Name: Muster *Matrikelnummer:*

B E I S P I E L K L A U S U R
Informatik B
Juli 2003

Teil I: Informatik 3

Seite **1** von 19

ACHTUNG

Die vorliegende Beispielklausur wurde im Juli 2003 geschrieben. Sie umfaßt lediglich den **Informatik 3 Teil** der Informatik B Klausur und ist für eine Bearbeitungszeit von **120 Minuten** gedacht. In der realen Klausur standen für beide Teile (Info3 und Info4) jeweils 120 Minuten zur Verfügung!

Dieses Muster soll den *möglichen Aufbau* und die *Art der Fragestellung* des Info3-Teils der Klausur beispielhaft verdeutlichen. Wir erheben deshalb hier keinen Anspruch auf vollständige Stoffüberdeckung. In der realen Klausur können und werden also auch andere Stoffschwerpunkte geprüft werden. Weiterhin kann die Zusammensetzung ('Quickies', Textaufgaben, Handsimulationen, Programmieraufgaben) variieren.

Diese Beispielklausur soll Ihnen helfen, sich mit der Herangehensweise zum Lösen der Aufgaben vertraut zu machen. Es nutzt Ihnen nichts, wenn Sie die Aufgaben oder Lösungen dieser Beispielklausur auswendig lernen. Prüfungsrelevant ist der in der Veranstaltung (Vorlesung und Übungen) gelehrte Stoff!

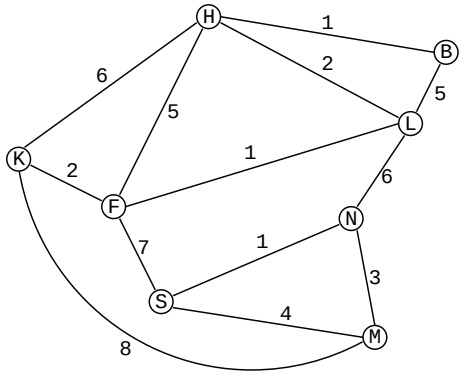
Im Juli 2003 bestand letztmalig die Möglichkeit, sich entweder über den von Prof. Heiß oder den von Prof. Wysotzki gelehrten Info3-Stoff prüfen zu lassen. Deshalb gab es in der Klausur die Aufgabe 4 in zwei Varianten, von denen lediglich eine zu bearbeiten war. In den kommenden Klausuren wird das wegfallen und es wird nur noch der von Prof. Heiß gelehrte Stoff geprüft.

Punkte	

Name: Muster *Matrikelnummer:*

Aufgabe 1: Graphen **(10 Punkte)**

(a) (2 Punkte) Auf der folgenden Graphik sind einige große Städte Deutschlands und die Zugverbindungen zwischen ihnen dargestellt.



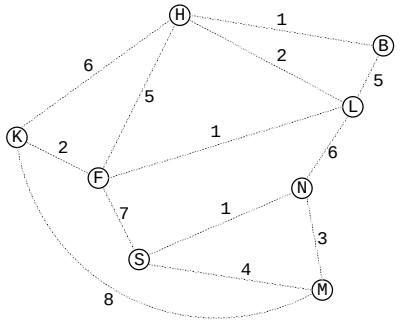
Dabei bedeuten die Abkürzungen *H*amburg, *B*erlin, *L*eipzig, *K*öln, *F*rankfurt, *N*ürnberg, *S*tuttgart, und *M*ünchen. Die Zahlen an den Kanten geben die Kosten für die Fahrt zwischen den an die entsprechende Kante angrenzenden Städten an.

Führen Sie eine Breitensuche von B nach S durch. Geben Sie dabei die Queue der bereits entdeckten (grauen) Knoten in jedem Schritt an. Die Reihenfolge der Behandlung der Nachbarn eines Knotens soll alphabetisch sein, d.h. $A < B < C < \ldots < Z$. Die Suche soll abbrechen, wenn das erste Mal der Zielknoten S in die Queue eingefügt wird. Sie können die Abkürzungen der Städtenamen verwenden.

