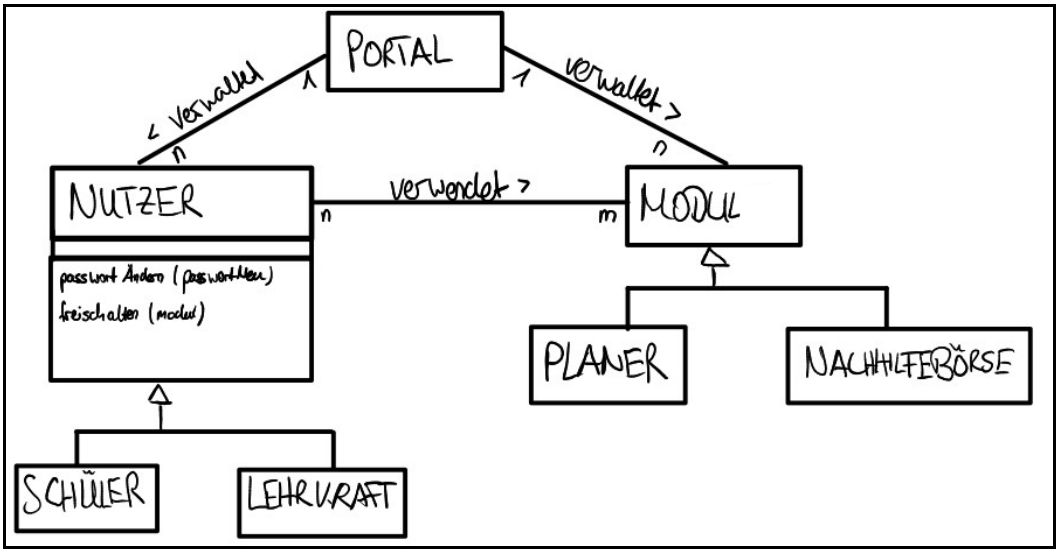
Informatik Abitur Bayern 2023 / I - Lösung

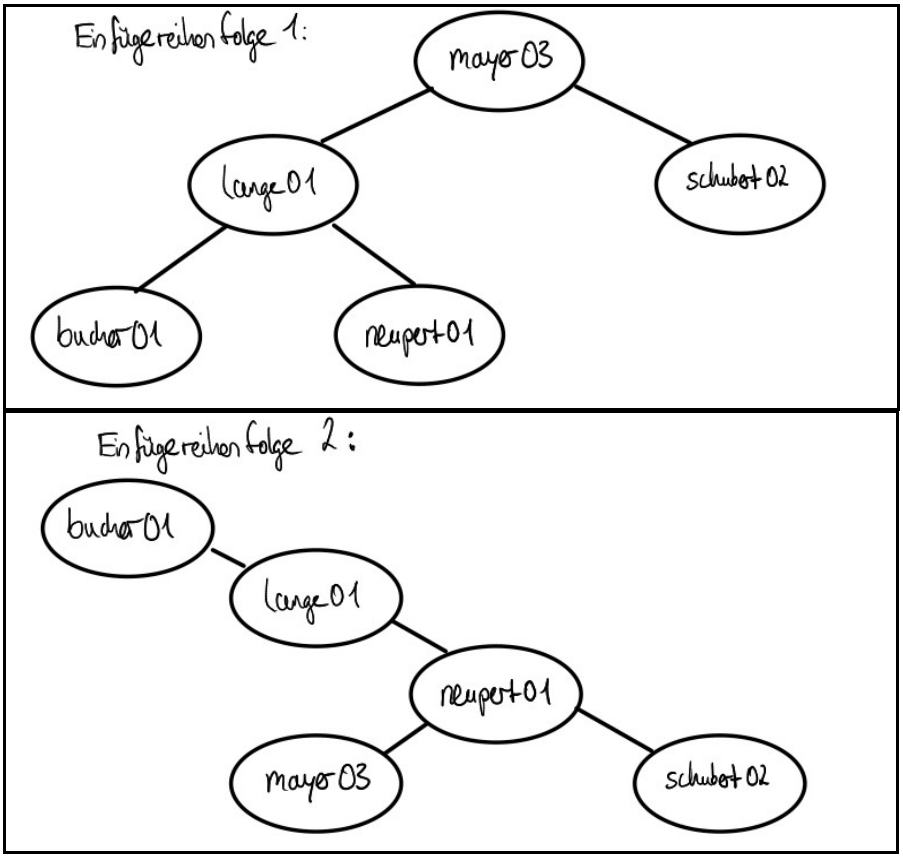
Autor:
Leesch(1-2)
Kokula (3)
Scholz (4)

