Matthias Jauernig
Kurs: 03In1

20.04.04
linux-related.de

4. Übungsaufgaben zur LV
Algorithmen & Datenstrukturen

Abgabetermin: Do, 22.04.04

1.) Methode „*public int AnzStarkZshgKomp()*“ implementieren:

```
public int AnzStarkZshgKomp() {
/** Ermittelt die Anzahl der stark zusammenhängenden Komponenten. Die Knoten
 *  i einer Komponente werden mit getMarke(i)=w markiert,
 *  1 <= w <= AnzStarkZshgKomp().
 */
   initMarke(0);
   int w=0;
   for(int i=1;i<=n;i++){
     if(getMarke(i)==0){
       w++;
       DFS(i); //von i aus erreichbare Knoten ermitteln
       for(int j=1;j<=n;j++)
         if(getMarke(j)==0 && getStzBg(j)!=0)
           setMarke(j,w);
       DFSruek(i); //ist i auch von j aus erreichbar?
       for(int j=1;j<=n;j++)
         if(getMarke(j)==w && getStzBg(j)==0)
           setMarke(j,0);
     }
   }
   return w;
}
```

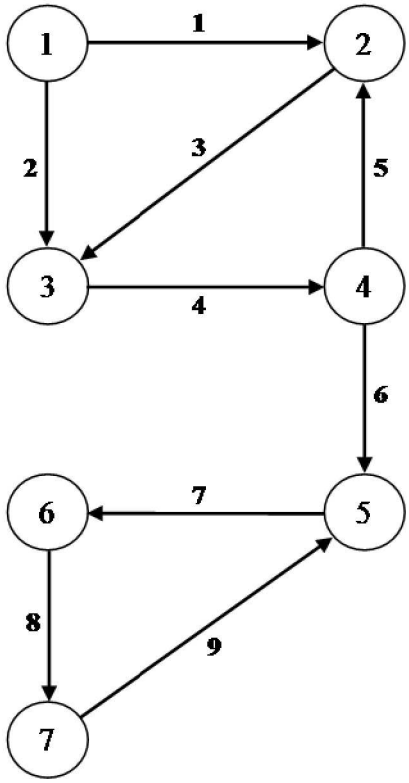
2.) Methode „*public String Analyse()*“ implementieren:

```
public String Analyse()
    /** Liefert Strings, mit Aussagen über Zusammenhang bzw. starken
     *  Zusammenhang.
     */
    {
      String str=new String();
      int nStark, nZus;
      for(int i=1;i<=n;i++){
        str=str+"Knoten "+i+": Marke="+getMarke(i);
        if(i%2!=0)  str+="\t";
        else  str+="\n";
      }
      str+="\n\n";
      nZus=AnzZshgKomp();
      nStark=AnzStarkZshgKomp();

      if(nZus==1 && nStark==1)
        str+="Der Graph besteht aus nur einer (stark) zusammenhängenden\n"+
             "Komponenten und bildet deshalb einen Kreis.\n";
      else if(nZus==nStark && nZus>1 && nZus<n)
        str+="Der Graph hat "+nZus+" zusammenhängende sowie stark\n"+
             "zusammenhängende Komponenten und ist nicht kreisfrei.\n";
      else if(nStark==n)
        str+="Der Graph hat "+nZus+" zusammenhängende Komponenten und\n"+
             "die Anzahl der stark zusammenhängenden Komponenten\n"+
             "entspricht der Anzahl der Knoten des Graphen ("+nStark+")\n"+
             "- demnach ist der Graph kreisfrei.\n";
```

```
    else if(nZus<nStark && nStark<n){
      str+="Anzahl der zusammenhängenden Komponenten ("+nZus+") kleiner\n"+
           "der Anzahl der stark zusammenhängenden Komponenten ("+nStark+")\n"+
           "kleiner n="+n+". ";
      //Kreise im Graphen zählen:
      int ElemKomp[]=new int[nStark];
      int kreise=nStark;
      for(int i=1;i<=n;i++)  ElemKomp[getMarke(i)-1]++;
      for(int i=0;i<nStark;i++)
        if(ElemKomp[i]==1)
          kreise--;
      if(kreise==0)  str+="Der Graph ist kreisfrei.\n";
      else  str+="Der Graph besitzt "+kreise+" Kreise!\n";
    }
    return str;
  }
```

Zum Testen der beiden Methoden habe ich mir folgenden Graphen ausgedacht (andere Tests seien hier aus Platzgründen nicht angefügt):



Rahmenprogramm (auszugsweise):

```
package adsuebungen;
import ADS.*;
//...
public class Dialog_Graph2 extends JDialog {
  JTextArea ta = new JTextArea();
  //...
  private void jbInit() throws Exception {
    //...
    //Testgraph aufbauen
    Graph G1=new Graph(7,9);
    G1.insertBg("1","2");
    G1.insertBg("1","3");
    G1.insertBg("2","3");
    G1.insertBg("3","4");
    G1.insertBg("4","2");
    G1.insertBg("4","5");
    G1.insertBg("5","6");
    G1.insertBg("6","7");
    G1.insertBg("7","5");
    //Tests darauf ausführen
    ta.append("Analyse des Testgraphen\n\n\n");
    ta.append("Anzahl stark zusammenhängender Komponenten: "+
              G1.AnzStarkZshgKomp()+"\n\n");
    ta.append("Informationen über den Graph:\n");
    ta.append(G1.Analyse());
  }
//...
}
```

Ausgabe des Tesprogramms: