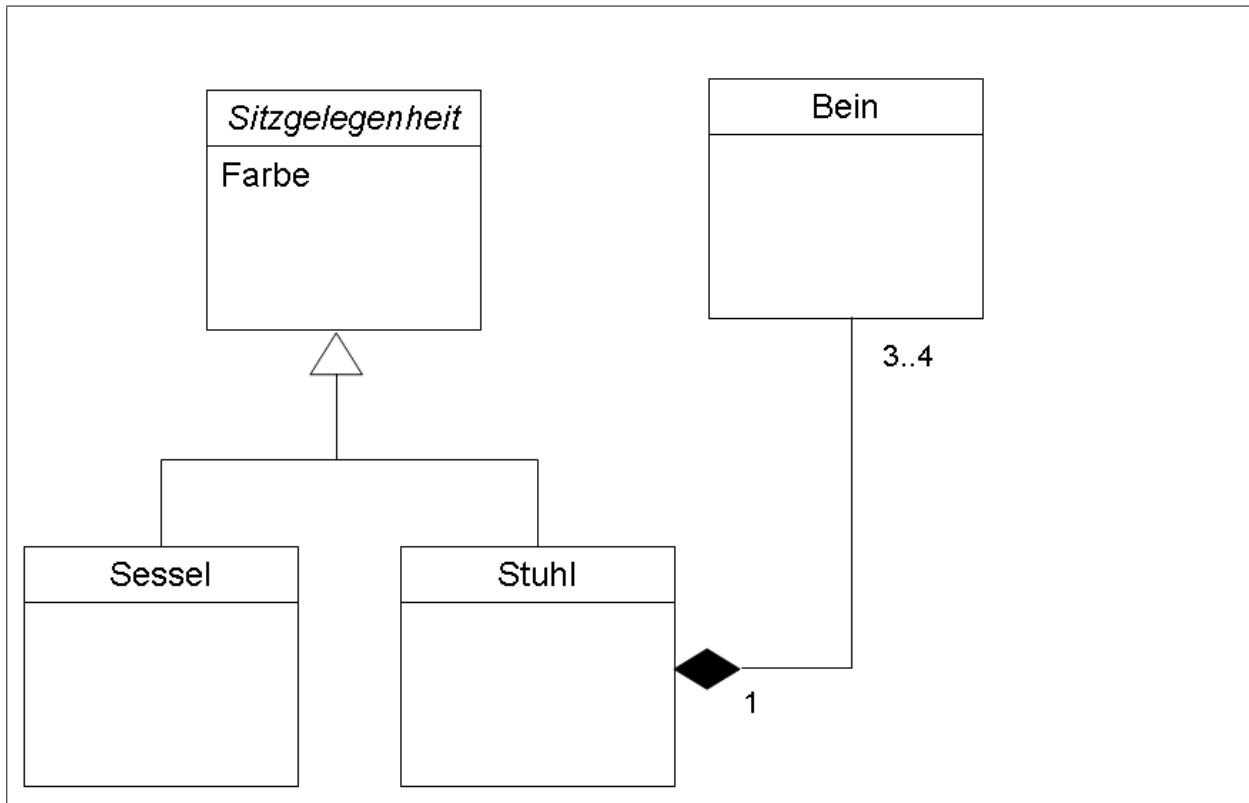
Ein Einkauf im Möbelhaus lässt sich folgendermaßen beschreiben:

Ein Kunde sucht die richtige Möbelabteilung. Er betrachtet Möbel. Außerdem führt er Gespräche mit Verkäufern. Gespräche können in Beratungsgespräche und kurze Informationsgespräche unterteilt werden. Beratungsgespräche enthalten ebenfalls das Betrachten von Möbeln. (3 Punkte)



- b) Modellieren Sie die folgenden Zusammenhänge als **Klassendiagramm**. Verwenden Sie an passender Stelle abstrakte Klassen und ziehen Sie Attribute so weit wie möglich in die Oberklassen.

