



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ПРИРОДНО-МАТЕМАТИЧКИ  
ФАКУЛТЕТ  
ДЕПАРТМАН ЗА МАТЕМАТИКУ И  
ИНФОРМАТИКУ



# Извештај

-Социјалне мреже-

Главоњић Јован

552/17

Изјављујем под пуном одговорношћу да сам задатке решавао самостално

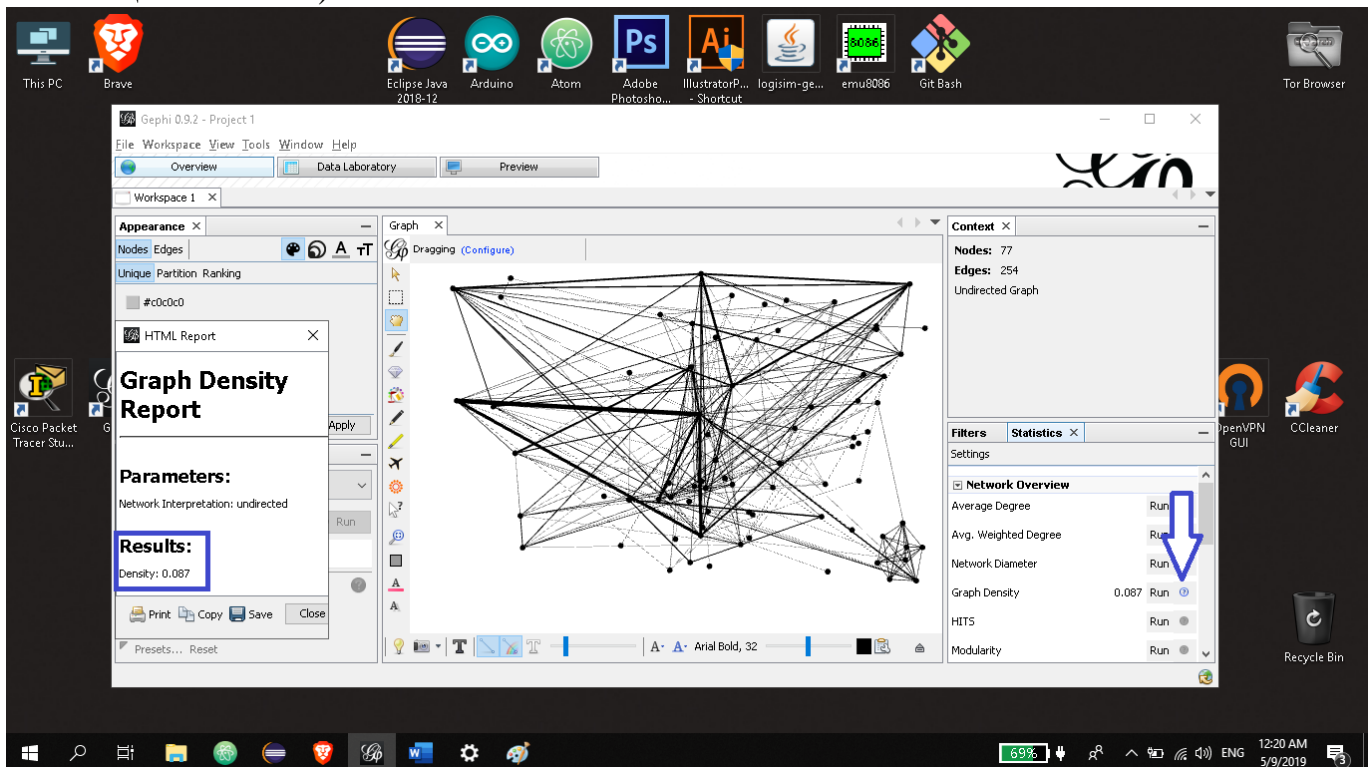
# Садржај

1. Основне метрике.....	3
• Густина мреже:.....	3
• Дијаметар мреже:.....	4
• Просечан коефицијент кластерисања:.....	5
• Просечна дистанца чворова: .....	6
• Просечан степен чвора:.....	7
2. Степен и централност.....	8
• Највећи степен .....	8
• Централности .....	10
❖ Betweenness: .....	10
❖ Closeness.....	11
❖ Eigenvector.....	12
3. Чворови у максималаном кору .....	14
4. Програмски део задатка.....	16
• Учитавање и детектовање .....	16
• Метрике.....	18
• Мејн класа .....	21

