实现思路

邻接矩阵是一个二维数组，数据项表示两点间是否存在边，如果图中有 N 个顶点，邻接矩阵就是 N*N 的数组

	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	0
D	1	1	0	0

如上图，有四个顶点A、B、C、D。 1表示有边，0表示没有边，也可以用布尔变量true和false来表示。顶点与自身相连用 0 表示

实现代码

```java
package 图;

import java.util.ArrayList;
import java.util.List;

/**
 * 无向图 -- 邻接矩阵
 * @param <Item> 顶点类型
 */
public class MatrixGraph<Item> {
    // 顶点数量
    private int vertexNum;
    // 边的数量
    private int edgeNum;
    // 邻接矩阵
    private boolean[][] matrix;
    // 存放所有顶点信息
    private Item[] vertexInfo;

    // 初始化有V个顶点的图，未加边
    public MatrixGraph(Item[] vertexInfo) {
        this.vertexNum = vertexInfo.length;
        this.vertexInfo = vertexInfo;

        matrix = new boolean[vertexNum][vertexNum];
    }

    public MatrixGraph(Item[] vertexInfo, int[][] edges) {
        this(vertexInfo);
        for (int[] twoVertex : edges) {
            addEdge(twoVertex[0], twoVertex[1]);
        }
    }

    public MatrixGraph(int vertexNum) {
        this.vertexNum = vertexNum;
        matrix = new boolean[vertexNum][vertexNum];
    }

```

```java
    public MatrixGraph(int vertexNum, int[][] edges) {
        this(vertexNum);
        for (int[] twoVertex : edges) {
            addEdge(twoVertex[0], twoVertex[1]);
        }
    }

    public void addEdge(int i, int j) {
        // 对称矩阵，所以a[i][j] = a[j][i]
        matrix[i][j] = true;
        matrix[j][i] = true;
        edgeNum++;
    }

    public int vertexNum() {
        return vertexNum;
    }

    public int edgenum() {
        return edgeNum;
    }

    public Item getVertexInfo(int i) {
        return vertexInfo[i];
    }

    // 求某顶点的所有邻接顶点
    public List<Integer> adj(int i) {
        List<Integer> vertexAdj = new ArrayList<>();
        for (int j = 0; j < matrix[i].length; j++) {
            if (matrix[i][j]) {
                vertexAdj.add(j);
            }
        }
        return vertexAdj;
    }

    // 某顶点的度
    public int degree(int i) {
        int degree = 0;
        for (int j = 0; j < matrix[i].length; j++) {
            if (matrix[i][j]) {
               degree++;
            }
        }
        return degree;
    }

    // 求图的最大度数
    public int maxDegree() {
        int max = 0;
        for (int i = 0; i < vertexNum; i++) {
            if (degree(i) > max) {
                max = degree(i);
```

```
            }
        }
        return max;
    }
    // 求图的平均度数
    // 边的条数 = 顶点度之和的一半  因为一条边对应两个顶点，这两个顶点的度数之和为2，所以边的数量是度之和的一半这样的
     // edgeNum = sum / 2, 则sum = 2 * edgeNum, 于是avgDegree = sum / vertexNum
     public double avgDegree() {
         return 2.0 * edgeNum / vertexNum;
     }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(vertexNum).append("个顶点, ").append(edgeNum).append("条边。\n");
        for (int i = 0; i < vertexNum; i++) {
            sb.append(i).append(": ").append(adj(i)).append("\n");
        }
        return sb.toString();
    }

    public static void main(String[] args) {
        String[] vertexInfo = {"v0", "v1", "v2", "v3", "v4"};
        int[][] edges = {{0, 1}, {0, 2}, {0, 3},
                {1, 3}, {1, 4},
                {2, 4}};
        MatrixGraph<String> graph = new MatrixGraph<>(vertexInfo,edges);

        System.out.println("顶点3的度为" + graph.degree(3));
        System.out.println("顶点3的邻接点为"+graph.adj(3));
        System.out.println("该图的最大度数为" + graph.maxDegree());
        System.out.println("该图的平均度数为" + graph.avgDegree());
        System.out.println("邻接矩阵如下:\n" + graph);
    }
}

/* 输出

顶点3的度为2
顶点3的邻接点为[0, 1]
该图的最大度数为3
该图的平均度数为2.4
邻接矩阵如下:
5个顶点, 6条边。
0: [1, 2, 3]
1: [0, 3, 4]
2: [0, 4]
3: [0, 1]
4: [1, 2]

*/
```