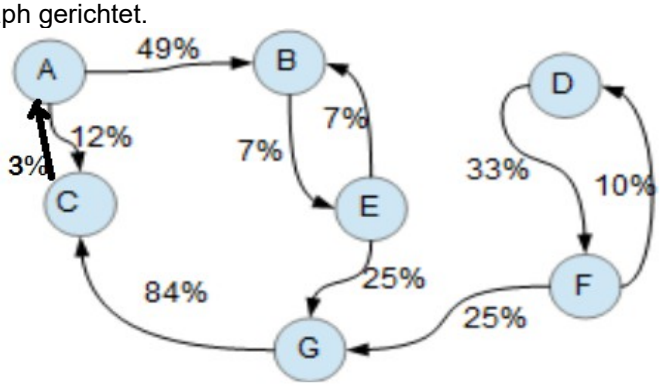
Informatik Abitur Bayern 2014 / I - Beispiellösung

Autor:
Reinold

1a Da die Besitzbeziehung gerichtet ist und keine Symmetrie bei den gegenseitigen Anteilen besteht, ist der Graph gerichtet. 6



1b 4

	A	B	C	D	E	F	G
A		49%	12%				
B					7%		
C	3%						
D						33%	
E		7%					25%
F				10%			25%
G			84%				

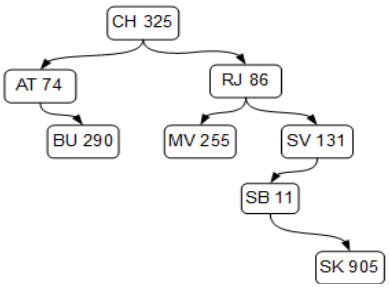
1c 10

```
boolean[] besucht;
besucht = new boolean[anzahlKnoten]
void Tiefensuche(String start)
{
   int startnummer = KnotennummerGeben(start);
   if (startnummer != -1)
   {
     for(int i = 0; i < anzahlKnoten; i++)
     {
       besucht[i] = false;
     }
     Besuchen(startnummer);
}

void Besuchen(int knotennummer)
{
   besucht[knotennummer] = true;
   //Ausgabe: "Besuche"+Knotenbezeicher[knotennummer]
   for(int i = 0; i < anzahlKnoten; i++)
   {
     if(adjazenzmatrix[knotennummer][i]>0 && !besucht[i])
     {
       Besuchen[i];
     }
   }
   //Ausgabe „fertig mit"+Knotenbezeicher[knotennummer]
}
```

Wenn man davon ausgeht, dass die Knoten in alphabetischer Reihenfolge im Feld abgelegt sind, so ergibt sich folgende Reihenfolge: A – B – E – G – C (alternativ: A – C [zurück zu A] –

B – E – G).
Das Ergebnis stellt die direkten und indirekten Anteile dar, die eine Firma an anderen Firmen besitzt.

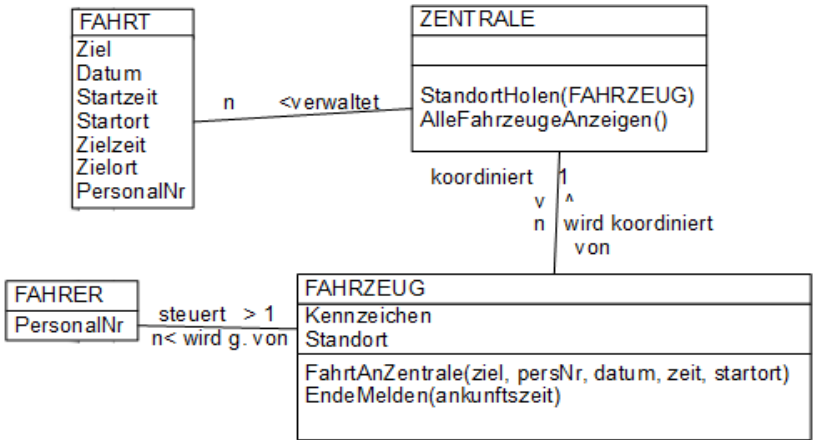
2a 

4

2b Reihenfolge: linker Teilbaum → rechter Teilbaum → Wurzel 3
BU 290 - AT 74 – MV 255 – SK 905 – SB 11 – SV 131 – RJ 86 – CH 325

