Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr
2020

46115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Theoretische Informatik/Algorithmus/Datenstrukturen**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 10

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

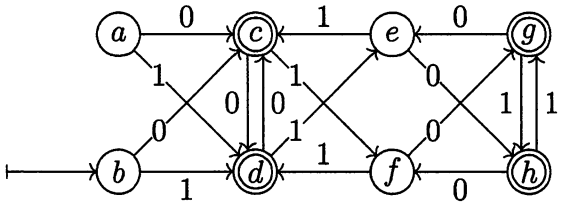
Aufgabe 1 (Reguläre Sprachen) [35 PUNKTE]

(a) [4 Punkte] Betrachten Sie die formale Sprache $L \subseteq \{0,1\}^*$ aller Wörter, die 01 oder 110 als Teilwort enthalten.

Geben Sie einen regulären Ausdruck für die Sprache L an.

(b) [10 Punkte] Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L aus Teilaufgabe (a) akzeptiert. (Hinweis: es werden nicht mehr als 6 Zustände benötigt.)

(c) [15 Punkte] Minimieren Sie den folgenden deterministischen endlichen Automaten:



Machen Sie dabei Ihren Rechenweg deutlich!

(d) [6 Punkte] Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort!

„Zu jeder regulären Sprache L über dem Alphabet Σ gibt es eine Sprache $L' \subseteq \Sigma^*$, die L enthält (d. h. $L \subseteq L'$) und *nicht* regulär ist."

Aufgabe 2 (Kontextfreie Sprachen) [32 PUNKTE]

(a) [4 Punkte] Betrachten Sie die formale Sprache L über dem Alphabet $\Sigma = \{a, b, c\}$ aller Wörter wv mit $w \in \{a, b\}^*$ und $v \in \{b, c\}^*$ so dass $w = b^k a^n$ und $v = (cb)^k$ für $n, k \in \mathbb{N}_0$.
Geben Sie zwei Wörter über Σ an, die in L enthalten sind und zwei Wörter, die nicht in L enthalten sind. Die Wörter sollen mindestens Länge 5 haben.

(b) [10 Punkte] Geben Sie eine kontextfreie Grammatik für die Sprache L an.

Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

(c) [2 Punkte] Geben Sie für eines Ihrer beiden Wörter aus Teilaufgabe (a), das in L enthalten ist, einen Ableitungs*baum* des Wortes mit Ihrer Grammatik aus Teilaufgabe (b) an.

(d) [10 Punkte] Beweisen Sie, dass die Sprache L aus Teilaufgabe (a) nicht regulär ist.

Fortsetzung nächste Seite!

(e) [6 Punkte] Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort!

„Wenn L eine kontextfreie und L' eine Sprache aus der Klasse $\mathcal{P}$ ist, dann ist die Konkatenation LL' wieder kontextfrei."

Aufgabe 3 (Entscheidbarkeit) [23 PUNKTE]

(a) [8 Punkte] Betrachten Sie das folgende Entscheidungsproblem:

Eingabe: eine (geeignet codierte) Turingmaschine M

Aufgabe: entscheiden, ob die Turingmaschine M auf *jedes* Eingabewort nach höchstens 42 Schritten hält.

Ist dieses Problem entscheidbar? Begründen Sie Ihre Antwort.

(b) [15 Punkte] Beweisen Sie mit Hilfe eines Reduktionsbeweises, dass das folgende Problem nicht entscheidbar ist:

Eingabe: zwei (geeignet codierte) Turingmaschinen M_1, M_2 sowie ein Eingabewort w

Aufgabe: entscheiden, ob M_1 auf Eingabewort w hält und M_2 auf w *nicht* hält.