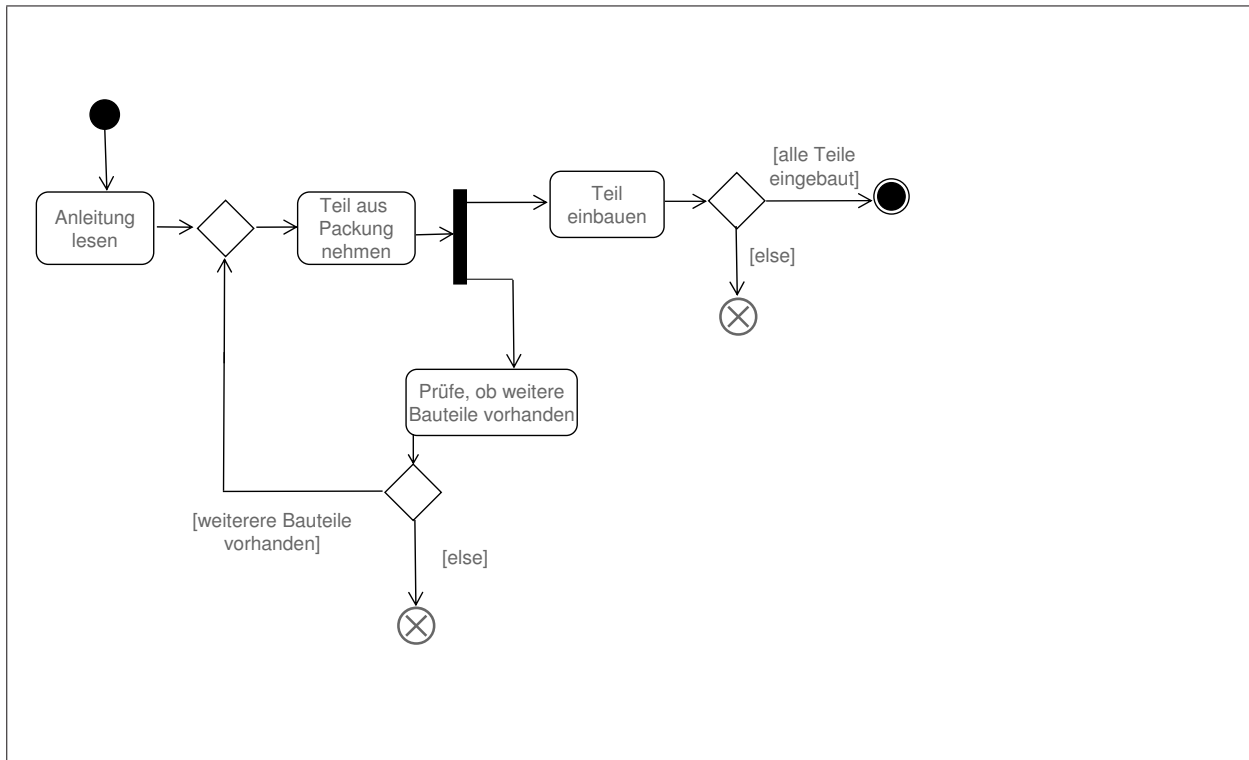
Ein Stuhl besteht aus 3 bis 4 Beinen. Jedes Bein gehört zu genau einem Stuhl. Ein Bein kann nicht ohne Stuhl existieren. Stühle haben eine Farbe und sind eine Sitzgelegenheit. Sessel haben keine Beine aber eine Farbe und sind ebenso eine Sitzgelegenheit. Außer Sesseln und Stühlen existieren keine weiteren Sitzgelegenheiten. (3 Punkte)



c) Modellieren Sie die folgenden Zusammenhänge als **Aktivitätsdiagramm**.