Schritt	Queue
1	B
2	
3	
4	
5	
6	
7	
8	
9	

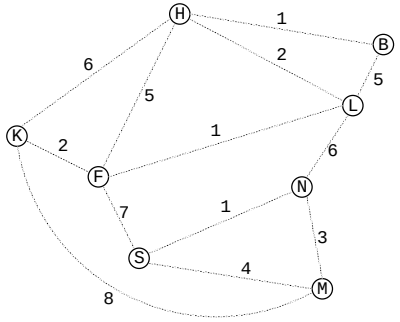
Punkte	

(b) **(1 Punkt)** Zeichnen Sie den in Aufgabenteil a entstandenen Breitensuchbaum in den Graphen ein.



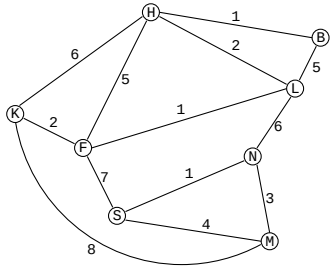
(c) **(2 Punkte)** Stellen Sie den Graph aus Aufgabenteil a als bewertete Adjazenzmatrix dar. Verwenden Sie als Reihenfolge B, H, L, K, F, N, S, M.

(d) **(1 Punkt)** Zeichnen Sie einen minimalen Spannbaum in den Graphen ein. Startknoten sei B.



Punkte	

(e) **(2 Punkte)** Führen Sie den Dijkstra-Algorithmus für den Graphen aus Aufgabenteil a durch. Startknoten sei B. Benutzen Sie die folgende Tabelle und geben Sie darin für jeden Schritt die ermittelten Kosten für einen Weg von B zu den anderen Knoten an. Machen Sie durch Einkreisen der Kosten deutlich, wann Relaxation stattfindet. Wenn ein Knoten endgültig bewertet wurde, wird dieses durch Unterstreichen kenntlich gemacht.



Schritt	B	H	L	K	F	N	S	M
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								

(f) **(1 Punkt)** Beschreiben Sie kurz die Problematik negativer Zyklen bei der Wegfindung mittels Dijkstra-Algorithmus.

(g) **(1 Punkt)** Wann ist ein ungerichteter Graph zusammenhängend? Kreuzen Sie die korrekte Antwort an.

Ein ungerichteter Graph ist zusammenhängend genau dann, wenn

- ◯ jeder Knoten von jedem anderen aus zu erreichen ist.
- ◯ es einen zyklenfreien Pfad zwischen einem ausgewählten Startknoten und einem ausgewählten Zielknoten gibt.
- ◯ es von jedem Knoten eine direkte Verbindung (ohne Zwischenknoten) zu jedem anderen Knoten gibt.
- ◯ es gerichtete Kanten gibt.

Punkte	

Name: Muster *Matrikelnummer:*

Aufgabe 2: Suchbäume **(14 Punkte)**

(a) (1 Punkt) Was sind *Randknoten* in einem Binärbaum? Kreuzen Sie *alle* richtigen Antworten an.

Randknoten sind
- ◯ Blätter.
- ◯ Knoten mit einem Nachfolger.
- ◯ Knoten mit zwei Nachfolgern.
- ◯ Knoten ohne Vorgänger.

(b) (1 Punkt) Was versteht man unter einem *vollständigen* Baum? Kreuzen Sie *alle* richtigen Antworten an.

Ein vollständiger Baum ist ein
- ◯ NP-vollständiger Baum.
- ◯ ein Baum, bei dem alle Blätter vollständig sind.
- ◯ ein Baum, bei dem alle Schichten vollständig sind.
- ◯ ein Binärbaum der Höhe n mit $2^{n+1} - 1$ Knoten.

(c) (1 Punkt) Wie hoch ist der Suchaufwand in einem degenerierten Binärbaum mit n Knoten? Kreuzen Sie die richtige Antwort an.

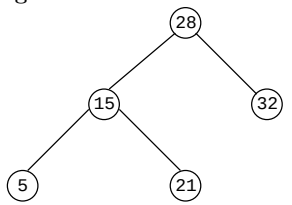
Der Suchaufwand in einem degenerierten Binärbaum mit n Knoten beträgt
- ◯ $\mathcal{O}(1)$.
- ◯ $\mathcal{O}(\log n)$.
- ◯ $\mathcal{O}(n)$.

(d) (1 Punkt) Zeichnen Sie einen maximal unausgewogenen AVL-Baum der Höhe 4.

Punkte	

Name: Muster *Matrikelnummer:*

Benutzen Sie für die folgenden Teilaufgaben e bis i den folgenden AVL-Baum *jeweils als Ausgangspunkt*. Nach jeder Operation soll die AVL-Eigenschaft wieder hergestellt werden.



