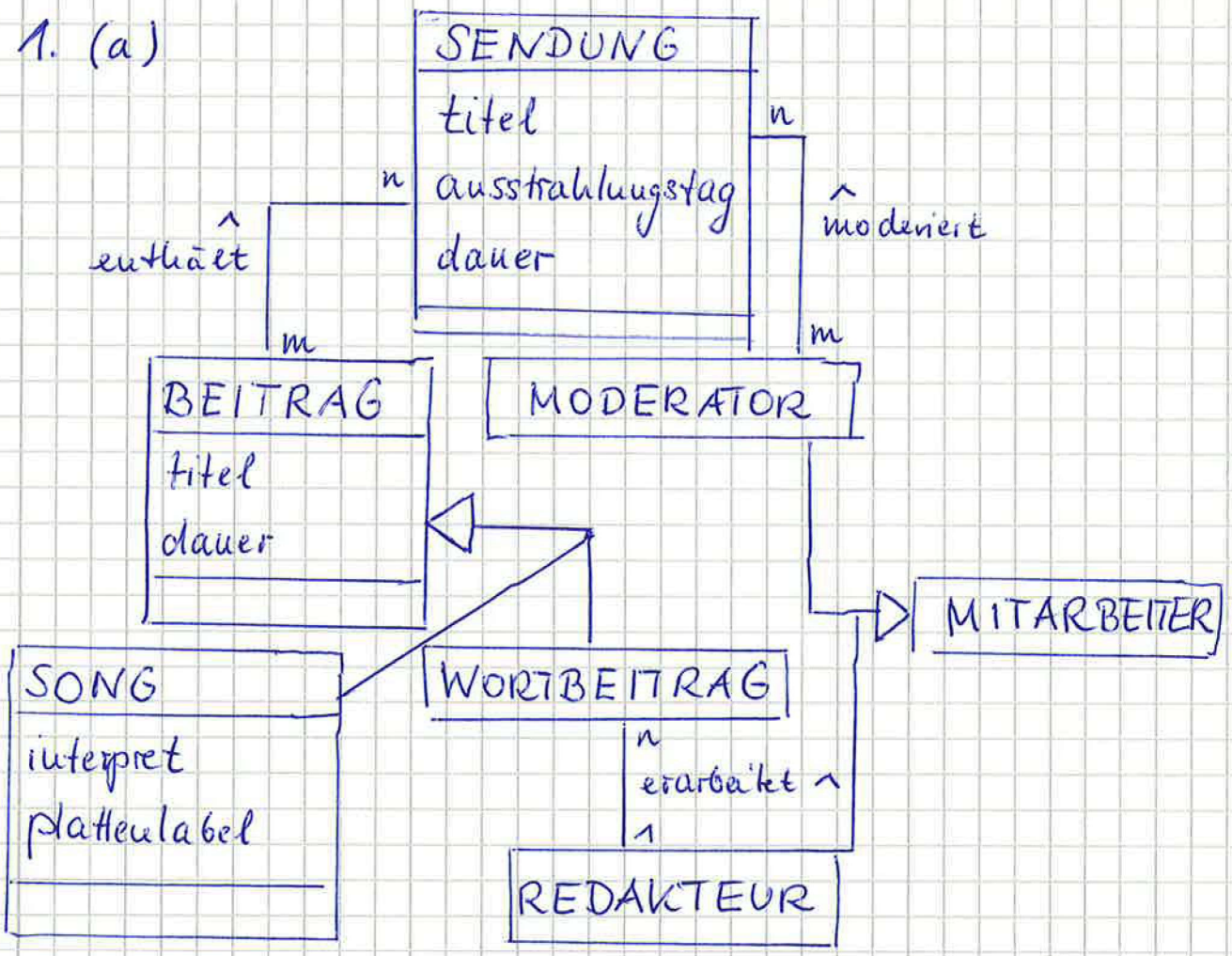
Krisch

Informatik Abitur 2022

INF 1. Modellierung und Programmierung I.