1



6

2a



5

Die Einfügereihenfolge legt die Anzahl der Ebenen im Binärbaum fest. Je mehr Ebenen der Binärbaum besitzt, desto länger benötigt die Suche.

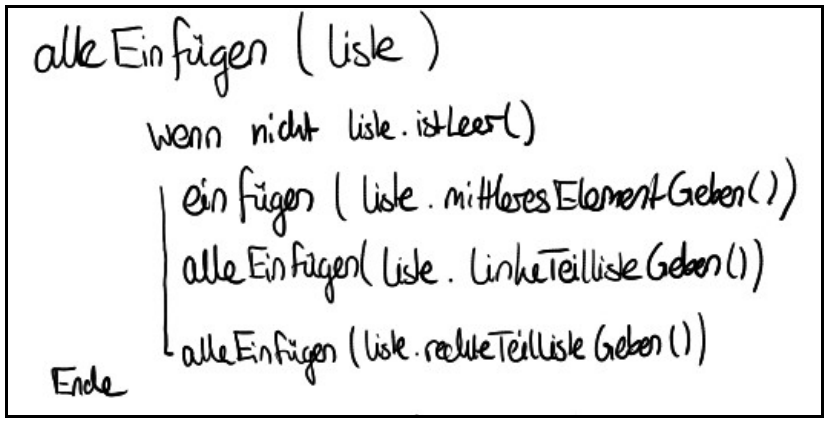
2b Ein Binärbaum mit 600 Einträgen hat mindestens $\left(\log_2 600\right)+1=10$ $Aufrunden\left(\left(\log_2 600+1\right)\right)=10$ Ebenen und somit insgesamt $2^{10}-1=1023$ Plätze. Im Idealfall bzw. Best-Case stehen $1023-600=423$ freie Plätze zur Verfügung. In diesem Fall muss keine Ebene hinzugefügt werden.
Im Worst-Case müssen 220 Ebenen hinzugefügt werden, wenn alle einzufügenden Objekte als Liste an ein Blatt maximaler Tiefe angehängt werden.

4

2c Dieses Vorgehen ist nicht optimal, da so der linke und rechte Teilbaum einen zur Liste entarteten Baum darstellt. 4

Grafiken fehlen

2d Rekursive Methode: 6



2e Nach diesem Schema können Loginnamen fremder Schüler*innen rekonstruiert werden. Damit und mit dem Standardpasswort kann sich jeder Nutzer mit den Zugangsdaten einer anderen Person einloggen, bevor sich diese Person das erste Mal eingeloggt hat. Aus der Sicht des Datenschutzes und -sicherheit ist das Verfahren deshalb ungeeignet. 2

2f

```
Class BAUM() {
    ...
    void nutzerMitStandardpasswortAusgeben() {
        Wurzel.nutzerMitStandardpasswortAusgeben();
    }
    ...
}
abstract class BAUMELEMENT() {
    ...
    abstract void nutzerMitStandardpasswortAusgeben();
    ...
}
class ABSCHLUSS() {
    ...
    void nutzerMitStandardpasswortAusgeben() {}
    ...
}
class KNOTEN() {
    ...
    void nutzerMitStandardpasswortAusgeben() {
        linkerNachfolger.nutzerMitStandardpasswortAusgeben();
        if (inhalt.passwortGeben().equals("GH7uu4")) {
            inhalt.ausgeben();
        }
        rechterNachfolger.nutzerMitStandardpasswortAusgeben();
    }
    ...
}
```

