

Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 9 REPORT

EMRE ÇELİK
141044024

Course Assistant:
AHMET SOYYIĞIT

Problem Solutions Approach

Önce AbstractGraphExtended sınıfı oluşturdum.addRandomEdgesToGraph methodunda limit parametresini kullanarak 0,limit aralığında random sayı oluşturup o kadar edge ekliyorum.Edge zaten varsa eklemedim.Eklenen sayıyı return ettim.

breadthFirstSearch methodunu kitaptan alıp , static olan methodu sınıf methodu olarak değiştirdim. Method , verilen parametreyle o indisten başlayıp breadh first search yapıp int(parentlar) arrayi return ediyor.

getConnectedComponentUndirectedGraph methodunda ise önce directed ise null döndürdüm.Matrix ya da List mi olduğunu instanceof ile kontrol ettim.

Birbiriyle komsuluğu olan edge leri ayrı ayrı graphlar yapıp.Bu graphları Graph arrayi olarak return ettim.Bunu yaparken Linkedlist ,Queue ,ArrayList veri yapılarını kullandım.Her vertex için breadthfirst methodunu çağırıp gezdiğim yerleri çıkardım ve bu şekilde ayrı ayrı graphlar elde ettim.

isBipartiteUndirectedGraph methodunda önce graph directed sa false döndürdüm.Sonrasında color arrayi tutup -1 ile initialize ettim.Vertexleri tutacağım bir Queue veri yapısı kullandım.Queue deki her eleman için tüm vertexleri ile bu elemanın edge olusturup olusturmadığına bakarak , eğer edge varsa ve color arrayinde o dest -1 ise queue ya bunu ekledim.Aksi halde ise bu elemandan dest e edge varsa ve bu 2 elemanın renkleri aynıysa false döndürdüm.Method ,queue boşalınca true döndürdü.

writeGraphToFile methodunda ise example dosyasındaki formatla aynı şekilde , bulunulan graphı dosyaya yazdım.

Test Cases

Test 1 : Undirected ListGraph oluşturup içine edge'ler ekledim ve getConnectedComponentUndirectedGraph ve isBipartite methodları için test yaptım ve oluşan graph iki ayrı connected graph olarak bölündü.Ve bipartite olduğu sonucuna varıldı.

Test 2 : Directed MatrixGraph ile breadthFirstSearch ve writeGraphToFile methodları için test yaptım.BreadthFirstSearch çıktısı gezilen vertexlerin parent arrayi olarak geldi. writeGraphToFile'da ise input dosyasıyla aynı formatta yazıldı.

Test 3 : Directed MatrixGraph ile breadthFirstSearch , addRandomEdgesToGraph ve writeGraphToFile methodları için test yaptım.Dosyadan okunan graphın üstüne 20 limit verip random edge ekledim(10 tane eklendi). Graph ,yeni eklenen edge'lerle birlikte output dosyasına yazılmıştır.

Undirected ListGraph ile getConnectedComponentUndirectedGraph ve isBipartite methodları testi

The screenshot shows the IntelliJ IDEA IDE with the following components:

- Top Toolbar:** Includes buttons for Run, Debug, and other IDE functions.
- Main Editor:** Displays the 'main' method of a class. The code creates a `ListGraph` object, inserts edges, and prints the graph structure. The output shows a graph with 10 nodes and 10 edges.
- Bottom Panel:** Shows the output of the program, displaying the graph structure as a list of edges and nodes.

