

## 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

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4
5 public class EdgeGraph<Item> {
6
7     public static class Edge {
8         private int either;
9         private int other;
10
11         public int either() {
12             return either;
13         }
14
15         public int other() {
16             return other;
17         }
18
19         public Edge(int either, int other) {
20             this.either = either;
21             this.other = other;
22         }
23
24         @Override
25         public String toString() {
26             return "Edge{" +
27                 "either=" + either +
28                 ", other=" + other +
29                 '}';
30         }
31     }
32
33     private int vertexNum;
34     private int edgeNum;
35     private List<Item> vertexInfo;
36     private List<Edge> edges;
37
38     public EdgeGraph(List<Item> vertexInfo) {
39         this.edges = new ArrayList<>();
40         this.vertexInfo = vertexInfo;
41         this.vertexNum = vertexInfo.size();
42     }
43
44     public EdgeGraph(List<Item> vertexInfo, int[][] edges) {
45         this(vertexInfo);
46         for (int[] twoVertex : edges) {
```

```

47         addEdge(twoVertex[0], twoVertex[1]);
48     }
49 }
50
51 public EdgeGraph(int vertexNum) {
52     this.edges = new ArrayList<>();
53     this.vertexNum = vertexNum;
54 }
55
56 public EdgeGraph(int vertexNum, int[][] edges) {
57     this(vertexNum);
58     for (int[] twoVertex : edges) {
59         addEdge(twoVertex[0], twoVertex[1]);
60     }
61 }
62
63 public void addEdge(int i, int j) {
64     Edge edge = new Edge(i, j);
65     this.edges.add(edge);
66     edgeNum++;
67 }
68
69 public List<Integer> adj(int i) {
70     List<Integer> adj = new ArrayList<>();
71
72     for (Edge edge : edges) {
73         if (edge.either == i) {
74             adj.add(edge.other);
75         } else if (edge.other == i) {
76             adj.add(edge.either);
77         }
78     }
79     return adj;
80 }
81
82 public int degree(int i) {
83     return adj(i).size();
84 }
85
86 public int maxDegree() {
87     int max = 0;
88     for (int i = 0; i < vertexNum; i++) {
89         if (degree(i) > max) {
90             max = degree(i);
91         }
92     }
93     return max;
94 }
95
96 public double avgDegree() {
97     return 2.0 * edgeNum / vertexNum;
98 }
99
100 public Item getVertexInfo(int i) {

```

```

101         return vertexInfo.get(i);
102     }
103
104     public int vertexNum() {
105         return vertexNum;
106     }
107
108     public int edgeNum() {
109         return edgeNum;
110     }
111
112     @Override
113     public String toString() {
114         StringBuilder sb = new StringBuilder();
115         sb.append(vertexNum).append("个顶点, ").append(edgeNum).append("条边.\n");
116         for (int i = 0; i < vertexNum; i++) {
117             sb.append(i).append(": ").append(adj(i)).append("\n");
118         }
119         return sb.toString();
120     }
121
122     public static void main(String[] args) {
123         List<String> vertexInfo = Arrays.asList("v0", "v1", "v2", "v3", "v4");
124         int[][] edges = {{0, 1}, {0, 2}, {0, 3},
125             {1, 3}, {1, 4},
126             {2, 4}};
127         EdgeGraph<String> graph = new EdgeGraph<>(vertexInfo, edges);
128         System.out.println("顶点3的度为" + graph.degree(3));
129         System.out.println("顶点3的邻接点为" + graph.adj(3));
130         System.out.println("该图的最大度数为" + graph.maxDegree());
131         System.out.println("该图的平均度数为" + graph.avgDegree());
132         System.out.println("邻接表如下:\n" + graph);
133     }
134 }
135
136 /* 输出
137 顶点3的度为2
138 顶点3的邻接点为[0, 1]
139 该图的最大度数为3
140 该图的平均度数为2.4
141 邻接表如下:
142 5个顶点, 6条边。
143 0: [1, 2, 3]
144 1: [0, 3, 4]
145 2: [0, 4]
146 3: [0, 1]
147 4: [1, 2]
148 */

```