Aufgabe 4 (Korrektheit von Algorithmen) [26 PUNKTE]

Wir betrachten den nachstehenden Algorithmus:

```
MinSuffixAverage(A[1..n])
  // Eingabe: Ein Feld A von n ≥ 1 Zahlen
  // Ausgabe: min_{1≤i≤n} 1/(n−i+1) Σ_{k=i}^{n} A[k]
1 b[0] = ∞
2 a[0] = 0
3 for s = n downto 1
4       l = n − s + 1
5       a[l] = ((n−s)·a[l−1]+A[s]) / (n−s+1)
6       b[l] = min(b[l − 1], a[l])
7 return b[n]
```

(a) [3 Punkte] Begründen Sie, dass der Wert der Variablen l der Nummer des aktuellen Schleifendurchlaufs entspricht (wenn wir mit dem Zählen der Schleifendurchläufe bei 1 beginnen).

Wir wollen beweisen, dass der Algorithmus korrekt ist, d. h., dass er für die Eingabe $A[1,\ldots,n]$ den Wert

$$\min_{1\le i\le n} \frac{1}{(n-i+1)} \sum_{k=i}^{n} A[k]$$

zurückgibt. Für $1 \le l \le n$ seien dazu folgende Werte definiert:

$$a_l := \frac{1}{l} \sum_{k=n-(l-1)}^{n} A[k] \qquad \text{und} \qquad b_l := \min_{n-(l-1)\le i\le n} \frac{1}{(n-i+1)} \sum_{k=i}^{n} A[k]$$

Fortsetzung nächste Seite!

Wir wollen durch Induktion zeigen, dass für alle $1 \leq l \leq n$ die Aussage

$$\mathcal{A}_l : \text{Nach dem } l\text{-ten Durchlauf der Schleife ist } a[l] = a_l$$

gilt.

(b) [2 Punkte] Führen Sie den Induktionsanfang $l = 1$ aus.

(c) [8 Punkte] Führen Sie den Induktionsschritt $l - 1 \to l$ (für $1 < l \leq n$) aus.

Wir wollen als nächstes durch Induktion zeigen, dass für alle $1 \leq l \leq n$ die Aussage

$$\mathcal{B}_l : \text{Nach dem } l\text{-ten Durchlauf der Schleife ist } b[l] = b_l$$

gilt.

(d) [2 Punkte] Führen Sie den Induktionsanfang $l = 1$ aus.

(e) [9 Punkte] Führen Sie den Induktionsschritt $l - 1 \to l$ (für $1 < l \leq n$) aus.
Hinweis: Verwenden Sie das Resultat aus der vorigen Aufgabe: $a[l] = a_l$ für alle $1 \leq l \leq n$.

(f) [2 Punkte] Folgern Sie, dass der Algorithmus MINSUFFIXAVERAGE korrekt ist.

Aufgabe 5 (Naive Polynomarithmetik) [15 PUNKTE]

In den folgenden Aufgaben nehmen wir an, dass ein Polynom

$$A = \sum_{0 \leq i < n} a_i x^i \in \mathbb{Z}[x]$$

vom Grad $< n$ im Rechner durch ein Array $K_A[0, \ldots, n-1]$ mit $K_A[i] = a_i$ dargestellt wird.

(a) [3 Punkte] Geben Sie einen Algorithmus zur Addition zweier Polynome in Pseudo-Code an:

PADD(A, B, n)
// Eingabe: Zwei Polynome $A, B \in \mathbb{Z}[x]$ vom Grad kleiner n mit
// $A = \sum_{0 \leq i < n} a_i x^i$ und $B = \sum_{0 \leq i < n} b_i x^i$
// Ausgabe: Das Polynom $C = \sum_{0 \leq i < n} c_i x^i$ mit $C = A + B$

(b) [2 Punkte] Bestimmen Sie die Laufzeit Ihrer Implementierung beim Aufruf PADD(A, B, n).

(c) [4 Punkte] Geben Sie einen Algorithmus zur Multiplikation zweier Polynome in Pseudo-Code an:

PMULT(A, B, n)
// Eingabe: Zwei Polynome $A, B \in \mathbb{Z}[x]$ vom Grad kleiner n mit
// $A = \sum_{0 \leq i < n} a_i x^i$ und $B = \sum_{0 \leq i < n} b_i x^i$
// Ausgabe: Das Polynom $C = \sum_{0 \leq i < 2n-1} c_i x^i$ mit $C = AB$

Fortsetzung nächste Seite!