Directed MatrixGraph ile breadthFirstSearch ve writeGraphToFile methodları testi

The screenshot displays an IDE with the following components:

- Top Bar:** Shows "Paz 17:36" and window management icons.
- Menu Bar:** Includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help.
- Project Explorer (Left):** Shows the project structure:
 - HW9 (src)
 - src
 - AbstractGraph
 - AbstractGraphExtended
 - Edge
 - Graph
 - ListGraph
 - Main
 - MatrixGraph
 - ExampleGraphInputOrOutputFile.txt
 - HW9.iml
 - outputGraph.txt
- Editor (Center):** Displays the code for `Main.java`:


```

1  package main;
2  public class Main {
3      public static void main(String[] args) throws Exception {
4          try {
5              Scanner sc = new Scanner(new File(pathname = "ExampleGraphInputOrOutputFile.txt"));
6              MatrixGraph graph = (MatrixGraph) AbstractGraph.createGraph(sc, undirected, true, type = "Matrix");
7
8              for(int i = 0; i < graph.getW(); i++) {
9                  Iterator itr = graph.edgeIterator(i);
10                 while (itr.hasNext())
11                     System.out.println(itr.next());
12             }
13
14             System.out.println("Breadth First Search Test : ");
15             int[] bds = graph.breadthFirstSearch(0);
16
17             for(int i = 0; i < bds.length; i++)
18                 System.out.print(bds[i] + " ");
19
20             System.out.println();
21             graph.writeGraphToFile(filename = "outputGraph.txt");
22
23             ListGraph gp = new ListGraph(11, false);
24             /* ListGraph gp = new ListGraph(11, false);
25              *
26              */
27         }
28     }
29 }
      
```
- Run Console (Bottom Left):** Shows the execution output:


```

/home/emre/Desktop/101.8.8_111/bin/java ...
[0, 1]: 1.0
[1, 0]: 1.0
[1, 2]: 1.0
[1, 4]: 1.0
[1, 5]: 1.0
[2, 5]: 1.0
[3, 6]: 1.0
[4, 6]: 1.0
[4, 7]: 1.0
[5, 7]: 1.0
[6, 8]: 1.0
[7, 8]: 1.0
Breadth First Search Test :
-1 0 1 0 1 1 3 4 6
Process finished with exit code 0
      
```
- Output View (Bottom Right):** Displays the content of `outputGraph.txt`:


```

1 0
2 0 1
3 0 3
4 1 2
5 1 4
6 1 5
7 2 5
8 3 6
9 4 6
10 4 7
11 5 7
12 6 8
13 7 8
      
```

Directed MatrixGraph ile breadthFirstSearch , addRandomEdgesToGraph ve writeGraphToFile methodları testi

Kod

Activities JetBrains-IDEA-ce

Paz 18:11

HW9 - [home/emre/Desktop/DATA STRUCTURES HOMEWORKS/HW9] - [HW9] - ./src/Main.java - IntelliJ IDEA 2016.3.5

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

HW9 - [src] - Main

Project

- HW9 - [home/emre/Desktop/DATA STRUCTURES HOMEWORKS/HW9]
- idea
- out
- src
 - AbstractGraph
 - AbstractGraphExtended
 - Edge
 - Graph
 - ListGraph
 - Main
 - MatrixGraph
 - MatrixGraphPutOrOutputFile.txt
 - ExampleGraphInputOrOutputFile.txt
 - HW9.iml
 - outputGraph.txt
- External Libraries

