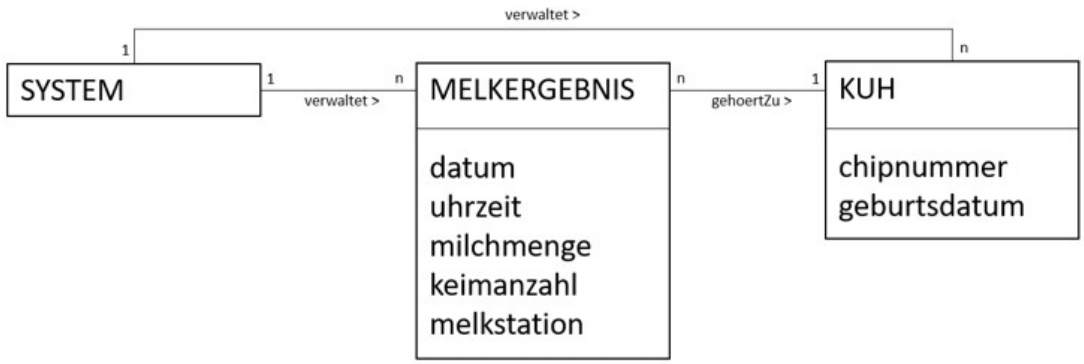
# Informatik Abitur Bayern 2024/I - Beispiellösung

Autor:
Kuntscher(1)
Zettler(2,3)

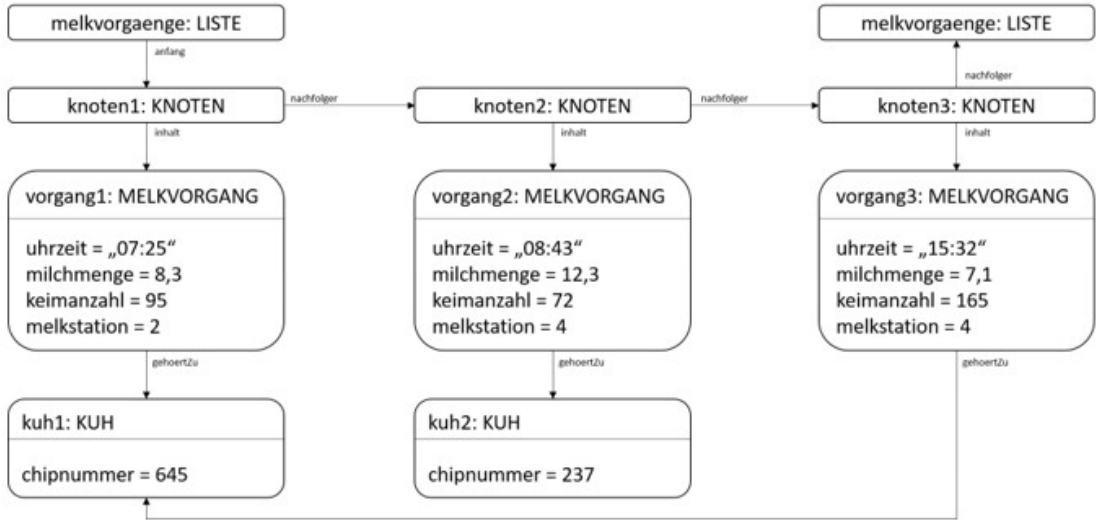
1a 5




b Vorteil: Die Anzahl der Melkvorgänge in der Liste ist nicht begrenzt. 2

Nachteil: Der Zugriff auf ein bestimmtes Melkergebnis ist aufwändig, da immer die Liste durchlaufen werden muss.

c 6




d 16

Klasse LISTE

```
double durchschnittlicheKeimanzahlBerechnen(int nr) {
    return anfang.keimanzahlSummieren(nr) /
anfang.melkvorgaengeZaehlen(nr);
}

void melkergebnisseLoeschen(int chipnr) {
    anfang = anfang.melkergebnisseLoeschen(chipnr);
}
```

Klasse LISTENELEMENT

```
abstract double keimanzahlSummieren(int nr);
abstract double melkvorgaengeZaehlen(int nr);
abstract double melkergebnisseLoeschen(int chipnr);
```

Klasse KNOTEN

```
double keimanzahlSummieren(int nr) {
    if (inhalt.gibMelkstation() == nr) {
```

```
        return inhalt.gibKeimanzahl() + nachfolger.keimanzahlSummieren(nr);
    } else {
        return nachfolger.keimanzahlSummieren(nr);
    }
}

int melkvorgaengeZaehlen(int nr) {
    if (inhalt.gibMelkstation() == nr) {
        return 1 + nachfolger.melkvorgaengeZaehlen(nr);
    } else {
        return nachfolger.melkvorgaengeZaehlen(nr);
    }
}

Listenelement melkergebnisseLoeschen(int chipnr) {
    nachfolger = nachfolger.melkergebnisseLoeschen(chipnr);
    if (inhalt.gibKuh().gibChiphnr() == chipnr) {
        return nachfolger;
    } else {
        return this;
    }
}
```

