7. (16 Punkte)

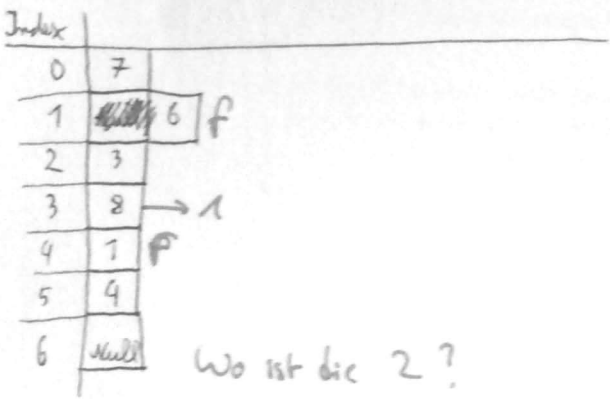
Gegeben sei die Hashfunktion $h(x) \stackrel{\text{def}}{=} 3x \mod 7$ und die Sondierungsfunktion $s(x,i) \stackrel{\text{def}}{=} h(x) + i^2 \mod 7$.

(a) Fügen Sie der Reihe nach die Elemente 3, 8, 2, 1, 4, 7, 6 in dieser Reihenfolge in ein gehashtes Array der Größe 11 ein. Notieren Sie in der unteren Tabelle die Position, in der das jeweilige Element gespeichert wird.

Index:	0	1	2	3	4	5	6	
Einfügen der 3 :			X					✓
Einfügen der 8 :				X				✓
Einfügen der 2 :							X	✓
Einfügen der 1 :				X				Kollision! Anderen!
Einfügen der 4 :						X		✓
Einfügen der 7 :	X							✓
Einfügen der 6 :					X			✓

4/8

(b) Zeichnen Sie eine Grafik, in der die Situation nach Einfügen der selben Zahlen in der selben Reihenfolge in ein gehashtes Array mit der selben Hashfunktion h dargestellt wird, wobei zur Kollisionsbehandlung Überlauflisten verwendet werden.



4/8

8. (8 Punkte)

Zeigen Sie: Sind $A, B \in \mathrm{P}$, so ist auch $A \cup B$ in P.

f

0/6

6.

(20 Punkte)

(a) Geben Sie einen Algorithmus an der als Eingabe eines Arrays a von Zahlen in Linearzeit das unten beschriebene leistet. Sei dazu x die erste Zahl des Arrays (also `a[0]`$=x$), bevor der Algorithmus beginnt zu arbeiten. Der Algorithmus findet einen geeigneten Index i im Array, und sortiert die Elemente in a so um, so dass alle der folgenden Bedingungen erfüllt sind:

i. `a[i]`$=x$

ii. Ist y ein Element des Arrays und $y \leq x$, so steht y vor x im Array.

iii. Ist y ein Element des Arrays und $y > x$, so steht y nach x im Array.

(b) Gegeben sei ein Array a mit dem Inhalt $(4, 5, 2, 7, 5, 1, 0)$ in dieser Reihenfolge. Vollziehen Sie Ihren Algorithmus auf Eingabe a nach und zeichnen Sie die Veränderungen in a im Verlauf des Algorithmus auf (beispielsweise wie in Aufgabe 3, oder durch eine Grafik), so dass man schrittweise nachvollziehen kann, welche Veränderungen im Array Ihr Algorithmus bewirkt. Geben Sie am Ende das Ergebnis Ihres Algorithmus auf Eingabe a an.

a.

```
public int[] a(int[] a) {
    int index = 0;
    for(int i = 1; i < a.length; i++) {
        if(a[i] <= a[index]) {
            int temp = a[i];
            a[i] = a[index];
            a[index] = temp;
            index = i;
        }
        if(a[i] > a[index]) {
            int count = a.length;
            while(count > index) {
                int temp = a[i];
                a[
                count--;
            }
        }
    }
}
```

Hier wollten Sie hoffentlich a[i] nach hinten verschieben. Dazu hätten Sie mit Zeitzeiger einen Zeiger auf die letzte Stelle des Arrays gesetzt. count würde außerhalb des if-Blocks vergessen

Ein guter Ansatz ist erkennbar. 7/14

b) ✗ 0/6

5. (12 Punkte)

Gegeben sei die folgende Liste mit Aussagen. Bitte kreuzen Sie an, welche der Aussagen wahr sind, und welche falsch.

