Hinweise zur Bearbeitung der Klausur zum Kurs 1663 „Datenstrukturen“

Bitte lesen Sie sich diese Hinweise vollständig und aufmerksam durch, bevor Sie mit der Bearbeitung der Klausur beginnen. Beachten Sie insbesondere die Punkte 7 und 8!

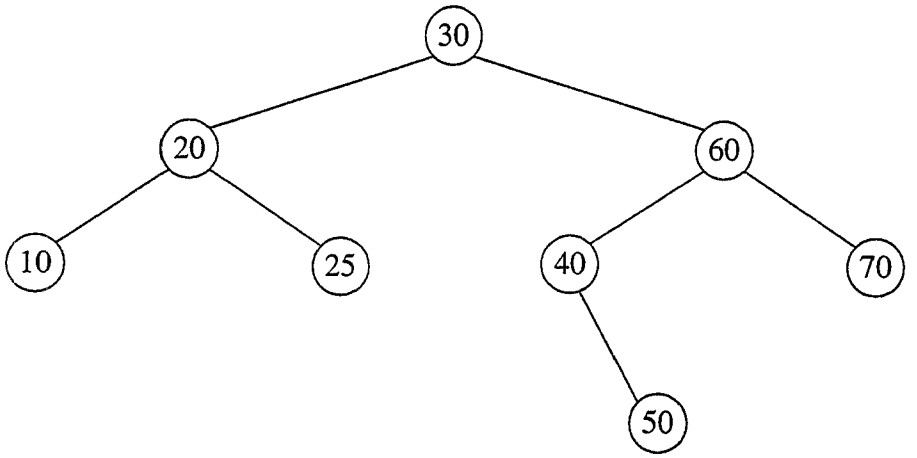
1. Die Klausurdauer beträgt 3 Stunden.
2. Prüfen Sie bitte die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst:
   - 2 Deckblätter
   - diese Hinweise
   - 1 Formblatt für eine Teilnahmebescheinigung zur Vorlage beim Finanzamt
   - 6 Aufgaben auf den Seiten 2 – 5
3. Bevor Sie mit der Bearbeitung der Klausuraufgaben beginnen, füllen Sie bitte die folgenden Teile der Klausur aus:
   (a) *sämtliche* Deckblätter mit Name, Anschrift sowie Matrikelnummer. Markieren Sie vor der Abgabe auf allen Deckblättern die von Ihnen bearbeiteten Aufgaben.
   (b) die Teilnahmebescheinigung, falls Sie diese wünschen.
4. Schreiben Sie Ihre Lösungen auf ihr eigenes Papier (DIN A4) und nicht auf die Seiten mit den Aufgabenstellungen. Heften Sie vor Abgabe der Klausur die Deckblätter (und evtl. die Teilnahmebescheinigung) an Ihre Bearbeitung.
5. Schreiben Sie bitte auf jedes Blatt oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Nummerieren Sie Ihre Seiten bitte durch.
6. Wenden Sie bei der Lösung der Aufgaben, soweit dies möglich ist, die Algorithmen und Notationen aus den Kurseinheiten an.
7. Schreiben Sie, wenn ***Algorithmen*** gefordert sind, keine kompletten PASCAL- oder MODULA-2-Programme, sondern beschränken Sie sich auf die wesentlichen Teile des Algorithmus, die z.T. auch in Prosa formuliert werden können. Formulieren Sie Algorithmen aber so, daß die elementaren Einzelschritte erkennbar werden.

   Sparen Sie aber bei solchen Aufgaben nicht mit Kommentaren. Wenn Ihre Lösung aufgrund fehlender Kommentare nicht verständlich ist, führt das zu Punktabzug.
8. Vermeiden Sie in jedem Fall bei der Definition von Funktionen in ***Algebren*** PASCAL- oder MODULA-2-Programme sowie Algorithmen. Geben Sie lediglich mathematische Definitionen an wie sie im Kurstext verwandt worden sind!

   Ebenso sind Mengendefinitionen in mathematischer Mengennotation durchzuführen. Geben Sie auf keinen Fall Datentypdefinitionen an, und verwenden Sie keine konkreten Datenstrukturen!
9. Als Hilfsmittel sind nur unbeschriebenes Konzeptpapier und Schreibzeug (kein Bleistift!) zugelassen.
10. Es sind maximal 100 Punkte erreichbar. Zur Erlangung eines Übungsscheins bzw. Zertifikats benötigen Sie etwa 50 Punkte.