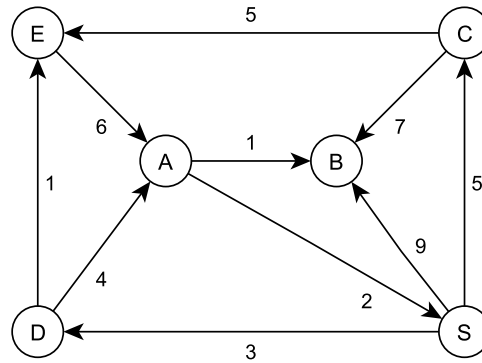
Die Aktivität „Zusammensetzen“ kann wie folgt beschrieben werden: Zunächst wird die Aufbauanleitung gelesen. Dann wird ein Teil aus der Packung entnommen. Anschließend wird das Teil eingebaut. Falls alle Teile eingebaut sind, wird die Aktivität beendet. Parallel zum Einbauen wird geprüft, ob weitere Bauteile vorhanden sind. Falls ja, wird ein weiteres Teil aus der Packung entnommen und anschließend eingebaut.

**Hinweis:** Es brauchen *keine* Objektknoten modelliert zu werden. (3 Punkte)



### 3 Dijkstra-Algorithmus (9 Punkte)

Gegeben sei folgender Graph:



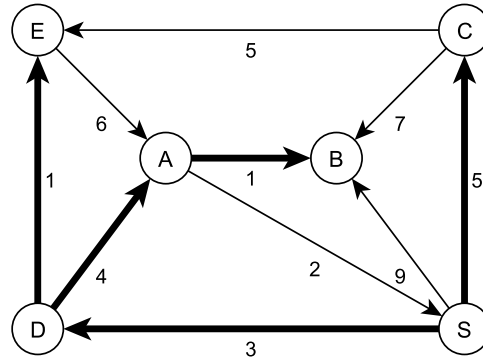
- a) Verwenden Sie den Dijkstra-Algorithmus um in dem Graphen die kürzesten Wege von Knoten  $S$  zu allen anderen Knoten zu finden. Verwenden Sie dafür folgende Tabelle! Geben Sie nach jeder Iteration des Algorithmus hier neben dem aktuell besuchten Knoten jeweils alle momentan kürzesten Wege von  $S$  zu allen Knoten an. (3 Punkte)

**Hinweis:** Wir empfehlen die parallele Bearbeitung von Teilaufgabe b.

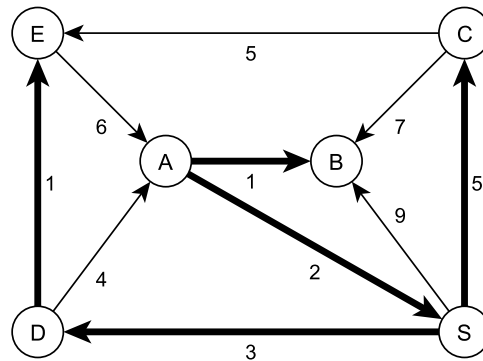
Iteration	Akt. Knoten	$S$	$A$	$B$	$C$	$D$	$E$
0	-	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	$S$	0	$\infty$	9	5	3	$\infty$
2	$D$	0	7	9	5	3	4
3	$E$	0	7	9	5	3	4
4	$C$	0	7	9	5	3	4
5	$A$	0	7	8	5	3	4
6	$B$	0	7	8	5	3	4



- b) Zeichnen Sie in der folgenden Darstellung den durch den Dijkstra-Algorithmus implizit erzeugten Spannbaum mit kürzesten Wegen ein! (2 Punkte)



- c) Handelt es sich bei dem durch den Dijkstra-Algorithmus erzeugten Spannbaum aus Aufgabenteil b) auch um einen minimalen Spannbaum? Falls nicht, zeichnen Sie in der folgenden Abbildung einen Spannbaum für den Graphen aus dem ersten Aufgabenteil ein, der geeignet ist zu zeigen, dass der durch Dijkstra gefundene Spannbaum nicht minimal ist. (3 Punkte)



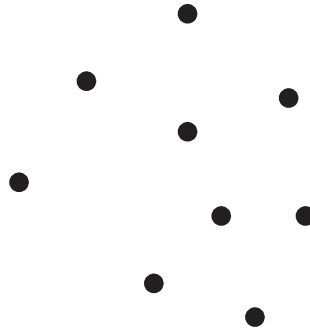
- d) Warum wird durch den Dijkstra-Algorithmus im Allgemeinen kein minimaler Spannbaum gefunden? (1 Punkt)

Da hier ein Wurzelknoten vorgegeben wird. Wie im Beispiel ersichtlich ist der vorgegebene Wurzelknoten im allgemeinen nicht der Wurzelknoten eines minimalen Spannbaums.