Hinweis: Ihre Lösungen müssen nachvollziehbar sein!

(e) (1 Punkt) Fügen Sie in den gegebenen AVL-Baum (S. 6 oben) den Schlüssel 29 ein.

(f) (2 Punkte) Fügen Sie in den gegebenen AVL-Baum (S. 6 oben) den Schlüssel 3 ein.

(g) (2 Punkte) Fügen Sie in den gegebenen AVL-Baum (S. 6 oben) den Schlüssel 25 ein.

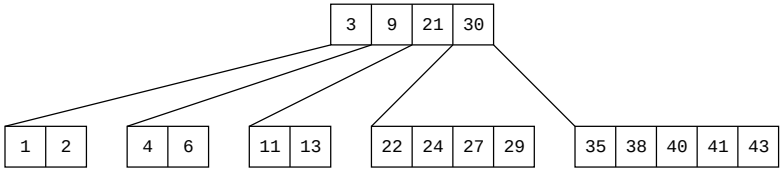
Punkte	

Name: Muster *Matrikelnummer:*

(h) (1 Punkt) Löschen Sie im gegebenen AVL-Baum (S. 6 oben) den Schlüssel 15.

(i) (2 Punkte) Löschen Sie im gegebenen AVL-Baum (S. 6 oben) den Schlüssel 32.

(j) (2 Punkte) Löschen Sie im gegebenen B-Baum der Ordnung 6 den Schlüssel 4.



Punkte	

Name: Muster *Matrikelnummer:*

Aufgabe 3: Hashing **(12 Punkte)**

(a) (4 Punkte) Fügen Sie die folgenden Schlüsselwerte in der angegebenen Reihenfolge mittels *double hashing* in eine Hashtabelle der Länge $n = 7$ ein.

$$K = \{31, 11, 5, 17, 25\}$$

Die benötigten Hashfunktionen lauten

$$\mathrm{h}_1(k) = k \bmod 7 \qquad \text{und} \qquad \mathrm{h}_2(k) = 1 + (k \bmod 5)\ ,$$

d.h. für die Indizes gilt gemäß Notation im Skript (Seite 3-19):

$$i_j = \begin{cases} \mathrm{h}_1(k) & \text{falls } j = 0 \\ (i_{j-1} + \mathrm{h}_2(k)) \bmod 7 & \text{falls } j \in \mathbb{N} \setminus \{0\} \quad (j > 0)\ . \end{cases}$$

Geben Sie für jeden Schlüssel jeweils alle getesteten Tabellenplätze i_j an. Geben Sie ausserdem die resultierende Hashtabelle (nach dem Einfügen aller Schlüssel) an, wobei leere Einträge durch den Wert -1 repräsentiert werden.

Punkte	

(b) **(1 Punkt)** Es sei K die Menge der Schlüsselwerte, die in einer Hashtabelle gespeichert sind. Man bezeichnet mit $\mathrm{t}(k)$ die *Suchzeit* eines Schlüsselwertes $k \in K$, also die Anzahl der Tabellenplätze, die man prüfen muß, um den Schlüssel k in der Hashtabelle zu finden. Ausgehend davon ist die *mittlere Suchzeit* für eine Hashtabelle definiert durch

$$\frac{1}{|K|} * \sum_{k \in K} \mathrm{t}(k) \ .$$

Geben Sie die mittlere Suchzeit für die fertige Hashtabelle aus Aufgabenteil a *nachvollziehbar* an.

(c) **(1 Punkt)** Geben Sie für die Hashfunktion aus Teilaufgabe a eine Folge mit 7 paarweise verschiedenen Schlüsseln an, so daß nach dem Einfügen der Folge in eine leere Hashtabelle mit $n = 7$ Plätzen eine *minimale* mittlere Suchzeit resultiert.

(d) **(2 Punkte)** Geben Sie für die Hashfunktion aus Teilaufgabe a eine Folge mit 7 paarweise verschiedenen Schlüsseln an, so daß nach dem Einfügen der Folge in eine leere Hashtabelle mit $n = 7$ Plätzen eine *maximale* mittlere Suchzeit resultiert. Begründen Sie Ihre Wahl!