# 1. Основне метрике

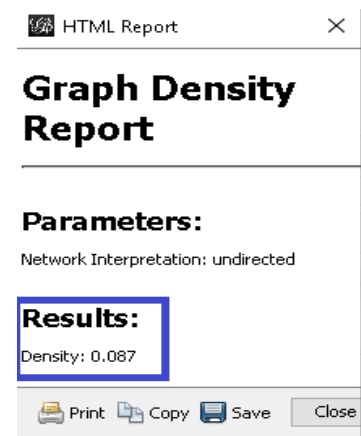
- **Густина мреже:**

Густина мреже у програму гефи се добија одабиром опције у менију за статистике са десне стране програма (плава стрелица показује на ту опцију на слици *слика 1.1*)



Слика 1.1

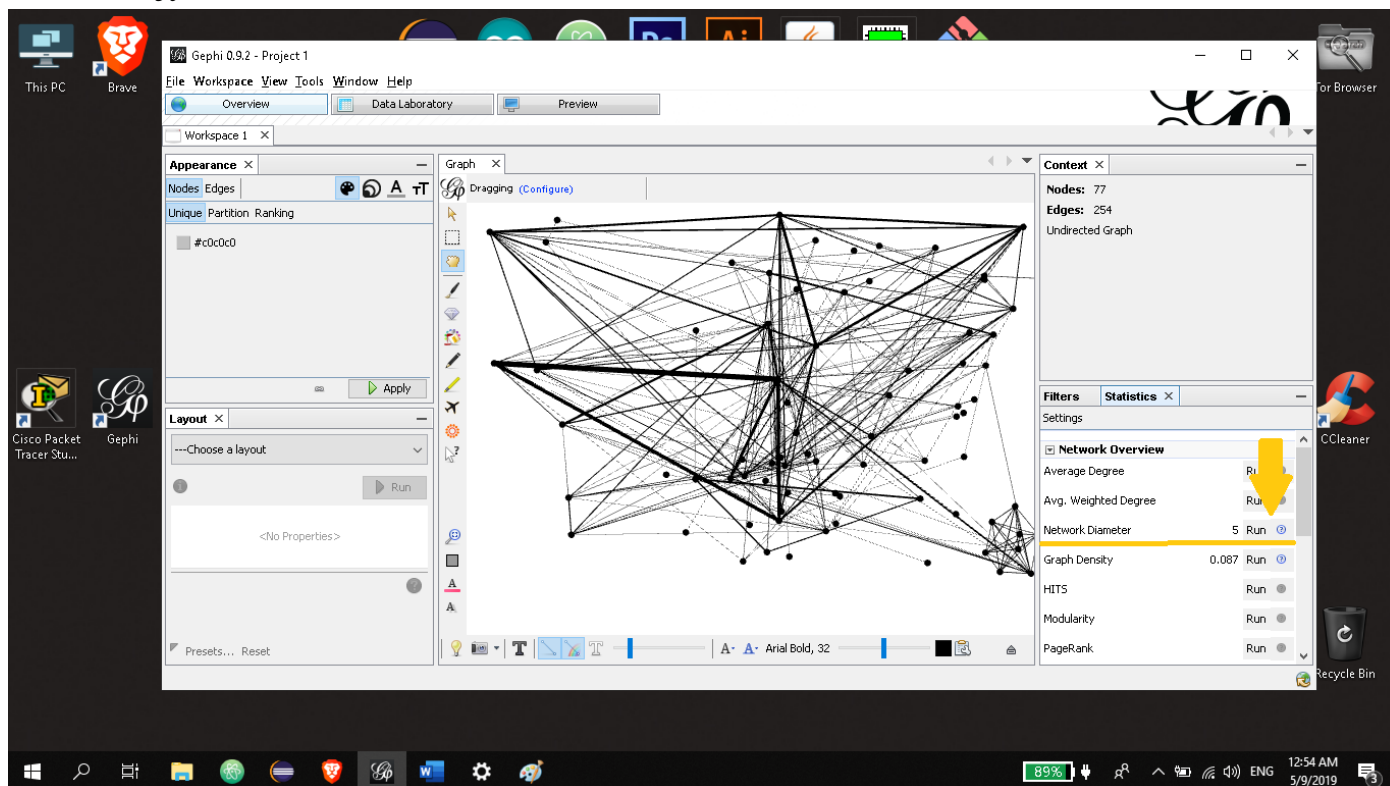
--Густина мреже ће се појавити у искачућем прозору (слика 1.2). У овоме случају густина износи: **0.087**



Слика 1.2

- **Дијаметар мреже:**

Дијаметар мреже је ништа друго него најдужа дистанца два чвора у мрежи. Добија се одабиром опције за приказ дијаметра (жута стрелица показује на ту опцију на слици *слика 1.3*).