#### 4 Konvexe Hülle (10 Punkte)

- a) Zeichnen Sie die konvexe Hülle für die folgende Menge von Punkten ein. (1 Punkt)

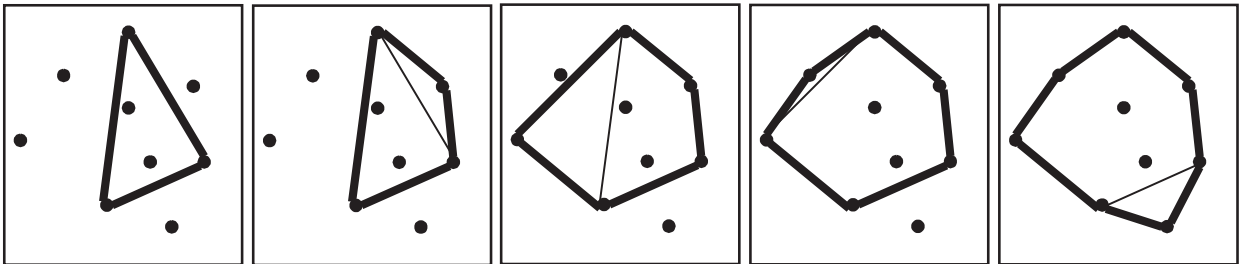


- b) Geben Sie eine kurze Definition der konvexen Hülle wie Sie sie in der Vorlesung **oder** der Übung kennen gelernt haben. (2 Punkte)

*Vorlesung:* Die konvexe Hülle einer Punktmenge ist das kleinste konvexe Polygon, das alle Punkte in seinem Inneren hat.

*Übung:* Sei  $M \subset \mathbb{R}^2$ . Die konvexe Hülle der Menge  $M$  ist die kleinste konvexe Menge ( $\subset \mathbb{R}^2$ ), die  $M$  enthält.

- c) Im Folgenden sehen Sie die Berechnung der konvexen Hülle an einem Beispiel dargestellt. Welcher Algorithmus wird hier zur Berechnung eingesetzt (Name)? (1 Punkt)



Ausdehnungsalgorithmus (stretching algorithm)



- d) Welchen Aufwand hat dieser Algorithmus für  $n$  Punkte? Bitte begründen Sie Ihre Antwort kurz. (3 Punkte)

Für alle Punkte müssen alle Winkel zu anderen Punkten berechnet werden, also  $O(n^2)$ .

- e) Im Folgenden sehen Sie Pseudocode für die Berechnung der konvexen Hülle nach Graham (Graham's Scan). Bitte vervollständigen Sie diesen. (3 Punkte)

beginne mit einem äußeren Punkt  $P_1$  (z.B. äußerst rechts unten)  
berechne die Winkel der Verbindungen aller Punkte zu  $P_1$  mit der x-Achse  
sortiere alle Punkte entsprechend dieser Winkel in eine Liste  
**für alle** Punkte  $P_i$  in der Liste  
    **falls**  $P_i$  von  $P_{i-1}$  über einen Innenwinkel  $\leq 180^\circ$  erreichbar  
        füge neue Kante von  $P_{i-1}$  zu  $P_i$  hinzu  
    **sonst**  
        entferne  $P_{i-1}$  aus der Hülle; ebenso alle  $P_{i-k}$   
        von denen  $P_i$  über Innenwinkel  $> 180^\circ$  erreichbar



## 5 Schleifeninvariante (9 Punkte)

Seien  $a_k = a_0 \cdot x^k$  die Glieder einer geometrischen Folge, wobei  $a_0$  das Anfangsglied und  $x$  das Verhältnis zwischen zwei benachbarten Gliedern ist. Das  $n$ -te Glied einer geometrischen Reihe lässt sich wie folgt berechnen:

$$s_n = \sum_{k=0}^n a_0 \cdot x^k = a_0 \frac{x^{n+1} - 1}{x - 1}, \quad x \neq 1$$