Punkte	

(e) **(4 Punkte)** Implementieren Sie die Funktionen aus den Aufgabenteilen a und b (zum Einfügen von Schlüsseln in eine Hashtabelle mittels *double hashing* und zum Berechnen der mittleren Suchdauer) in Java.

Fassen Sie die Hashfunktionen $\mathrm{h}_1(k)$ und $\mathrm{h}_2(k)$ zu einer geschlossenen Hashfunktion $\mathrm{h}(k, j)$ zusammen. Beachten Sie, daß im Unterschied zum Aufgabenteil a die Länge der Hashtabelle beliebig sein soll. Der Teiler für die Modulo-Operation in $\mathrm{h}_2(k)$ soll um 2 kleiner als der entsprechende Teiler von $\mathrm{h}_1(k)$ sein. Der Teiler für $\mathrm{h}_1(k)$ wiederum wird als Argument (in Abhängigkeit von der konkreten Tabellenlänge) übergeben.

Implementieren Sie eine Methode zum Einfügen von Schlüsseln (mittels *double hashing*), die in einem Array gegeben sind (die Schlüssel seien vom Typ `int`). Jeder Schlüssel soll nur einmal (d.h. nicht mehrfach) in die Tabelle eingefügt werden. Die gefüllte Hashtabelle soll zurückgeliefert werden. Die Hashtabelle kann als (mit -1 initialisiertes) Array über Einträgen vom Typ `int` dargestellt und als Argument übergeben werden.

Implementieren Sie außerdem eine Methode zur Berechnung der mittleren Suchzeit. Hierbei soll eine gefüllte Hashtabelle und ein Array mit den zu suchenden Schlüsseln übergeben werden und die mittlere Suchzeit zurückgeliefert werden. Gehen Sie dabei davon aus, daß aus der übergebenen Hashtabelle keine Daten gelöscht wurden (d.h. es wurden ausschließlich Einfügeoperationen durchgeführt).

Punkte	

Name: Muster *Matrikelnummer:*

Name: Muster *Matrikelnummer:*

Aufgabe 4 (Variante A): Stoff nach Prof. Wysotzki **(14 Punkte)**

ACHTUNG! Lösen Sie von dieser Aufgabe *ENTWEDER* Variante A (Stoff nach Prof. Wysotzki, Seiten 12 bis 15) *ODER* Variante B (Stoff nach Prof. Heiß, Seite 16 bis 19)!
Es wird nur *EINE* der beiden Varianten gewertet!

(a) (1 Punkt) Was bedeutet *greedy-lösbar*? Kreuzen Sie die richtige Antwort an.

Ein Problem ist greedy-lösbar genau dann, wenn

- ◯ ein greedy-Algorithmus angewendet wird.
- ◯ es ein globales Optimum gibt.
- ◯ es ein lokales Entscheidungskriterium gibt, welches zum globalen Optimum führt.
- ◯ es ein Kriterium gibt, nach dem auf dem Weg durch den Suchbaum bei jedem Schritt lokal entschieden werden kann, in welche Richtung die Suche weitergehen soll.

(b) (1 Punkt) Welche der folgenden Suchalgorithmen finden für Graphen mit nicht-negativen Kantenbewertungen garantiert den optimalen Weg, falls ein solcher existiert? Kreuzen Sie die richtigen Antworten an.

- ◯ Greedy-Suchverfahren
- ◯ Gradientenmethode
- ◯ Best-First-Search
- ◯ Branch-and-Bound
- ◯ A*-Algorithmus

(c) (1 Punkt) Welcher der folgenden Suchalgorithmen baut zum Finden der optimalen Lösung *stets* den gesamten Suchbaum auf? Kreuzen Sie die richtige Antwort an.

- ◯ Greedy-Algorithmus
- ◯ Tiefensuche
- ◯ Breitensuche
- ◯ Branch-and-Bound
- ◯ A^*-Algorithmus

(d) (1 Punkt) In welcher Hinsicht ist Branch-and-Bound als Spezialfall von A^* zu sehen?

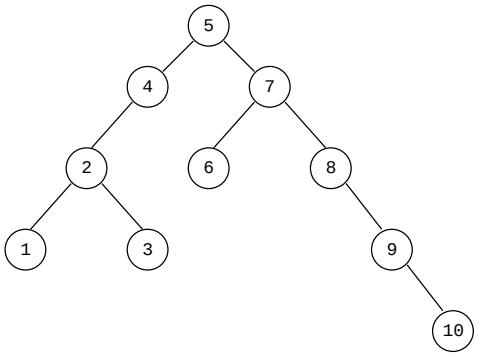
(e) **(1 Punkt)** Was ist ein c-höhenbalancierter Binärbaum? Kreuzen Sie die richtige Antwort an.