8

3a

10

in Klasse ANMELDELISTE:

```
public void ausgebenUndLöschen(String ziel)
{
      anfang = anfang.ausgebenUndLöschen(ziel);
}
```

in Klasse LISTENELEMENT:

```
public abstract Listenelement ausgebenUndLöschen(String ziel);
```

in Klasse ABSCHLUSS:

```
public Listenelement ausgebenUndLöschen(String ziel)
{
      return this;
}
```

```
in Klasse KNOTEN:

public Listenelement ausgebenUndLöschen(String ziel)
{
	nachfolger = nachfolger.ausgebenUndLöschen(ziel);
	if (inhalt.istErstwunschGleich(ziel) ||
inhalt.istZweiwunschGleich(ziel))
	{
		inhalt.ausgeben();
		return nachfolger;
	}
	else
	{
		return this;
	}
}
```

3b → Rom: 2 freie Plätze 4
→ Paris: 1 freier Platz
→ Berlin: 0 freie Plätze

Albert Tross: Zuteilung zu Rom
→ Rom: 1 freier Platz
→ Paris: 1 freier Platz
→ Berlin: 0 freie Plätze

Thor Schluss: Zuteilung zu Paris
→ Rom: 1 freier Platz
→ Paris: 0 freie Plätze
→ Berlin: 0 freie Plätze

Martha Pfahl: Zuteilung Paris schlägt fehl → Zuteilung Rom
→ Rom: 0 freie Plätze
→ Paris: 0 freie Plätze
→ Berlin: 0 freie Plätze

Dennis Schläger: Zuteilung Rom schlägt fehl → Zuteilung Berlin schlägt fehl → Fehler!

3c Der Algorithmus bevorzugt Anmeldungen, die weiter oben in der Liste angegeben sind. Z.B. erhalten die ersten x Personen in der Liste auf jeden Fall ihren Erstwunsch, wobei x die geringste Anzahl an Plätzen für eine Reise ist. 2
Dies kann man dadurch beeinflussen, dass abwechselnd das erste und das letzte Listenelement eine Zuteilung erhalten.

3d 9

```
in Klasse ZUTEILUNGSLISTE:

public double anteilErfüllteErstwünsche()
{
	int anzahlErfüllt = 0;
	return anzahlErfüllteErstwünsche(anzahlErfüllt)/längeGeben();
}

public double anzahlErfüllteErstwünsche(int anzahlBisher)
{
	anfang = anfang.anteilErfüllteErstwünsche(ziel, anzahlBisher);
}


in Klasse LISTENELEMENT:

public abstract Listenelement anzahlErfüllteErstwünsche(String ziel, int
```

```
anzahlBisher);


in Klasse ABSCHLUSS:

public Listenelement anzahlErfüllteErstwünsche(String ziel, int
AnzahlBisher)
{
	return 0;
}


in Klasse KNOTEN:

public Listenelement anzahlErfüllteErstwünsche(String ziel, int
anzahlBisher)
{
	nachfolger = nachfolger.listeErfüllteErstwünsche(AnzahlBisher);
	if (inhalt.istErstwunschGleich(ziel)
	{
		anzahlBisher++;
		return nachfolger;
	}
	else
	{
		return this;
	}
}
```

4a

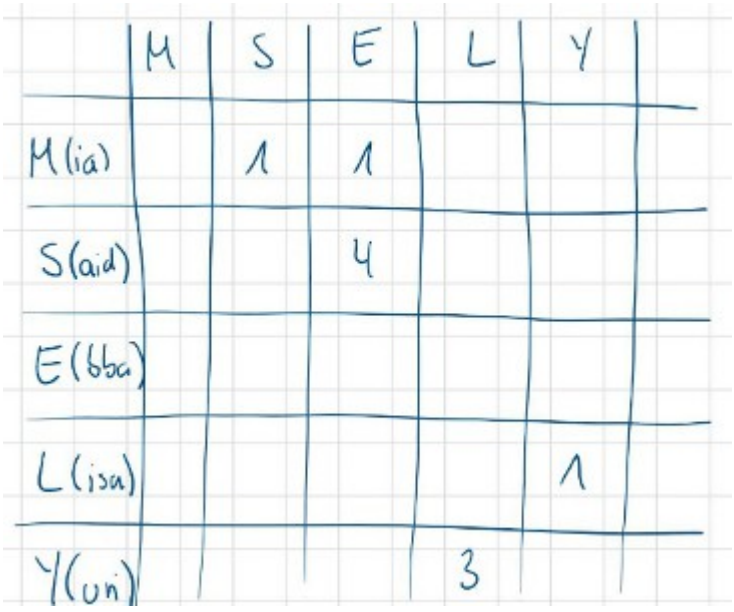
1. Mia gibt Said in Mathematik Nachhilfe.
2. Mia gibt Ebba in Mathematik Nachhilfe.
3. Said gibt Ebba in Englisch Nachhilfe.