Main.java

```
1 import java.io.File;
2 import java.util.Iterator;
3 import java.util.List;
4 import java.util.Scanner;
5
6 public class Main {
7     public static void main(String[] args) throws Exception {
8
9         try {
10
11             Scanner sc = new Scanner(new File(pathname: "ExampleGraphInputOrOutputFile.txt"));
12
13             MatrixGraph graph = (MatrixGraph) AbstractGraph.createdGraph(sc, indexed true, type: "Matrix");
14
15             System.out.println("Graph");
16
17             for(int i = 0; i < graph.getNum(); i++) {
18                 Iterator itr = graph.edgeIterator(i);
19                 while (itr.hasNext())
20                     System.out.println(itr.next());
21             }
22
23             System.out.println("After adding random edges");
24
25             System.out.println(graph.addRandomEdgesToGraph( edgeLimit: 20) + " edge was added");
26
27             for(int i = 0; i < graph.getNum(); i++) {
28                 Iterator itr = graph.edgeIterator(i);
29                 while (itr.hasNext())
30                     System.out.println(itr.next());
31             }
32
33             System.out.println("Breadth First Search Text : ");
34             int[] bds = graph.breadthFirstSearch( start: 0);
35
36             for(int i = 0; i < bds.length; i++)
37                 System.out.print(bds[i] + " ");
38
39             System.out.println();
40
41             graph.writeGraphToFile( filename: "outputGraph.txt");
42
43             /* ListGraph gp = new ListGraph(11,false);
44              gp.insertNew Edge(1,1);
45              gp.insertNew Edge(1,2);
46              gp.insertNew Edge(2,3);
47              gp.insertNew Edge(2,4);
48              gp.insertNew Edge(4,5);
49              gp.insertNew Edge(5,6);
50              gp.insertNew Edge(6,7);
51              gp.insertNew Edge(7,8);
52              gp.insertNew Edge(8,9);
53              gp.insertNew Edge(9,10);
54              gp.insertNew Edge(10,11);
55              gp.insertNew Edge(11,12);
56              gp.insertNew Edge(12,13);
57              gp.insertNew Edge(13,14);
58              gp.insertNew Edge(14,15);
59              gp.insertNew Edge(15,16);
60              gp.insertNew Edge(16,17);
61              gp.insertNew Edge(17,18);
62              gp.insertNew Edge(18,19);
63              gp.insertNew Edge(19,20);
64              gp.insertNew Edge(20,21);
65              gp.insertNew Edge(21,22);
66              gp.insertNew Edge(22,23);
67              gp.insertNew Edge(23,24);
68              gp.insertNew Edge(24,25);
69              gp.insertNew Edge(25,26);
70              gp.insertNew Edge(26,27);
71              gp.insertNew Edge(27,28);
72              gp.insertNew Edge(28,29);
73              gp.insertNew Edge(29,30);
74              gp.insertNew Edge(30,31);
75              gp.insertNew Edge(31,32);
76              gp.insertNew Edge(32,33);
77              gp.insertNew Edge(33,34);
78              gp.insertNew Edge(34,35);
79              gp.insertNew Edge(35,36);
80              gp.insertNew Edge(36,37);
81              gp.insertNew Edge(37,38);
82              gp.insertNew Edge(38,39);
83              gp.insertNew Edge(39,40);
84              gp.insertNew Edge(40,41);
85              gp.insertNew Edge(41,42);
86              gp.insertNew Edge(42,43);
87              gp.insertNew Edge(43,44);
88              gp.insertNew Edge(44,45);
89              gp.insertNew Edge(45,46);
90              gp.insertNew Edge(46,47);
91              gp.insertNew Edge(47,48);
92              gp.insertNew Edge(48,49);
93              gp.insertNew Edge(49,50);
94              gp.insertNew Edge(50,51);
95              gp.insertNew Edge(51,52);
96              gp.insertNew Edge(52,53);
97              gp.insertNew Edge(53,54);
98              gp.insertNew Edge(54,55);
99              gp.insertNew Edge(55,56);
100             gp.insertNew Edge(56,57);
101             gp.insertNew Edge(57,58);
102             gp.insertNew Edge(58,59);
103             gp.insertNew Edge(59,60);
104             gp.insertNew Edge(60,61);
105             gp.insertNew Edge(61,62);
106             gp.insertNew Edge(62,63);
107             gp.insertNew Edge(63,64);
108             gp.insertNew Edge(64,65);
109             gp.insertNew Edge(65,66);
110             gp.insertNew Edge(66,67);
111             gp.insertNew Edge(67,68);
112             gp.insertNew Edge(68,69);
113             gp.insertNew Edge(69,70);
114             gp.insertNew Edge(70,71);
115             gp.insertNew Edge(71,72);
116             gp.insertNew Edge(72,73);
117             gp.insertNew Edge(73,74);
118             gp.insertNew Edge(74,75);
119             gp.insertNew Edge(75,76);
120             gp.insertNew Edge(76,77);
121             gp.insertNew Edge(77,78);
122             gp.insertNew Edge(78,79);
123             gp.insertNew Edge(79,80);
124             gp.insertNew Edge(80,81);
125             gp.insertNew Edge(81,82);
126             gp.insertNew Edge(82,83);