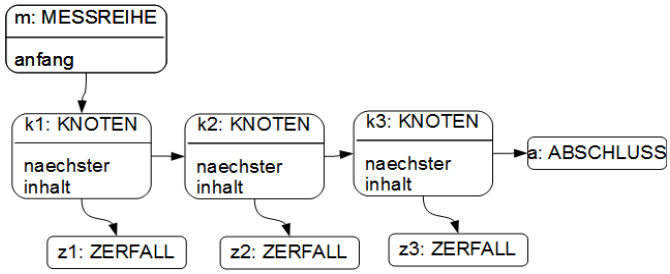
2c Da STA SV 131 nur einen Nachfolger hat, der lexikographisch größer sein muss als der Vorgänger von STA SV 131, kann STA RJ 86 ohne weitere Änderungen künftig STA SB 11 referenzieren. 2

2d Der Baum sollte im Idealfall ausgeglichen sein, d. h. die Tiefe aller Blätter variiert höchstens um 1. 6
In einem Baum mit n Ebenen befinden sich höchstens 2^n-1 Knoten.
$k \leq 2^n-1$
$\Leftrightarrow k + 1 \leq 2^n \mid ld$
$\Leftrightarrow ld(k+1) \leq n$
Mit k = 7.000.000 ergibt sich $n \geq 22{,}7$, der Baum benötigt also minimal 23 Ebenen.

3a 

16

3b Um ein umfangreiches Projekt erfolgreich durchzuführen, müssen gewisse Aufgaben erledigt sein, ehe die nächste Phase begonnen werden kann. 4
- Analyse
- Systemdesign
- Implementierung
- Test, Bewertung, Abnahme
- Installation, Wartung

4a 

4b

20

```
public class MESSREIHE
{
    LISTENELEMENT anfang;

    public MESSREIHE()
    {
        anfang = new ABSCHLUSS();
    }

    float maxEnergieGeben()
    {
        return anfang.maxEnergieGeben(0);
    }

    float mittlereZerfallszeitGeben()
    {
        return anfang.mittlereZerfallszeitGeben(0,0);
    }
}

public abstract class LISTENELEMENT
{
    abstract float maxEnergieGeben(float e);
    abstract float mittlereZerfallszeitGeben(int bisherige_anzahl,float bisheriger_mittelwert);
}

public class KNOTEN extends LISTENELEMENT
{
    ZERFALL inhalt;
    LISTENELEMENT naechster;

    public KNOTEN(ZERFALL inhaltNeu, LISTENELEMENT naechsterNeu)
    {
        inhalt = inhaltNeu;
        naechster = naechsterNeu;
    }

    float maxEnergieGeben(float e)
    {
        if(inhalt.energieGeben() >e)
        {
            e = inhalt.energieGeben();
        }
        return naechster.maxEnergieGeben(e);
    }

    float mittlereZerfallszeitGeben(int bisherigeAnzahl, float bisherigerMittelwert)
    {
        float neuerMittelwert;
       neuerMittelwert=(bisherigeAnzahl*bisherigerMittelwert+inhalt.zerfallszeitGeben())/(bisherigeAnzahl+1);
        return naechster.mittlereZerfallszeitGeben(bisherigeAnzahl+1, neuerMittelwert);
    }
}

public class ABSCHLUSS extends LISTENELEMENT
{
    float maxEnergieGeben(float e)
    {
        return e;
    }

    float mittlereZerfallszeitGeben(int bisherige_anzahl, float bisheriger_mittelwert)
    {
        return bisheriger_mittelwert;
    }
}

public class ZERFALL
{
    float zerfallszeit;
    float energie;
    int anzahlGammaquanten;

    public ZERFALL(float zerfallszeitNeu, float energieNeu, int anzahlGammaquantenNeu)
    {
        zerfallszeit = zerfallszeitNeu;
        energie = energieNeu;
        anzahlGammaquanten = anzahlGammaquantenNeu;
    }

    float zerfallszeitGeben()
    {
        return zerfallszeit;
    }

    float energieGeben()
    {
        return energie;
    }
}
```