(d) [2 Punkte] Bestimmen Sie die Laufzeit Ihrer Implementierung beim Aufruf PMULT(A, B, n).

(e) [2 Punkte] Beschreiben Sie einen Algorithmus PMULTAMONOM(A, B, n) der $C = AB$ in $O(n)$ Zeit berechnet, falls A ein Monom (d. h. ein Polynom von der Form $a_k x^k$ mit $k < n$) ist.

(f) [2 Punkte] Bestimmen Sie die Laufzeit Ihrer Implementierung beim Aufruf PMULTAMONOM(A, B, n).

Aufgabe 6 (Münzwechseln) [49 PUNKTE]

Gegeben sei eine *feste* Menge $\mathcal{D} = \{d_1, \ldots, d_m\}$ von $m > 0$ natürlichen Zahlen $1 = d_1 < \cdots < d_m$, die wir *Münzwerte* nennen.

(a) [1 Punkt] Begründen Sie, dass sich jede Zahl $s \in \mathbb{N}$ in der Form

$$s = \sum_{i=1}^{m} c_i \cdot d_i$$

mit $c, \in \mathbb{N}_0$ schreiben lässt.

Wenn

$$s = \sum_{i=1}^{m} c_i \cdot d_i,$$

so nennen wir die Folge $C = (c_1, \ldots, c_m)$ der Länge m eine *Darstellung* von s (bezüglich der Münzwerte $\mathcal{D}$). Wir interpretieren dies so, dass sich (der Geldwert) s mithilfe der m verschiedenen Münzwerte die uns zur Verfügung stehen, wechseln lässt: Wenn wir für $1 \leq i \leq m$ gerade c_i Münzen vom Wert d_i nehmen, erhalten wir insgesamt s. Den Wert

$$\text{cost}(C) := \sum_{i=1}^{m} c_i$$

nennen wir die *Kosten* der Darstellung C. Die Kosten einer Darstellung von s zählen also einfach nur, wie viele Münzen wir insgesamt benötigen, um s gemäß C zu wechseln.

Wir suchen eine Darstellung $C^* = (c_1^*, \ldots, c_m^*)$ von s mit *minimalen* Kosten (d. h. wir wollen möglichst *wenige* Münzen verwenden). So eine Darstellung von s nennen wir *optimal*, ihre Kosten bezeichnen wir mit $OPT(s) := OPT(C^*)$; für $s < 0$ definieren wir (zur Vereinfachung der Notation) $OPT(s) := \infty$.

(b) [3 Punkte] Was ist der Wert $OPT(0)$? Was sind die Werte $OPT(d_i)$ für $1 \leq i \leq m$?

(c) [5 Punkte] Sei $C^* = (c_1^*, \ldots, c_i^*, \ldots, c_m^*)$ eine optimale Darstellung von $s > 0$ mit $c_i^* \geq 1$ für ein $1 \leq i \leq m$. Beweisen Sie, dass dann $C := (c_1^*, \ldots, c_i^* - 1, \ldots, c_m^*)$ eine optimale Darstellung von $(s - d_i)$ ist.

(d) [6 Punkte] Beweisen Sie, dass für $s > 0$

$$OPT(s) = 1 + \min_{1 \leq i \leq m} OPT(s - d_i).$$

Fortsetzung nächste Seite!

(e) [5 Punkte] Formulieren Sie auf Basis der Rekursionsgleichung aus Teilaufgabe (d) einen rekursiven Algorithmus zur Berechnung von $OPT(s)$ in Pseudocode:

CHANGEREC (s, D)
// Eingabe: Eine Instanz des Münzwechselproblems, beschrieben durch
// den Zielwert $s \geq 0$ und
// die Folge $D = (d_1, ..., d_m)$ der Münzwerte mit $1 = d_1 < ... < d_m$
// Ausgabe: $OPT(s)$