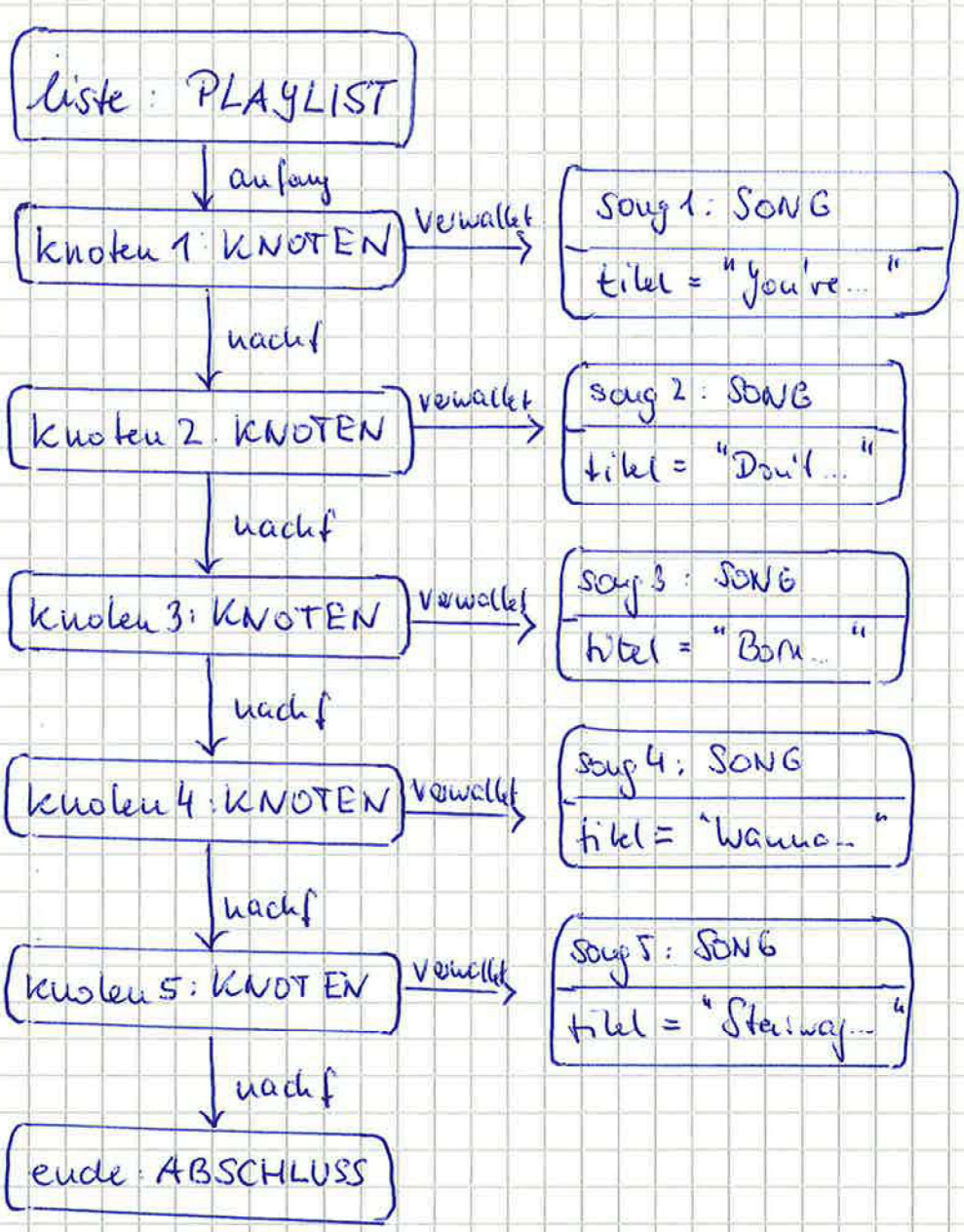
10 1. (a)



4 (b) (i) Die Häufigkeit, wie oft ein bestimmter Spot abgespielt wurde, soll bestimmt werden.

(ii) Der Anteil der Werbung in einer bestimmten Sendung soll ermittelt werden.

(c)



(d) in PLAYLIST:

```
void einfuegenNach(String titelVorgaenger,
                   String titelNeu) {
    anfang.einfuegenNach(titelVorgaenger, titelNeu);
}
```

in LISTENELEMENT

```
abstract void einfuegenNach(String titelVorgaenger,
                            String titelNeu);
```

in KNOTEN:

```
KNOTEN (SONG s, LISTENELEMENT l) {
        verwaltet = s;
        nachfolger = l;
}

void einfuegenNach (String titelVorgaenger, String titelNeu) {
    if (verwaltet.istGleich(titelVorgaenger)) {
            KNOTEN k = new KNOTEN (new SONG (titelNeu),
                                   nachfolger);
            nachfolger = k;
    } else {
        nachfolger.einfuegenNach (titelVorgaenger, titelNeu);
    }
}
```

in ABSCHLUSS:

```
void einfuegenNach (String titelVorgaenger, String titelNeu) {
    System.err.println("Vorgänger nicht vorhanden!");
}
```

(e) Da Stapel nach LIFO arbeiten, werden fruhzeitig abgegebene Songwünsche benachteiligt.