Beachten Sie: Für jede richtig angekreuzte Antwort erhalten Sie einen Punkt, für jede falsch angekreuzte Anwort bekommen Sie einen Punkt abgezogen. Insgesamt können Sie auf diese Aufgabe aber nicht weniger als 0 Punkte bekommen.

wahr	falsch	Aussage
☒	☐	Ist f die Worst-Case Laufzeit eines Algorithmus A und g seine Average-Case Laufzeit, so gilt stets $g \leq_{ae} f$.
☐	☐	Ist $c \in \mathbb{R}$ so ist $c \cdot n^2 \neq O(n^2)$
☐	☐	Ist A eine NP-vollständige Menge, so lässt sich jedes $B \in$ NP auf A reduzieren.
☒	☐	P $\subseteq$ NP
☐	☒	$O(n^2) = O(n^3)$
☐	☐	Ist A eine NP-vollständige Menge, so gilt für jedes $B \in$ NP: $B \leq A$.
☐	☐	Ist $k \in \mathbb{N}$, so ist $\pi^k \cdot n = O(n)$.
☐	☐	$O(\log_\pi n) = O(\log_e n)$
☐	☐	$O(\sqrt{n}) = O(\log n)$
☐	☐	Sind f, g Funktionen mit $f \leq_{ae} g$, so ist $g \in O(f)$.
☐	☐	Für alle f, für die $f(n) = O(n^3)$ gilt, gilt auch $f(n) = O(n^2)$.
☐	☐	Für jede Funktion f gilt $f(n) = O(2^{2^n})$.

3/10

4.

(12 Punkte)

(a) Bestimmen Sie die asymptotische Laufzeit des untigen Algorithmus bei Aufruf der Funktion h im schlechtesten Fall (Sie können dabei die O-Notation verwenden). Bestimmen Sie dazu die asymptotische Laufzeit aller unten aufgeführten Anweisungen, Blöcke und Methoden im schlechtesten Fall. Notieren Sie die Laufzeit jeweils als Kommentar am Code.

(b) Bestimmen Sie die durch h berechnete Funktion.

```
// Laufzeit: 7 Takte
public static int f(int x) {
    if (x == 0) return 1;
    if (x == 1) return 1;
    return f(x-1) + f(x-2);
}
```

Warum? Könnte auch $x = 10^{10}$ sein

// Im schlechtesten Fall ist (x=2), wodurch beide if-Abfragen durchlaufen werden.

Rekursion?

```
// Laufzeit: 9n + 8 Takte
public static int g(int x) {
    int[] a = new int[x+1];
    if (x == 0) return 1;
    a[0] = 1;
    a[1] = 1;
    for (int i=2; i<=x; i++) {
        a[i] = a[i-1] + a[i-2];
    }
    return a[x];
}
```

// Im schlechtesten Fall ist x nicht 0. stimmt

Schleifenlaufzeit?

Laufzeiten der Methoden?

wobei?

```
// Laufzeit: 3 Takte
public static int h(int x) {
    return f(x) - g(x);
}
```

Aufruf der Funktionen, bzw. Berechnung derselben ist unsinnig!

0/10

b) + 0/2

1. (6 Punkte)

Geben Sie eine genaue Definition der Klasse P an.

✗ 0/6

2. (18 Punkte)

Gegeben sei der Graph $G = (V, E)$ mit

- $V = \{1, 2, 3, 4, 5, 6, 7\}$
- $E = \{\{1,3\}, \{4,6\}, \{3,4\}, \{5,6\}, \{2,7\}, \{1,7\}, \{2,5\}\}$

(a) Handelt es sich bei G um einen gerichteten Graphen?

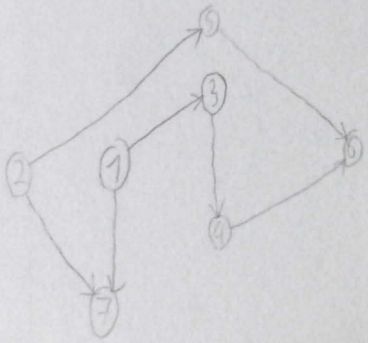
(b) Geben Sie die Adjazenzmatrix von G an.

(c) Stellen Sie G mit Hilfe von Adjazenzlisten dar.

(d) Geben Sie eine mögliche Reihenfolge an, in der die Knoten bei einer Breitensuche in die dabei verwendete Queue eingefügt würden, vorausgesetzt man startet die Suche bei Knoten 4.

(e) Geben Sie eine mögliche Reihenfolge an, in der die Knoten bei einer Tiefensuche besucht würden, wenn als erstes Knoten 4 besucht wird.

a. Ja, es handelt sich um einen gerichteten Graphen f

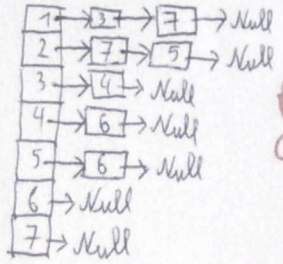


b.

	1	2	3	4	5	6	7
1	0	0	1	0	0	0	1
2	0	0	0	0	1	0	1
3	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

f (v) 3/3

c.



f (v) 3/3

d. ✗ 0/4

e. 4 → 6 → 1 → 3 → 1 → 7 → 2 → 5 → 2 f

3/4

3.

(8 Punkte)

Sie sehen unten die Darstellung eines unsortierten Arrays von Zahlen. Sortieren Sie dieses Array aufsteigend (das Endergebnis ist also 1,3,5,7,8,9,11) mit Hilfe des Heapsort-Algorithmus. Halten Sie dabei jede Änderung, die im Array vorgenommen wird (z.B. wenn zwei Zahlen vertauscht werden) in einem eigenen Zwischenschritt fest.

0	1	2	3	4	5	6
7	8	3	1	11	5	9

0/8