(f) [4 Punkte] Wir betrachten nun die Menge $\mathcal{D} = \{1, 2\}$ von Münzwerten. Begründen Sie, dass für dieses $\mathcal{D}$ die Laufzeit $T(s)$ des rekursiven Algorithmus beim Aufruf CHANGEREC(s, D) folgender asymptotischer Rekursionsgleichung genügt:

$$T(s) = T(s-1) + T(s-2) + \Theta(1)$$
$$(\text{mit } T(s) = \Theta(1) \text{ für } s = O(1))$$

(g) [6+2 Punkte] Wir betrachten die Rekursionsungleichung (für eine Konstante $c > 0$)

$$T(s) \geq T(s-1) + T(s-2) + c \text{ für } s > 1 \qquad \text{und} \qquad T(s) \geq c \text{ für } s \leq 1.$$

Beweisen Sie, dass $T(s) = \Omega(\sqrt{2}^s)$.

Argumentieren Sie damit, warum eine direkte Implementierung der Rekursion von CHANGEREC im Allgemeinen nicht effizient ist.

(h) [6+8+3 Punkte] Formulieren Sie ein dynamisches Programm zur Berechnung von $OPT(s)$ in Pseudocode:

CHANGEDP (s, D)
// Eingabe: Eine Instanz des Münzwechselproblems, beschrieben durch
// den Zielwert $s \geq 0$ und
// die Folge $D = (d_1, ..., d_m)$ der Münzwerte mit $1 = d_1 < ... < d_m$
// Ausgabe: $OPT(s)$

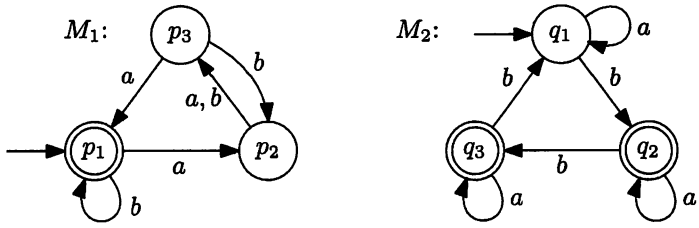
Begründen Sie, warum Ihr Algorithmus korrekt ist und analysieren Sie die dessen Laufzeit.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1 (Automatentheorie und reguläre Sprachen) [44 PUNKTE]

Gegeben seien die beiden folgenden deterministischen endlichen Automaten $M_1 = (Q_1, \{a, b\}, \delta_1, p_1, F_1)$ und $M_2 = (Q_2, \{a, b\}, \delta_2, q_1, F_2)$.



(a) [16 Punkte] Konstruieren Sie den Produktautomaten M_3 mit $L(M_3) = L(M_1) \cap L(M_2)$

(b) [6 Punkte] Welche Veränderungen an Ihrem Automaten M_3 sind nötig, um einen Automaten M_3' zu erhalten, mit $L(M_3') = L(M_1) \setminus L(M_2)$?

Hinweis: Eine explizite Angabe des modifizierten Automaten, etwa in Form eines Zustandsübergangsdiagramms, ist *nicht* erforderlich.

(c) [16 Punkte] Geben Sie einen regulären Ausdruck α mit $L(\alpha) = L(M_1)$ an. Gehen Sie systematisch vor und dokumentieren Sie Ihr Vorgehen geeignet.

(d) [6 Punkte] Gibt es zwei nicht reguläre Sprachen L_1 und L_2 deren Schnitt $L_1 \cap L_2$ regulär ist? Begründen Sie Ihre Antwort.

Aufgabe 2 (Turingmaschinen) [30 PUNKTE]

Ziel dieser Aufgabe ist es, eine deterministische 1-Band-Turingmaschine zu konstruieren, die für jedes Eingabewort $w \in \{a, b\}^*$ das Wort $w\#w$ berechnet und dann hält. Dabei ist $\#$ ein Platzhaltersymbol, das nicht im Eingabewort vorkommt.

(a) [8 Punkte] Beschreiben Sie das Vorgehen einer solchen Turing-Maschine in Worten.

(b) [18 Punkte] Geben Sie ein Zustandsübergangsdiagramm einer Turingmaschine an, die diese Berechnung durchführt.

(c) [4 Punkte] Dokumentieren Sie, welche Zustände Ihrer Turingmaschine Ihre Schritte aus Teilaufgabe (a) implementieren.