127             gp.insertNew Edge(83,84);
128             gp.insertNew Edge(84,85);
129             gp.insertNew Edge(85,86);
130             gp.insertNew Edge(86,87);
131             gp.insertNew Edge(87,88);
132             gp.insertNew Edge(88,89);
133             gp.insertNew Edge(89,90);
134             gp.insertNew Edge(90,91);
135             gp.insertNew Edge(91,92);
136             gp.insertNew Edge(92,93);
137             gp.insertNew Edge(93,94);
138             gp.insertNew Edge(94,95);
139             gp.insertNew Edge(95,96);
140             gp.insertNew Edge(96,97);
141             gp.insertNew Edge(97,98);
142             gp.insertNew Edge(98,99);
143             gp.insertNew Edge(99,100);
144             gp.insertNew Edge(100,101);
145             gp.insertNew Edge(101,102);
146             gp.insertNew Edge(102,103);
147             gp.insertNew Edge(103,104);
148             gp.insertNew Edge(104,105);
149             gp.insertNew Edge(105,106);
150             gp.insertNew Edge(106,107);
151             gp.insertNew Edge(107,108);
152             gp.insertNew Edge(108,109);
153             gp.insertNew Edge(109,110);
154             gp.insertNew Edge(110,111);
155             gp.insertNew Edge(111,112);
156             gp.insertNew Edge(112,113);
157             gp.insertNew Edge(113,114);
158             gp.insertNew Edge(114,115);
159             gp.insertNew Edge(115,116);
160             gp.insertNew Edge(116,117);
161             gp.insertNew Edge(117,118);
162             gp.insertNew Edge(118,119);
163             gp.insertNew Edge(119,120);
164             gp.insertNew Edge(120,121);
165             gp.insertNew Edge(121,122);
166             gp.insertNew Edge(122,123);
167             gp.insertNew Edge(123,124);
168             gp.insertNew Edge(124,125);
169             gp.insertNew Edge(125,126);
170             gp.insertNew Edge(126,127);
171             gp.insertNew Edge(127,128);
172             gp.insertNew Edge(128,129);
173             gp.insertNew Edge(129,130);
174             gp.insertNew Edge(130,131);
175             gp.insertNew Edge(131,132);
176             gp.insertNew Edge(132,133);
177             gp.insertNew Edge(133,134);
178             gp.insertNew Edge(134,135);
179             gp.insertNew Edge(135,136);
180             gp.insertNew Edge(136,137);
181             gp.insertNew Edge(137,138);
182             gp.insertNew Edge(138,139);
183             gp.insertNew Edge(139,140);
184             gp.insertNew Edge(140,141);
185             gp.insertNew Edge(141,142);
186             gp.insertNew Edge(142,143);
187             gp.insertNew Edge(143,144);
188             gp.insertNew Edge(144,145);
189             gp.insertNew Edge(145,146);
190             gp.insertNew Edge(146,147);
191             gp.insertNew Edge(147,148);
192             gp.insertNew Edge(148,149);
193             gp.insertNew Edge(149,150);
194             gp.insertNew Edge(150,151);
195             gp.insertNew Edge(151,152);
196             gp.insertNew Edge(152,153);
197             gp.insertNew Edge(153,154);
198             gp.insertNew Edge(154,155);
199             gp.insertNew Edge(155,156);
200             gp.insertNew Edge(156,157);
201             gp.insertNew Edge(157,158);
202             gp.insertNew Edge(158,159);
203             gp.insertNew Edge(159,160);
204             gp.insertNew Edge(160,161);
205             gp.insertNew Edge(161,162);
206             gp.insertNew Edge(162,163);
207             gp.insertNew Edge(163,164);
208             gp.insertNew Edge(164,165);
209             gp.insertNew Edge(165,166);
210             gp.insertNew Edge(166,167);
211             gp.insertNew Edge(167,168);
212             gp.insertNew Edge(168,169);
213             gp.insertNew Edge(169,170);
214             gp.insertNew Edge(170,171);
215             gp.insertNew Edge(171,172);
216             gp.insertNew Edge(172,173);
217             gp.insertNew Edge(173,174);
218             gp.insertNew Edge(174,175);
219             gp.insertNew Edge(175,176);
220             gp.insertNew Edge(176,177);
221             gp.insertNew Edge(177,178);
222             gp.insertNew Edge(178,179);
223             gp.insertNew Edge(179,180);
224             gp.insertNew Edge(180,181);
225             gp.insertNew Edge(181,182);
226             gp.insertNew Edge(182,183);
227             gp.insertNew Edge(183,184);
228             gp.insertNew Edge(184,185);
229             gp.insertNew Edge(185,186);
230             gp.insertNew Edge(186,187);
231             gp.insertNew Edge(187,188);
232             gp.insertNew Edge(188,189);
233             gp.insertNew Edge(189,190);
234             gp.insertNew Edge(190,191);
235             gp.insertNew Edge(191,192);
236             gp.insertNew Edge(192,193);
237             gp.insertNew Edge(193,194);
238             gp.insertNew Edge(194,195);
239             gp.insertNew Edge(195,196);
240             gp.insertNew Edge(196,197);

