实现思路

顾名思义它更关注边，我们可以用一个Edge来抽象边，它有两个int成员表示该边的两个顶点，如果是加权图，再多一个int型的weight表List<Edge>中，就是我们所说的边的数组了

实现代码

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

public class EdgeGraph<Item> {

    public static class Edge {
        private int either;
        private int other;

         public int either() {
             return either;
         }

         public int other() {
             return other;
         }

         public Edge(int either, int other) {
             this.either = either;
             this.other = other;
         }

         @Override
         public String toString() {
             return "Edge{" +
                     "either=" + either +
                     ", other=" + other +
                     '}';
         }
    }

    private int vertexNum;
    private int edgeNum;
    private List<Item> vertexInfo;
    private List<Edge> edges;

    public EdgeGraph(List<Item> vertexInfo) {
        this.edges = new ArrayList<>();
        this.vertexInfo = vertexInfo;
        this.vertexNum = vertexInfo.size();
    }

    public EdgeGraph(List<Item> vertexInfo, int[][] edges) {
        this(vertexInfo);
        for (int[] twoVertex : edges) {
```

```
            addEdge(twoVertex[0], twoVertex[1]);
        }
    }

    public EdgeGraph(int vertexNum) {
        this.edges = new ArrayList<>();
        this.vertexNum = vertexNum;
    }

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

    public void addEdge(int i, int j) {
        Edge edge = new Edge(i, j);
        this.edges.add(edge);
        edgeNum++;
    }

    public List<Integer> adj(int i) {
        List<Integer> adj = new ArrayList<>();

        for (Edge edge : edges) {
            if (edge.either == i) {
                adj.add(edge.other);
            } else if (edge.other == i) {
                adj.add(edge.either);
            }
        }
        return adj;
    }

    public int degree(int i) {
        return adj(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;
    }

     public Item getVertexInfo(int i) {
```

```
        return vertexInfo.get(i);
    }

    public int vertexNum() {
        return vertexNum;
    }

    public int edgeNum() {
        return edgeNum;
    }

    @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) {
        List<String> vertexInfo = Arrays.asList("v0", "v1", "v2", "v3", "v4");
        int[][] edges = {{0, 1}, {0, 2}, {0, 3},
                {1, 3}, {1, 4},
                {2, 4}};
        EdgeGraph<String> graph = new EdgeGraph<>(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]
*/
```