4

Ein solcher Algorithmus wäre nicht geeignet, denn...

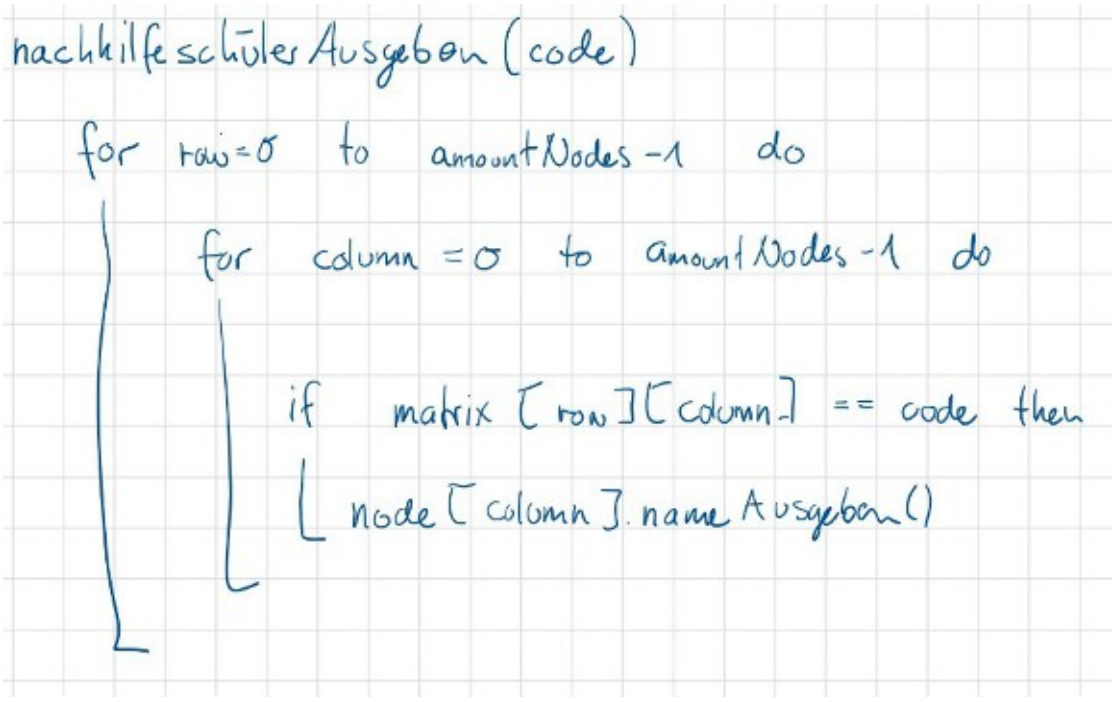
1. der Teilgraph mit Mia ist nicht vom Teilgraph mit Lisa aus erreichbar
2. Mia ist nicht von Ebba aus erreichbar, da kein gerichteter Pfad von Ebba zu Mia existiert.

4b



4

4c



5

4d Wenn z. B. ein Nachhilfeschüler in zwei unterschiedlichen Fächern von derselben Person Nachhilfe bekommt, kann dies nicht im Graphen abgebildet werden.
Durch die zusätzliche Klasse NACHHILFEART, welche als Attribute Referenzen auf den Nachhilfelehrer als auch auf den Nachhilfeschüler hat, kann das Problem gelöst werden. Ein weiteres Attribut Fach ermöglicht schlussendlich mehrere Beziehungen zwischen Nachhilfelehrer und Nachhilfeschüler in verschiedenen Fächern.

7

80