```

Output

The screenshot displays the IntelliJ IDEA IDE interface. The top bar shows the title 'HW9 - /home/emre/Desktop/DATA STRUCTURES HOMEWORKS/HW9 - [HW9] - .../src/Main.java - IntelliJ IDEA 2016.3.5'. The menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The toolbar contains icons for opening files, saving, running, and other IDE functions.

The Project view on the left shows the project structure:

- HW9 /home/emre/Desktop/DATA STRUCTURES HOMEWORKS/HW9
 - .idea
 - out
 - src
 - AbstractGraph
 - AbstractGraphExtended
 - Edge

The Main.java file is open in the editor, showing the following code:

```
1 import java.io.File;
2 import java.util.Iterator;
3 import java.util.List;
4 import java.util.Scanner;
5
6 public class Main {
7     public static void main(String[] args) throws Exception {
8
9         try {
```

The Run view at the bottom shows the execution output for the 'Main' class:

```
Graph
[(0, 1): 1.0]
[(0, 3): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(2, 5): 1.0]
[(3, 6): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(5, 7): 1.0]
[(6, 8): 1.0]
[(7, 8): 1.0]
After adding random edges
10 edge was added
[(0, 1): 1.0]
[(0, 3): 1.0]
[(1, 2): 1.0]
[(1, 4): 1.0]
[(1, 5): 1.0]
[(1, 6): 1.0]
[(2, 2): 1.0]
[(2, 5): 1.0]
[(2, 6): 1.0]
[(3, 4): 1.0]
[(3, 6): 1.0]
[(4, 3): 1.0]
[(4, 6): 1.0]
[(4, 7): 1.0]
[(4, 8): 1.0]
[(5, 6): 1.0]
[(5, 7): 1.0]
[(6, 8): 1.0]
[(7, 1): 1.0]
[(7, 5): 1.0]
[(7, 8): 1.0]
[(8, 6): 1.0]
Breadth First Search Test :
-1 0 1 0 1 1 1 4 4
Process finished with exit code 0
```

The status bar at the bottom indicates 'Compilation completed successfully in 2s 54ms (a minute ago)' and '43:1 LF: UTF-8'.

File Output