Klasse ABSCHLUSS

```
double keimanzahlSummieren(int nr) {
    return 0.0;
}

int melkvorgaengeZaehlen(int nr) {
    return 0
}

Listenelement melkergebnisseLoeschen(int chipnr) {
    return this;
}
```

e

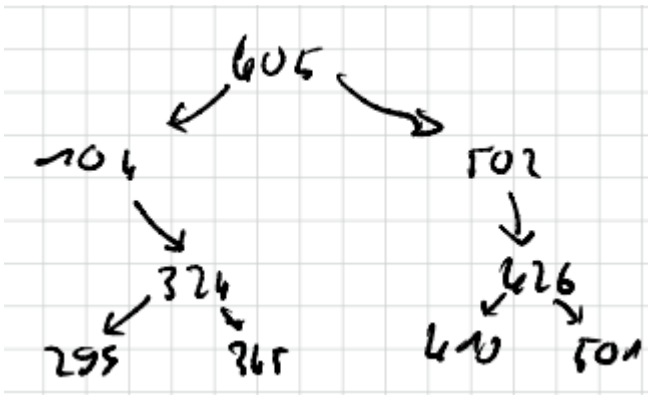
| m1(972) | m1(241) |
|---|---|
| m2(972,0,0) | m2(241,0,0) |
| m2(972,0,0) | m2(241,0,0) |
| m2(972,0,0) | m2(241,0,0) |
| m2(972,0,0) | m2(241,1,1) |
| m2(972,1,1) | m2(241,1,1) |
| m2(972,1,1) | m2(241,1,0) |
| m2(972,1,1) | m2(241,1,1) |
| m2(972,1,1) | m2(241,2,2) |
| m2(972,1,0) | m2(241,2,2) |
| m2(972,1,0) | m2(241,3,3) |
| m2(972,1,1) | m2(241,3,3) |
| m2(972,1,1) | m2(241,3,0) |
| Rückgabewert: 1 | Rückgabewert: 3 |

9

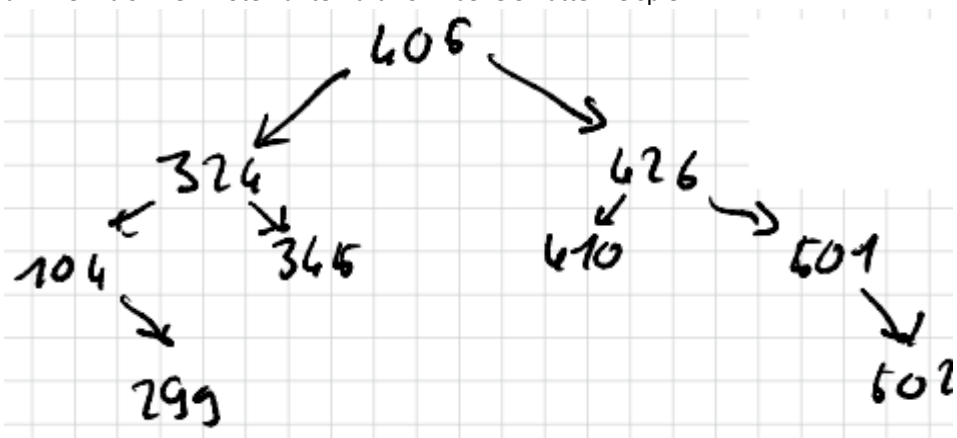
Methode m1 ermittelt durch rekursiven Aufruf von Methode m2 für alle Listenelemente einer Bestimmten Kuh die maximale Anzahl aufeinanderfolgender Melkvorgänge mit erhöhter Keimbelastung (>100).

Parameter p2 enthält dabei die aktuelle Anzahl aufeinanderfolgender Melkvorgänge mit erhöhter Keimbelastung. Parameter p1 enthält den Maximalwert von p1.

2a 3




b Ein optimaler Baum hätte möglichst wenige Knoten auf unterster Ebene, wie etwa der folgende: 6
Als Wurzel kommen die Chipnummern 345, 405, 410 in Frage, da ansonsten der linke oder der rechte Teilbaum mehr als zwei Knoten unterhalb von Ebene 3 hätte. Beispiel:




Mit 345 als Wurzel ist der linke Teilbaum eindeutig festgelegt. Als Wurzel des rechten Teilbaums kommen 410, 426, 501 in Frage. Für 410 und 501 als Wurzel des rechten Teilbaums gibt es jeweils nur einen Fall. Mit 426 als Wurzel des rechten Teilbaums gibt es gibt es links und rechts davon jeweils zwei Möglichkeiten, insgesamt 2 * 2 = 4 Möglichkeiten. Mit 345 als Wurzel gibt es insgesamt also 6 Möglichkeiten.