3

(f) Stapel: vorne Einfügen und Entnehmen
Liste: hinten Einfügen und vorn Entnehmen

2

2. (a) Statt nachfolger nun rechte Nachfolger und linker Nachfolger

2

(b) (IA) $n = 0$ ein Abschluss (einziges Element des Baums)
$n + 1 = 1$ ✓

6

(IV) Baum mit n Knoten, $n+1$ Abschlüsse

(IS) neuen Knoten hinzufügen. ($n \to n+1$)

→ Entfernen eines Abschlusses

→ Hinzufügen Knoten mit zwei Abschlüssen

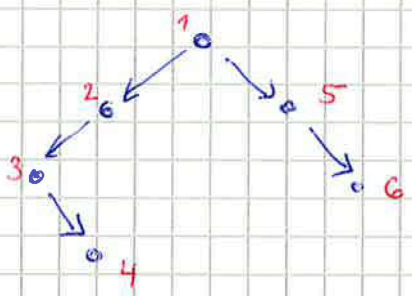
Nach IV nun $n + 1 - 1 + 2 = (n+1) + 1$ Abschlüsse ⇒ Behauptung gezeigt ✓

Speicherplatz bei 70 000 000 Knoten.

$70\,000\,001 \cdot 32\,\text{Byte} \approx 2{,}24\,\text{GB}$

(c) Preorder / Tiefensuche:

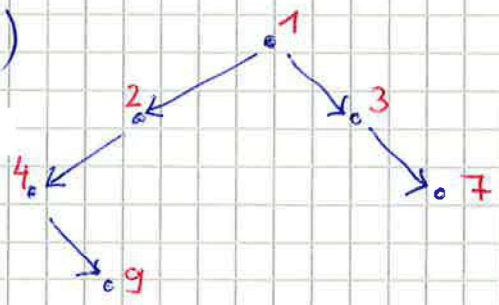
5



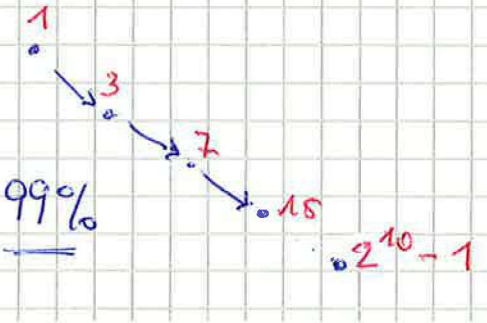
Rückführung möglich beginnend mit erstem Eintrag

(d)

6



(e) „Schlimmste" Struktur:



5

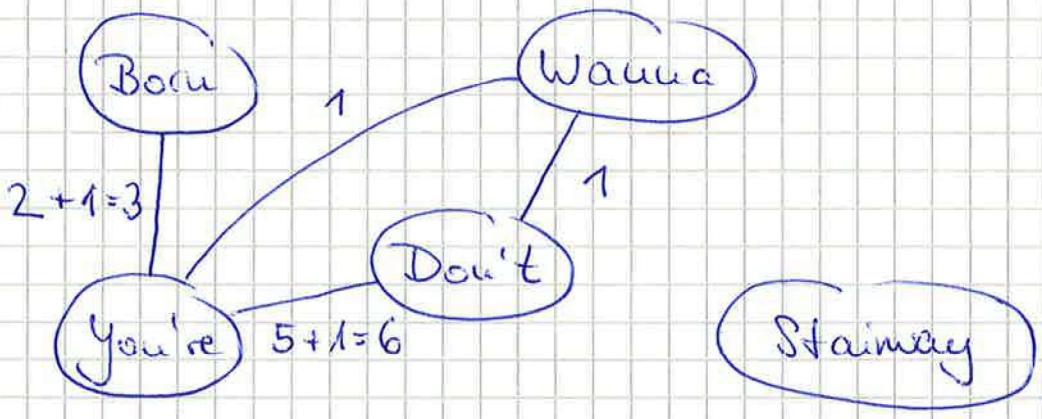
unbelegt: $\frac{2^{10} - 10}{2^{10}} \approx 99\%$

4. (a) gewichtet ~> Häufigkeit der Abfolge als Kantengewicht

4

ungewichtet ~> keine Rolle spielt, welcher Bug zuerst gelöst wurde

(b)



6

	Bom	Don't	Stairway	Wanna	You're
Bom	—	—	—	—	3
Don't	—	—	—	1	6
Stairway	—	—	—	—	—
Wanna	—	1	—	—	1
You're	3	6	—	1	—

(c)

```
void favListeAusgeben (String titel) {
    int i = knotenIndexGeben (titel);
    if (i < 0) {
        System.err.println ("Titel ex. nicht!");
    } else {
        favListeAusgeben (i);
    }
}
```

12

```
void favListeAusgeben (int i) {
    System.out.println (knoten[i].getInhalt().
            getTitel() + " ");
    knoten[i].setBesucht(true);
    int max = 0;
    int next = -1;
    for (int k = 0; k < anzahlKnoten; k++) {
        if (matrix[i][k] > max && !knoten[k].
                getBesucht()) {
            max = matrix[i][k];
            next = k;
        }
    }
    if (max > 0) {favListeAusgeben(next);}
}
```

Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.