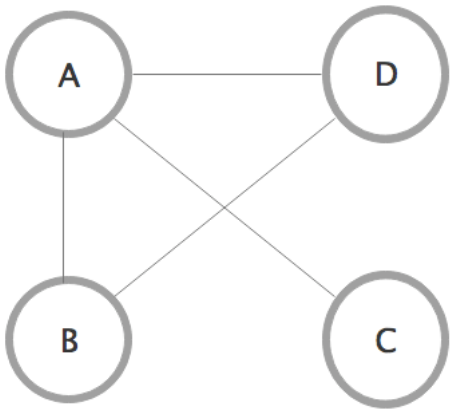
实现思路

邻接表是一个链表数组（或者是链表的链表），每个单独的链表表示了有哪些顶点与当前顶点邻接



顶点	包含邻接顶点的链表
A	B——>C——>D
B	A——>D
C	A
D	A——>B

实现代码

```java
package 图;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;

/**
 * 无向图
 * @param <Item>
 */
public class UndiGraph<Item> {
    // 顶点数量
    private int vertexNum;
    // 边的数量
    private int edgeNum;
    // 邻接表
    private List<List<Integer>> adj;
    // 顶点信息
```

```java
    private List<Item> vertexInfo;

    public UndiGraph(List<Item> vertexInfo) {
        this.vertexInfo = vertexInfo;
        this.vertexNum = vertexInfo.size();
        adj = new ArrayList<>();
        for (int i = 0; i < vertexNum; i++) {
            adj.add(new LinkedList<>());
        }
    }

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

    public int vertexNum() {
        return vertexNum;
    }

    public int edgeNum() {
        return edgeNum;
    }

    public void addEdge(int i, int j) {
        adj.get(i).add(j);
        adj.get(j).add(i);
        edgeNum++;
    }
    // 不需要set，所以不用返回List，返回可迭代对象就够了
    public Iterable<Integer> adj(int i) {
        return adj.get(i);
    }

    public Item getVertexInfo(int i) {
        return vertexInfo.get(i);
    }

    public int degree(int i) {
        return adj.get(i).size();
    }

    public int maxDegree() {
        int max = 0;
        for (int i = 0;i < vertexNum;i++) {
            if (degree(i) > max) {
                max = degree(i);
            }
        }
        return max;
    }

```

```
    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.get(i)).append("\n");
        }
        return sb.toString();
    }

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

        UndiGraph<String> graph = new UndiGraph<>(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("邻接表如下:\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
邻接表如下:
5个顶点, 6条边。
0: [1, 2, 3]
1: [0, 3, 4]
2: [0, 4]
3: [0, 1]
4: [1, 2]
*/
```