Aus Symmetrieüberlegungen gibt es für 410 als Wurzel weitere 6 Möglichkeiten.

Mit 405 als Wurzel bestehen linker und rechter Teilbaum jeweils aus 4 Knoten. Dafür gibt es jeweils 4 Möglichkeiten mit nur einem Knoten auf unterster Ebene. Die Kombinationen aus linken und rechten Teilbäumen führen auf insgesamt 4 * 4 = 16 Möglichkeiten.

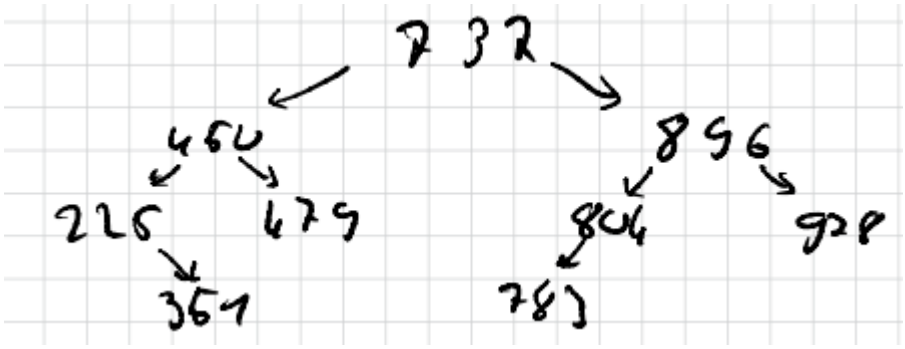
Insgesamt sind es also 16 + 6 + 6 = 28 Möglichkeiten.

c Ein Baum mit n Ebenen bieten 2^n – 1 Plätze 2
2^10 - 1 = 1023 → zu wenige Plätze; 2^11 - 1= 2047; der Baum braucht also minimal 11 Ebenen.

d 737 – 431 – 225 – 351 – 479 – 450 – 896 – 804 – 783 – 866 – 942 – 978 4

Aus der Preorder – Traversierung kann der Baum rekonstruiert werden. Es handelt sich also um eine Möglichkeit zur linearen Speicherung des Baumes.

e 8



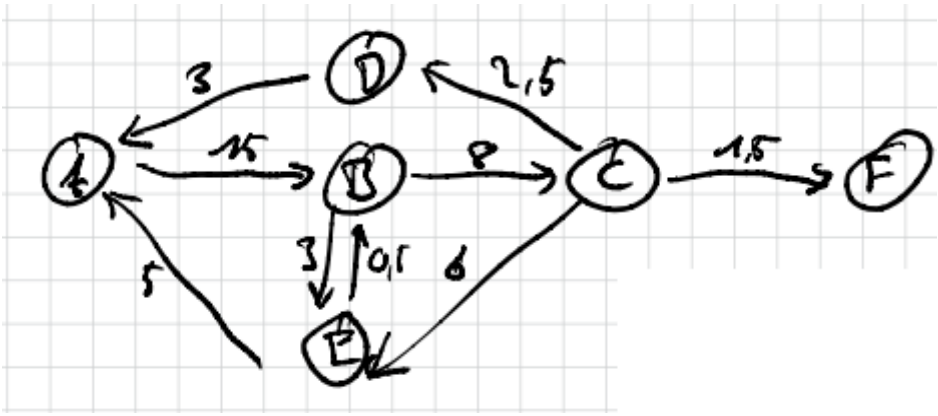

866 hat rechts einen Abschluss, gibt links zurück.

942 hat rechten Nachfolger, links ist Abschluss → gibt rechten NF zurück.

431 hat rechts und links keinen Abschluss, es wird *Löschen2* aufgerufen. Der Knoten wird durch den Knoten mit der kleinsten Chipnummer rechts ersetzt.

Allgemein arbeitet die Methode Löschen2 aber nicht korrekt; dies beeinträchtigt die Lösung der Aufgabe allerdings nicht!

3a 5




Der Graph ist gerichtet und gewichtet.

b 7

```
public void BilanzBerechnen()
{
    double bilanz;
    for (int i = 0; i < anzahl; i = i+1){
            bilanz = 0.0;
            for (int j = 0; j < anzahl; j = j + 1){
                bilanz = bilanz + matrix[i][j]+ matrix[j][i];
            }
            knoten[i].BilanzSetzen(bilanz);
     }
}
```

c 7

```
public void JahresabschlussAusgeben(){
    BilanzBerechnen();
    int flop, top;
    for (int i = 1; i < anzahl; i = i+1){
      if(knoten[i].BilanzGeben()<knoten[flop].BilanzGeben())
          flop = i;
      if(knoten[top].BilanzGeben()<knoten[i].BilanzGeben())
          top = i;
    }
    System.out.println(knoten[top].NameGeben()+“ wird eingeladen von “
 + knoten[flop].NameGeben());
}
```