Das folgende Fragment enthält an Java angelehnten Pseudocode, der algorithmisch das  $n$ -te Glied einer geometrischen Reihe berechnet.

```

1      // Q: n >= 0
2      int i = 0;
3      double s = a;
4      // P: ?
5      while (i < n) {
6          i = i + 1;
7          s = s + a * xi;
8      }
9      //R: s = ∑k=0n a · xk

```

Vor der Ausführung gelte die Vorbedingung Q:  $n \geq 0$ . Nach der Ausführung gelte die Nachbedingung  $R: s = \sum_{k=0}^n a \cdot x^k$

**Hinweis:** Gehen Sie von einem idealen Rechnermodell mit unendlichen Ganzzahlen und beliebig genauer Gleitkommarithmetik aus!

a) Begründen Sie, warum die Schleife terminiert! (1 Punkt)

Terminierungsfunktion:  $T = x - i$  Da  $i$  in jedem Schleifendurchlauf inkrementiert wird, ist  $T$  monoton fallend. Durch das Abbruchkriterium gilt  $T \geq 0$ .

b) Geben Sie für die **while**-Schleife eine Schleifeninvariante  $P$  an, die geeignet ist, die Nachbedingung  $R$  herzuleiten. (2 Punkte)

Invariante  $P: \left( s = \sum_{k=0}^i a \cdot x^k \right) \wedge (i < n + 1)$



c) Zeigen Sie, dass die Schleifeninvariante  $P$  unmittelbar vor der `while`-Schleife gilt. (1 Punkt)

Aus Vorbedingung  $Q : n \geq 0$  und Initialisierung: "s=a; i=0;"  $\Rightarrow (s = \sum_{k=0}^0 (a \cdot x^k) = a) \wedge (i < n + 1) \Rightarrow$   
Invariante gilt vor der Schleife!

d) Zeigen Sie:  $P \wedge B \Rightarrow wp(A, P)$ , wobei  $A$  der Schleifenrumpf und  $B$  die Bedingung der `while`-Schleife ist. (4 Punkte)

$\{P \wedge B\} A \{P\}$   
 $wp("i=i+1; s=s+a \cdot x^i;", P)$   
 $= wp("i=i+1;", wp("s=s+a \cdot x^i;", s = \sum_{k=0}^i a \cdot x^k \wedge i < n + 1))$   
 $= wp("i=i+1;", s + a \cdot x^i = \sum_{k=0}^i a \cdot x^k \wedge i < n + 1)$   
 $\iff (s + a \cdot x^{i+1} = \sum_{k=0}^{i+1} a \cdot x^k \wedge i + 1 < n + 1)$   
 $\iff (s + a \cdot x^{i+1} = \sum_{k=0}^i (a \cdot x^k) + a \cdot x^{i+1} \wedge i + 1 < n + 1)$   
 $\iff (s + a \cdot x^{i+1} = \sum_{k=0}^i (a \cdot x^k) + a \cdot x^{i+1} \wedge i + 1 < n + 1)$   
 $\iff (s = \sum_{k=0}^i (a \cdot x^k) \wedge i < n) \iff P \wedge B = (s = \sum_{k=0}^i a \cdot x^k \wedge i < n + 1) \wedge (i < n)$



- e) Folgern Sie aus P und der negierten Schleifenbedingung, dass nach dem letztmaligen Durchlaufen der Schleife die Nachbedingung R gilt.