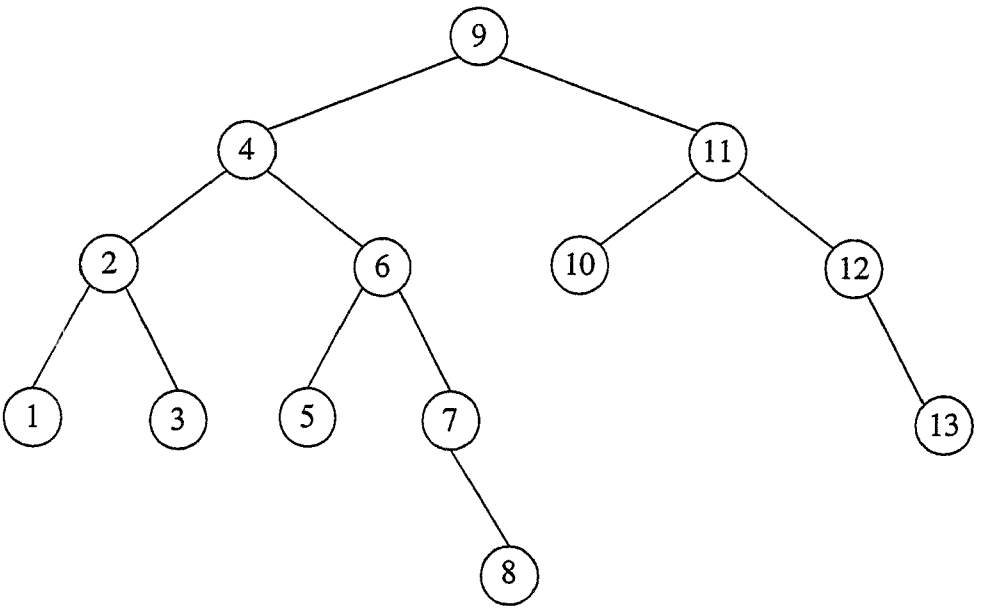
20 Punkte **Aufgabe 1 (AVL-Baum)**

8 Punkte (a) Gegeben sei der folgende AVL-Baum:



Fügen Sie nacheinander die Objekte 55, 58 und 45 in den Baum ein. Falls das AVL-Kriterium bei einer Einfügeoperation verletzt wird, markieren Sie den betroffenen Knoten und rebalancieren Sie den Baum unter Angabe des Verfahrens. Zeichnen Sie nach jedem Einfügen eines Objektes den Ergebnisbaum.

8 Punkte (b) Gegeben sei der folgende AVL-Baum:



Löschen Sie aus dem Baum das Element 10. Rebalancieren Sie den Baum unter Angabe der notwendig werdenden Rotationen, falls das AVL-Kriterium durch das Löschen verletzt wird. Markieren Sie wieder den betroffenen Knoten.

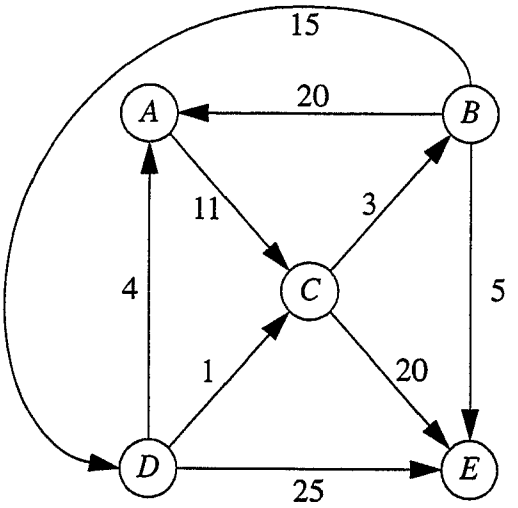
4 Punkte (c) Welcher Zeitbedarf entsteht im worst case für Suchen, Einfügen und Löschen im AVL-Baum? Begründen Sie Ihre Antworten.

Aufgabe 2 (Selection Sort) *12 Punkte*

Sortieren Sie die Zahlenfolge 35, 62, 28, 50, 11, 45 mittels *Selection Sort*. Zeigen Sie jeweils für jeden Wert der Laufvariable i das komplette Array, so wie es am Beginn bzw. am Ende eines jeden Schleifendurchlaufs aussieht. Markieren Sie jeweils durch Unterstreichen das bis dahin sortierte Teil-Array.

Aufgabe 3 (Floyds Algorithmus) *18 Punkte*

Berechnen Sie in dem unten gezeigten Graphen G mit dem Algorithmus aus dem Kurstext die kürzesten Pfade zwischen allen Knoten. Bearbeiten Sie die Knoten in lexikographischer Reihenfolge und geben Sie als Zwischenschritte den Graphen nach der vollständigen Bearbeitung der einzelnen Knoten an. Zeichnen Sie dabei die in einem Schritt neu eingefügten Kanten mit gestrichelten Linien.

