LV Algorithmen und Komplexität **WS 2013/14**

Aufgaben für das Thema: *Graphenrepräsentationen*

Autoren: Julian Hilsberg, Hannes Kretschmer

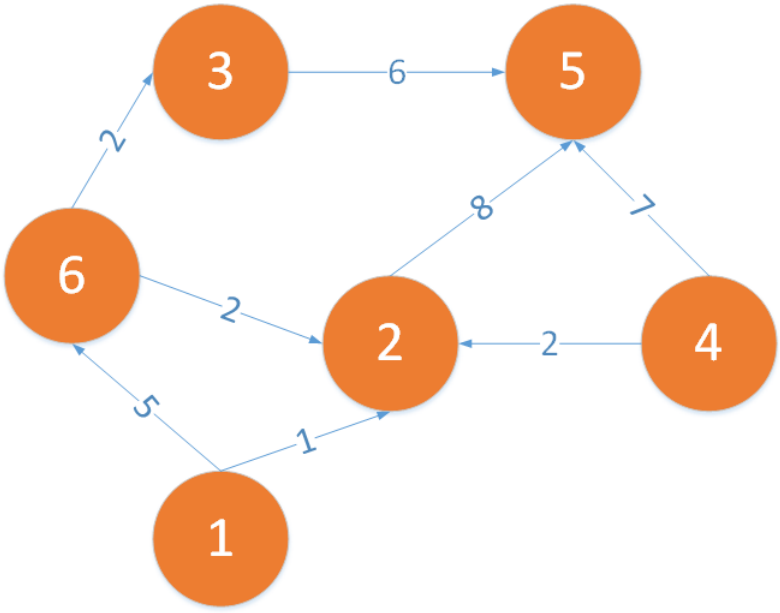
1 Adjazenzmatrix

Rekonstruiere den ungerichteten Graphen aus der gegebenen Adjazenzmatrix.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

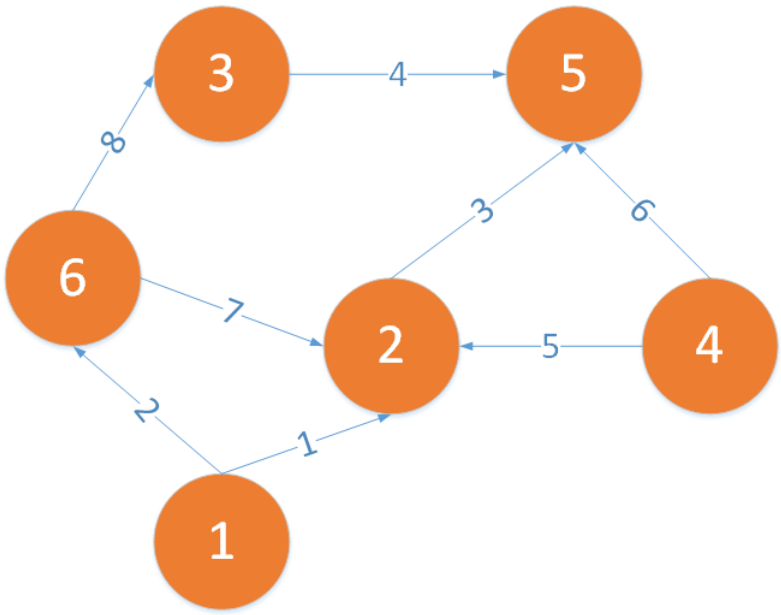
2 Bewerter Graph

Bilde die bewertete Adjazenzmatrix für den gegebenen Graphen.



3 Inzidenzmatrix

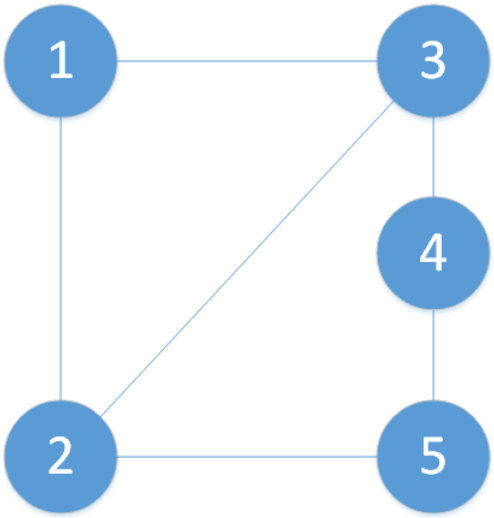
Bilde die Inzidenzmatrix für den gegebenen Graphen. Kanten sind nummeriert, nicht bewertet!



4 Implementierung

Implementiere eine Adjazenzmatrix in einer Programmiersprache deiner Wahl. Füge Kanten ein und gib die Matrix vor und nach dem Einfügen der Kanten in der Konsole aus.

1 Lösung zu Aufgabe 1



2 Lösung zu Aufgabe 2

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 2 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 & 0 & 0 \end{pmatrix}$$

3 Lösung zu Aufgabe 3

$$A = \begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & v_1 \\ -1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & v_2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & v_3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & v_4 \\ 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & v_5 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & v_6 \end{pmatrix}$$

4 Lösung zu Aufgabe 4

Listing 1: Klassenimplementierung der Adjazenzmatrix in Java

```
public class AdjMatrix {

        private boolean[][] adjazenzMatrix = null;

        private int kantenZahl = 0;

        public AdjMatrix(int knotenZahl) {

                this.adjazenzMatrix = new boolean[knotenZahl][knotenZahl];
        }

        public int getKantenZahl() {

                return this.kantenZahl;
        }

        public int getKnotenZahl() {

                return this.adjazenzMatrix.length;
        }

        public void kanteEinfuegen(int quellKnoten, int zielKnoten) {

                if (quellKnoten >= this.adjazenzMatrix.length
                                || zielKnoten >= this.adjazenzMatrix.length)
                        return;

                if (this.adjazenzMatrix[quellKnoten][zielKnoten] == false) {
                        this.adjazenzMatrix[quellKnoten][zielKnoten] = true;
                        this.kantenZahl++;
                }
        }

        public void knotenEinfuegen() {

                boolean[][] neuMatrix = new boolean[this.adjazenzMatrix.length + 1]
                                                   [this.adjazenzMatrix[0].length + 1];

                for (int i = 0; i < this.adjazenzMatrix.length; i++)
                        for (int j = 0; j < this.adjazenzMatrix[i].length; j++)
                                neuMatrix[i][j] = this.adjazenzMatrix[i][j];

                this.adjazenzMatrix = neuMatrix;
        }

        public String toString() {

                String retVal = "";
                for (int i = 0; i < this.adjazenzMatrix.length; i++) {
                        for (int j = 0; j < this.adjazenzMatrix[i].length; j++)
                                if (this.adjazenzMatrix[i][j] == true)
                                        retVal = retVal + "1 ";
                                else
                                        retVal = retVal + "0 ";
                        retVal = retVal + "\n";
                }
                return retVal;
        }
}
```