Ein c-höhenbalancierter Binärbaum ist ein Binärbaum, bei dem

- ◯ sich die Zahl der Knoten in den Unterbäumen jedes Knotens um höchstens eine Konstante c unterscheiden darf.
- ◯ sich die Länge der jeweils maximalen Pfade in den Unterbäumen jedes Knotens um höchstens eine Konstante c unterscheiden darf.
- ◯ die Schlüssel in den Knoten eine Invarianz erfüllen.

(f) **(2 Punkte)** Betrachten Sie den folgenden Binärbaum und kreuzen Sie in der unten stehenden Tabelle *alle* zutreffenden Konstanten bezüglich der Ausgewogenheit des Baumes an.

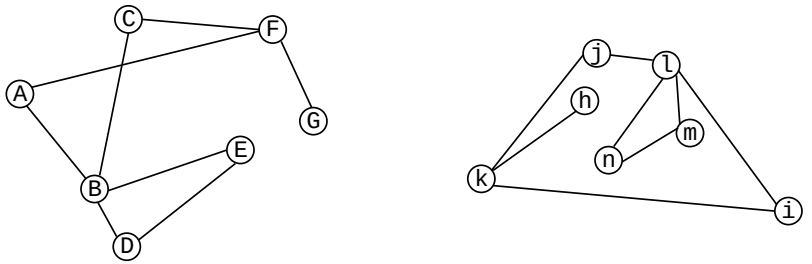
Hinweis: $\frac{5}{11} = 0,45\overline{45}$



Konstante c bzw. α	$\frac{1}{10}$	$\frac{1}{5}$	$\frac{1}{3}$	1	2	3
c-höhenbalanciert						
α-gewichtsbalanciert						

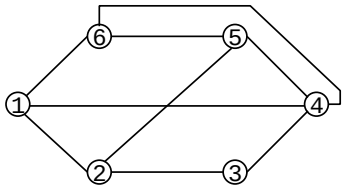
Punkte	

(g) **(2 Punkte)** Betrachten Sie die folgenden beiden Graphen.



Sind die Graphen isomorph zueinander? Begründen Sie Ihre Antwort, indem Sie ein entsprechendes Mapping der Knoten angeben (bei Isomorphie ein *passendes* Mapping, bei Nichtisomorphie ein Gegenbeispiel).

(h) **(2 Punkte)** Färben Sie den folgenden Graphen mit einer minimalen Zahl von Farben so ein, daß keine benachbarten Knoten dieselbe Farbe haben. Geben Sie die chromatische Zahl des Graphen an.
Hinweis: Verwenden Sie Farben aus folgender Menge:
Farben = {rot, grün, gelb, blau, schwarz, weiß}.



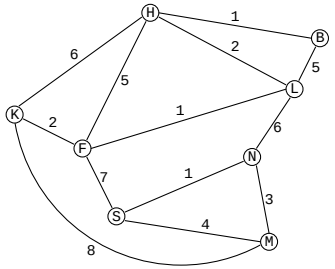
Knoten	1	2	3	4	5	6
Farbe						

Chromatische Zahl:

Punkte	

Name: Muster *Matrikelnummer:*

(i) (3 Punkte) Betrachten Sie den folgenden (bereits aus Aufgabe 1 bekannten) Graphen.



Zusätzlich sind folgende Schätzungen $h^*(k)$ der Restwegkosten von einem Knoten k zum Zielknoten S gegeben:

Knoten	B	F	H	K	L	M	N	S
$h^*(k)$	10	6	9	6	6	3	1	0

Zeigen Sie durch Handsimulation, wie der A^*-Algorithmus den kürzesten Weg vom Startknoten B zum Zielknoten S findet. Stellen Sie dabei in der Tabelle den Zustand der Queue *nach jedem* Schritt dar. Beachten Sie auch folgende Hinweise:

- Sie brauchen die Queue nicht sortieren, wenn Sie die im nächsten Schritt zu expandierende Teillösung durch Unterstreichen markieren.
- Falls in der Queue mehrere Wege mit unterschiedlichen Bewertungen zum gleichen Zwischenknoten führen, so wird nur der am besten bewertete Weg behalten ('dynamische Programmierung'). Machen Sie dies mittels Durchstreichen der zu löschenden Wege kenntlich. Zyklen sind zu vermeiden.
- Falls es in der Queue mehrere Wege mit gleicher Bewertung gibt, so sollen diese Wege alphabetisch nach ihrem letzten Knoten sortiert werden:
  $B < F < H < K < L < M < N < S$.