Zeigen sie dazu:  $P \wedge \neg B \Rightarrow R$  (1 Punkt)

Zu zeigen:  $P \wedge \neg B \Rightarrow R$

$P \wedge \neg B$

$$\Leftrightarrow \left( s = \sum_{k=0}^i a \cdot x^k \right) \wedge (i < n + 1) \wedge \neg(i < n)$$

$$\Leftrightarrow \left( s = \sum_{k=0}^i a \cdot x^k \right) \wedge \underbrace{(i < n + 1) \wedge (i \geq n)}_{\Rightarrow i=n}$$

$$\Rightarrow s = \sum_{k=0}^n a \cdot x^k = R$$



## 6 Minimale Höhe eines R-Baumes (9 Punkte)

Ein R-Baum habe pro Knoten minimal  $a$  und maximal  $2a$  MBRs (Minimum Bounding Rectangles). Folglich hat er minimal  $a$  und maximal  $2a$  Kindknoten. *Ausnahme:* Die minimale Anzahl der Kindknoten bzw. MBRs der Wurzel ist 2. Sei  $a > 0$  und  $n \geq 2a$ . Beweisen Sie, dass für die Baumhöhe  $h$  die Ungleichung gilt

$$\log_{2a} \left( n - \frac{n}{2a} + 1 \right) - 1 \leq h$$

falls insgesamt  $n$  MBRs im Baum vorhanden sind. (9 Punkte)

**Hinweis:** Die Höhe eines Baumes ist 0, wenn er nur aus der Wurzel besteht.

Ein R-Baum hat minimale Höhe, wenn alle Knoten maximal besetzt sind. D.h. alle Knoten haben  $2a$  MBRs und  $2a$  Kinder.

Es folgt:

$$n \leq 2a \cdot \sum_{i=0}^h (2a)^i$$

Anwendung der Geometrischen Reihe:

$$\begin{aligned} n &\leq 2a \cdot \left( \frac{(2a)^{h+1} - 1}{2a - 1} \right) \\ \frac{n}{2a} &\leq \frac{(2a)^{h+1} - 1}{2a - 1} \\ \frac{(2a - 1) \cdot n}{2a} &\leq (2a)^{h+1} - 1 \\ \frac{(2a - 1) \cdot n}{2a} + 1 &\leq (2a)^{h+1} \\ \log_{2a} \left( \frac{(2a - 1) \cdot n}{2a} + 1 \right) &\leq h + 1 \\ \log_{2a} \left( \frac{(2a - 1) \cdot n}{2a} + 1 \right) - 1 &\leq h \\ \log_{2a} \left( n - \frac{n}{2a} + 1 \right) - 1 &\leq h \end{aligned}$$

(1)



## 7 Entwurf eines Algorithmus in Pseudocode (7 Punkte)

Gegeben seien zwei Reihungen  $A = (a_1, a_2, \dots, a_n)$  und  $B = (b_1, b_2, \dots, b_n)$ . Weiterhin sei eine Zahl  $m$  gegeben.

- a) Beschreiben Sie einen Algorithmus in Pseudocode, der in  $O(n \log n)$  Schritten WAHR zurückgibt, falls ein Element  $a$  aus  $A$  und ein Element  $b$  aus  $B$  existiert, sodass gilt  $a + b = m$  und FALSCH sonst. (6 Punkte)

```

1 ALGORITHMUS(A, B, m)
2   B := Mergesort(B)
3   for i:=1 to länge[A]
4     do d:= m - A[i]
5         found := Binäre-Suche(B,d) {liefert WAHR, wenn B d enthält}
6         if ( found = WAHR)
7             return WAHR
8   return FALSCH

```

Alternativen:

- Statt Sortierung Aufbau eines balancierten binären Suchbaumes oder Heaps.
- Sortierung A aufsteigend, B absteigend. Gleichzeitiges Durchlaufen von A und B (Zähler für A bzw. B erhöhen, je nachdem ob Summe der aktuellen Elemente kleiner bzw. größer gleich m ist.)
- Weitere Alternativen sind möglich.

- b) Begründen Sie kurz, dass Ihr Algorithmus tatsächlich die geforderte asymptotische Laufzeit aufweist. (1 Punkt)

Sortierung (Zeile 2) benötigt  $O(n \log n)$ .  
 Schleife (Zeile 3) benötigt n Durchläufe. In jedem Durchlauf binäre Suche mit  $O(\log n)$ . Gesamtlaufzeit der Schleife daher  $O(n \log n)$ .  
 Algorithmus benötigt  $O(n \log n)$ .