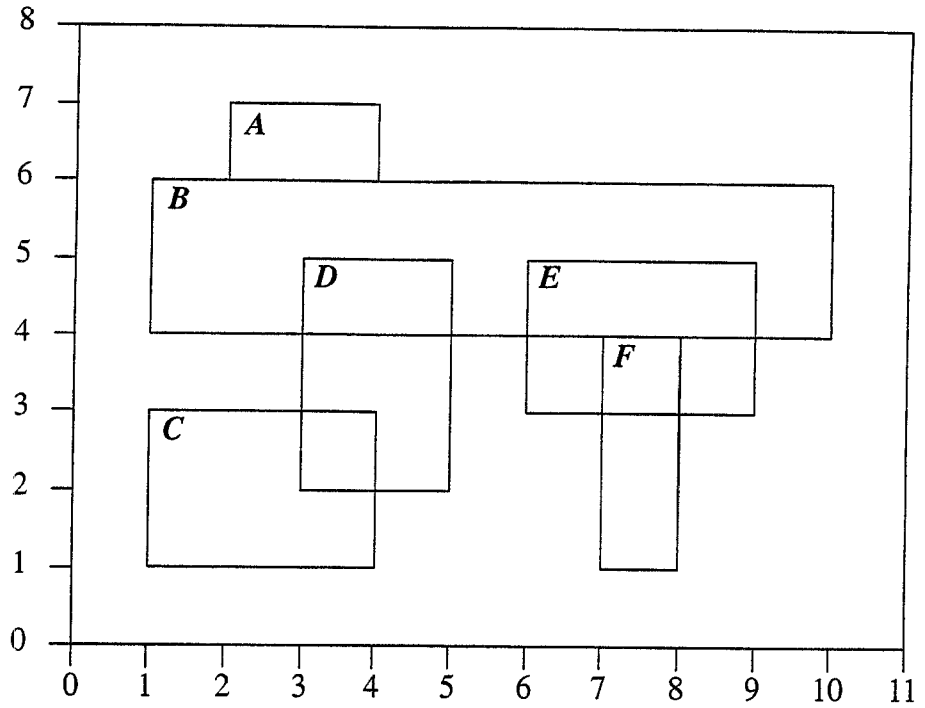


Aufgabe 4 (Rechteckschnitt-Problem) *15 Punkte*

(a) Das Rechteckschnitt-Problem kann auf das Segmentschnitt- und Punkteinschluss-Problem reduziert werden. Beschreiben Sie, wie diese Reduktion erfolgt. *5 Punkte*

(b) Erläutern Sie, welche Aktionen während des Sweeps beim Plane-Sweep-Algorithmus für das Punkteinschluss-Problem durchgeführt werden müssen. Erklären Sie außerdem, wozu hier ein Segmentbaum benötigt wird. *5 Punkte*

(c) Angenommen, für die im Bild dargestellten Rechtecke soll mittels Plane-Sweep das Punkteinschluss-Problem gelöst werden. Gehen Sie davon aus, dass sich die Sweep-Line von links nach rechts bewegt. Zeichnen Sie einen Segmentbaum, der den Zustand der Sweepline-Status-Struktur zwischen den Events, die bei x=3 und x=4 stattfinden, darstellt. *5 Punkte*



18 Punkte

Aufgabe 5 (Sortieren)

Gegeben sei ein Array A von n Integer-Zahlen aus dem Bereich 1, ..., k mit $k < n$, die in aufsteigender Reihenfolge zu sortieren sind. Duplikate sind hierbei erlaubt.

14 Punkte (a) Geben Sie einen möglichst effizienten Sortieralgorithmus an, der die Schranke $\Omega(n \log n)$ unterbietet, wenn zusätzlich ein temporärer Speicherplatzbereich der Länge k zur Verfügung steht.

4 Punkte (b) Bestimmen Sie die Zeitkomplexität Ihres Algorithmus.

17 Punkte

Aufgabe 6 (Rekursionsgleichungen)

6 Punkte (a) Lösen Sie die folgenden Rekursionsgleichungen auf. Ein formaler Beweis der Korrektheit ist nicht erforderlich. Vereinfachen Sie jeweils das Endergebnis mit Hilfe der O-Notation.

(i) $f(0) = 0$
$f(n) = f(n-1) + 2, n > 0$

(ii) $f(0) = 0$
$f(n) = 2f(n-1) + 1, n > 0$

(iii) $f(0) = 1$
$f(1) = 2$
$f(n) = 2f(n-1) - f(n-2), n > 1$

(b) Um das größte Element in einem Array A von N integer-Elementen zu finden, wird die Prozedur *Max* verwendet. Sie liefert das größte Element des n-elementigen Teil-Arrays $A[m, ..., m+n-1]$. Um das gesamte Array zu durchsuchen, wird Max(A, 1, N) aufgerufen. *6 Punkte*

```
procedure Max(A : array[1..N] of integer; m, n : integer) : integer;
var m1, m2 : integer;
begin
  if n = 1 then
    return A[m]
  else
    m1 := Max(A, m, ⌊n/2⌋);
    m2 := Max(A, m + ⌊n/2⌋, ⌈n/2⌉);
    if m1 > m2 then
      return m1
    else return m2
    fi
  fi
end Max;
```

Sei $M(n)$ eine Funktion, die die Kosten von *Max* in Abhängigkeit von ihrem dritten Argument n bestimmt. Definieren Sie M rekursiv. Die Kosten seien dabei bestimmt von den Vergleichsoperationen. Sie können davon ausgehen, dass n eine Zweierpotenz ist.

(c) Lösen Sie die rekursive Gleichung aus (b) auf. Zeigen Sie die Korrektheit Ihrer Lösung durch vollständige Induktion. *5 Punkte*