Schritt	Queue
1	(B, 10)
2	
3	
4	
5	
6	
7	
8	

Punkte	

Name: Muster *Matrikelnummer:*

Aufgabe 4 (Variante B): Stoff nach Prof. Heiß **(14 Punkte)**

ACHTUNG! Lösen Sie von dieser Aufgabe *ENTWEDER* Variante A (Stoff nach Prof. Wysotzki, Seiten 12 bis 15) *ODER* Variante B (Stoff nach Prof. Heiß, Seite 16 bis 19)!
Es wird nur *EINE* der beiden Varianten gewertet!

(a) (1 Punkt) Wie ist die asymptotische Laufzeit des Dijkstra-Algorithmus, wenn man statt der Heap-basierten Prioritätswarteschlange eine verkettete Liste verwendet? Kreuzen Sie die richtige Antwort an!

- ◯ $\mathcal{O}(E^2)$
- ◯ $\mathcal{O}(V^2)$
- ◯ $\mathcal{O}(EV)$
- ◯ $\mathcal{O}(E \log V)$
- ◯ $\mathcal{O}(V \log E)$

(b) (1 Punkt) Welche der folgenden Algorithmen besitzen eine Greedy-Struktur? Kreuzen Sie *alle* richtigen Antworten an!

- ◯ Quicksort (Sortieren)
- ◯ Dijkstra (Kürzeste Wege)
- ◯ Prim (Minimalbaum)
- ◯ Kruskal (Minimalbaum)

(c) (1 Punkt) Mit welchem Aufwand lässt sich in einem gerichteten Graphen mit n Knoten und m Kanten eine topologische Sortierung durchführen?

(d) (1 Punkt) Wann besitzt ein Problem eine 'optimale Substruktur'?

Punkte	

Name: Muster *Matrikelnummer:*

(e) (3 Punkte) Gegeben sei folgende Rekurrenzgleichung:

$$\mathrm{a}(i,k) = \begin{cases} 0 & \text{falls } i = 0 \wedge k = 0 \\ 1 & \text{falls } i = 0 \wedge k > 0 \\ 1 & \text{falls } i > 0 \wedge k = 0 \\ \mathrm{a}(i-1,k) + \mathrm{a}(i,k-1) & \text{falls } i > 0 \wedge k > 0 \end{cases}$$

mit $i, k \in \mathbb{N} \cup \{0\}$. Geben Sie ein möglichst effizientes Java-Programm mit polynomieller Laufzeit an, das bei Eingabe von m und n den Wert von $\mathrm{a}(m,n)$ berechnet.

(f) (1 Punkt) Welche Komplexität hat Ihre Berechnung aus Aufgabenteil e?

Punkte	

Name: Muster *Matrikelnummer:*

(g) (3 Punkte) Nehmen Sie an, Sie wollen das Traveling Salesman Problem (TSP, n Städte, Euklidische Distanz) mit einem Genetischen Algorithmus lösen.

1. Wie codieren Sie eine Lösung (formal)?
2. Geben Sie einen für Ihre Codierung geeigneten Kreuzungsoperator an (formal oder informal). Begründen Sie Ihre Wahl kurz!

Punkte	

Name: Muster *Matrikelnummer:*

(h) **(3 Punkte)** Gegeben seien m unterschiedliche Prozesse und n identische Prozessoren (mit $n \ll m$). Jeder Prozess i habe einen Rechenbedarf von a_i ($i = 1, \ldots, m$) Sekunden. Gesucht ist eine Zuordnung der Prozesse zu den Prozessoren, so daß die Gesamtbearbeitungszeit minimal wird.

Formulieren Sie das Problem als heuristisches Suchproblem.

1. Wie codieren Sie eine Lösung (formal)?
2. Wie lautet die Zielfunktion (formal)?
3. Wie sieht bei Ihrer Codierung ein Elementarschritt aus?

Punkte	