Fortsetzung nächste Seite!

Aufgabe 3 (Berechenbarkeitstheorie) [16 PUNKTE]

Für ein Wort $w \in \Sigma^*$ bezeichne w^r das Umkehrwort, das aus w entsteht, indem man die Zeichenfolge umdreht. Für eine beliebge Sprache A bezeichne $A^r = \{w^r \mid w \in A\}$ die Umkehrsprache.

Angenommen, A ist unentscheidbar (= nicht rekursiv). Zeigen Sie, dass dann auch A^r unentscheidbar ist.

Aufgabe 4 (Sortieren) [9 PUNKTE]

Ein bekanntes Sortierverfahren ist Insertion-Sort, ein inkrementeller Algorithmus. Insertion-Sort geht im i-ten Durchlauf davon aus, dass das Teilfeld bis zur Stelle $i-1$ schon sortiert ist. Dann speichert Insertion-Sort das i-te Element des gegebenen Feldes zwischen, um es an der korrekten Stelle im schon sortierten Teilfeld einzufügen.

(a) [3 Punkte] Analysieren Sie, wie viele Schlüsselvergleiche Insertion-Sort im besten und im schlechtesten Fall benötigt, um ein Feld von n verschiedenen Zahlen zu sortieren. Geben Sie jeweils die genaue Anzahl an.

(b) [6 Punkte] Sei nun die Eingabe ein Feld mit den Zahlen $1, 2, \ldots, n$ in *zufälliger* Reihenfolge. Schätzen Sie asymptotisch scharf ab, wie viele Schlüsselvergleiche Insertion-Sort in diesem Fall vornimmt.

Tipp: Gehen Sie davon aus, dass n durch 4 teilbar ist. Betrachten Sie der Einfachheit halber nur die Elemente im letzten Viertel des Feldes. Wie weit müssen manche (wie viele?) dieser Elemente nach links wandern?

Aufgabe 5 (Dijkstras Algorithmus) [21 PUNKTE]

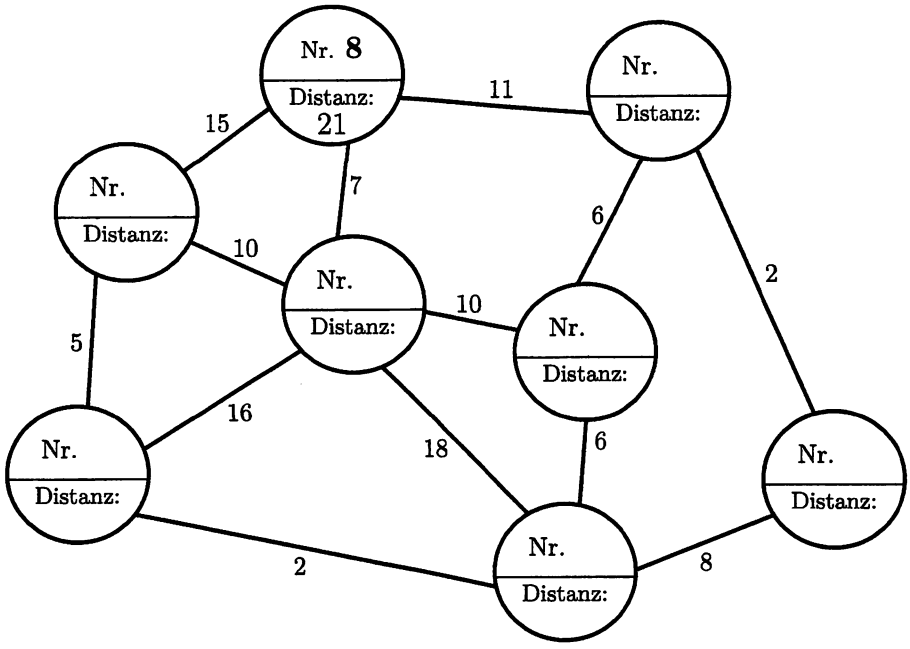
Auf folgendem ungerichteten Graphen wurde Dijkstras Algorithmus (wie unten beschrieben) ausgeführt, doch wir wissen lediglich, welcher Knoten als letztes schwarz (*black*) wurde (Nr. 8) und was seine Distanz zum Startknoten (Nr. 1) ist. Die Gewichte der Kanten sind angegeben.

Finden Sie den Startknoten, nummerieren Sie die Knoten in der Reihenfolge, in der sie schwarz wurden und geben Sie in jedem Knoten die Distanz zum Startknoten an!

Markieren Sie die Kanten, die zu dem Kürzesten-Wege-Baum gehören, den Dijkstras Algorithmus ausgibt.

Der Startknoten ist eindeutig!

Fortsetzung nächste Seite!



Dijkstra(Graph G, Edgeweights w, Vertex s)

```
Initialize(G, s)
S = ∅
Q = new PriorityQueue(V, d)
while not Q.Empty() do
    u = Q.ExtractMin()
    S = S ∪ {u}
    foreach v ∈ Adj[u] do
        Relax(u, v, w)
    u.color = black
```

Initialize(Graph G, Vertex s)

```
foreach u ∈ V do
    u.color = white
    u.d = ∞
s.color = gray
s.d = 0
```

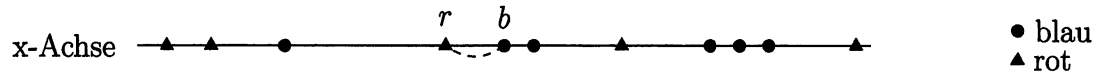
Relax(Vertex u, Vertex v, Edgeweights w)

```
if v.d > u.d + w(u, v) then
    v.color = gray
    v.d = u.d + w(u, v)
    Q.DecreaseKey(v, v.d)
```

Fortsetzung nächste Seite!

Aufgabe 6 (Nächstes rot-blaues Paar auf der x-Achse) [40 PUNKTE]

Gegeben seien zwei nichtleere Mengen R und B von roten bzw. blauen Punkten auf der x-Achse. Gesucht ist der minimale euklidische Abstand $d(r, b)$ über alle Punktepaare (r, b) mit $r \in R$ und $b \in B$. Hier ist eine Beispielinstanz:



Die Eingabe wird in einem Feld A übergeben. Jeder Punkt $A[i]$ mit $1 \leq i \leq n$ hat eine x-Koordinate $A[i].\mathrm{x}$ und eine Farbe $A[i].\mathrm{color} \in \{$ rot, blau $\}$. Das Feld A ist nach x-Koordinate sortiert, d.h. es gilt $A[1].\mathrm{x} < A[2].\mathrm{x} < \cdots < A[n].\mathrm{x}$, wobei $n = |R| + |B|$.

(a) [10 Punkte] Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in $O(n)$ Zeit berechnet.

(b) [10 Punkte] Begründen Sie kurz die Laufzeit Ihres Algorithmus.

(c) [10 Punkte] Begründen Sie die Korrektheit Ihres Algorithmus.

(d) [10 Punkte] Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in $o(n)$ Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit $\Theta(n)$ haben.)

Aufgabe 7 (Heaps) [20 PUNKTE]

Sei H ein Max-Heap, der n Elemente speichert. Für ein Element v in H sei $h(v)$ die Höhe von v, also die Länge eines längsten Pfades von v zu einem Blatt im Teilheap mit Wurzel v.

(a) [6 Punkte] Geben Sie eine rekursive Definition von $h(v)$ an, in der Sie sich auf die Höhen der Kinder $v.\mathrm{left}$ und $v.\mathrm{right}$ von v beziehen (falls v Kinder hat).

(b) [8 Punkte] Geben Sie eine möglichst niedrige obere asymptotische Schranke für die Summe der Höhen aller Elemente in H an, also für $\sum_{v \in H} h(v)$ und begründen Sie diese.

Tipp: Denken Sie daran, wie man aus einem beliebigen Feld einen Max-Heap macht.

(c) [6 Punkte] Sei H' ein Feld der Länge n. Geben Sie einen Algorithmus an, der in Linearzeit testet, ob H' ein Max-Heap ist.