*Слика 1.3*

--Унутар мреже коју ми посматрамо дијаметар износи: **5**

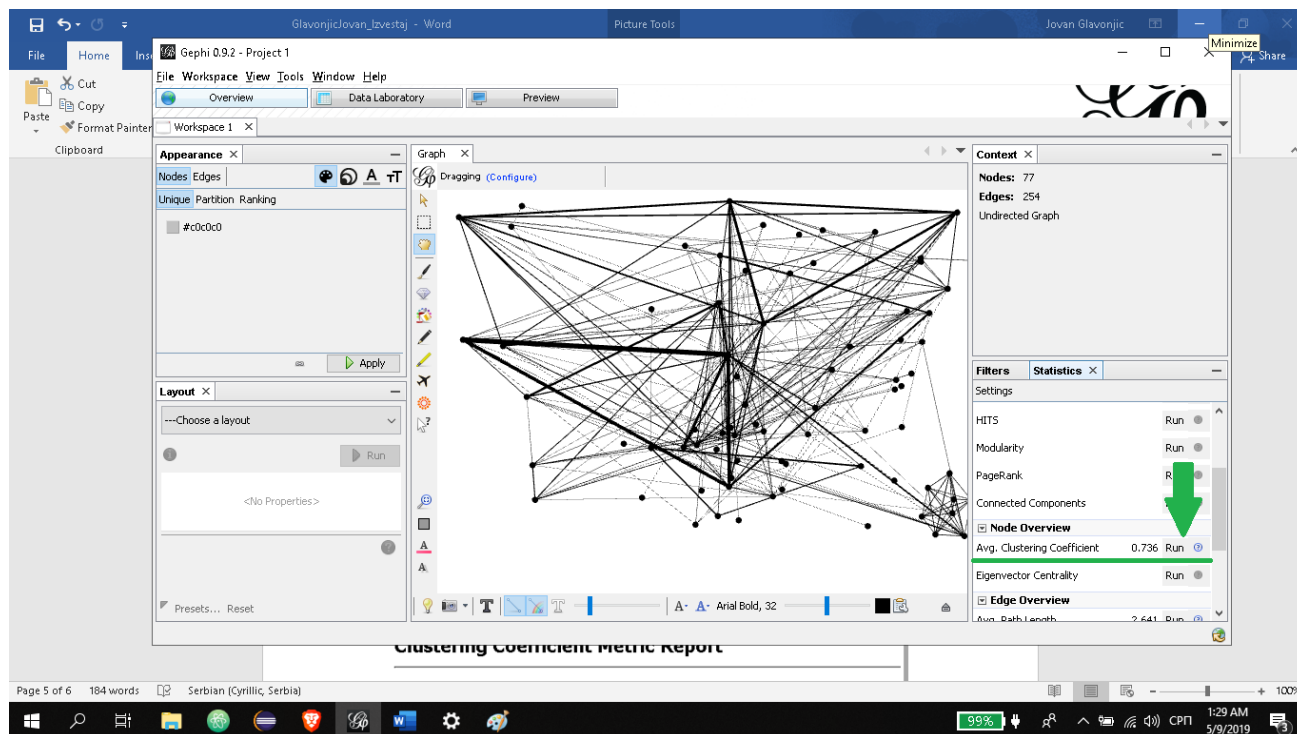
Као и приликом претходне опције, искачући прозор у програму ће дати резултат (*слика 1.4*).



*Слика 1.4*

- **Просечан коефицијент кластерисања:**

Просечан коефицијент кластерисања приказује како се одвија кластерисање, тј како су чворови уграђени са њиховим суседима. Добија се одабиром опције за приказ просечног коефицијента кластерисања унутар дела за преглед чворова (зелена стрелица показује на ту опцију на слици *слика 1.5*).



Слика 1.5

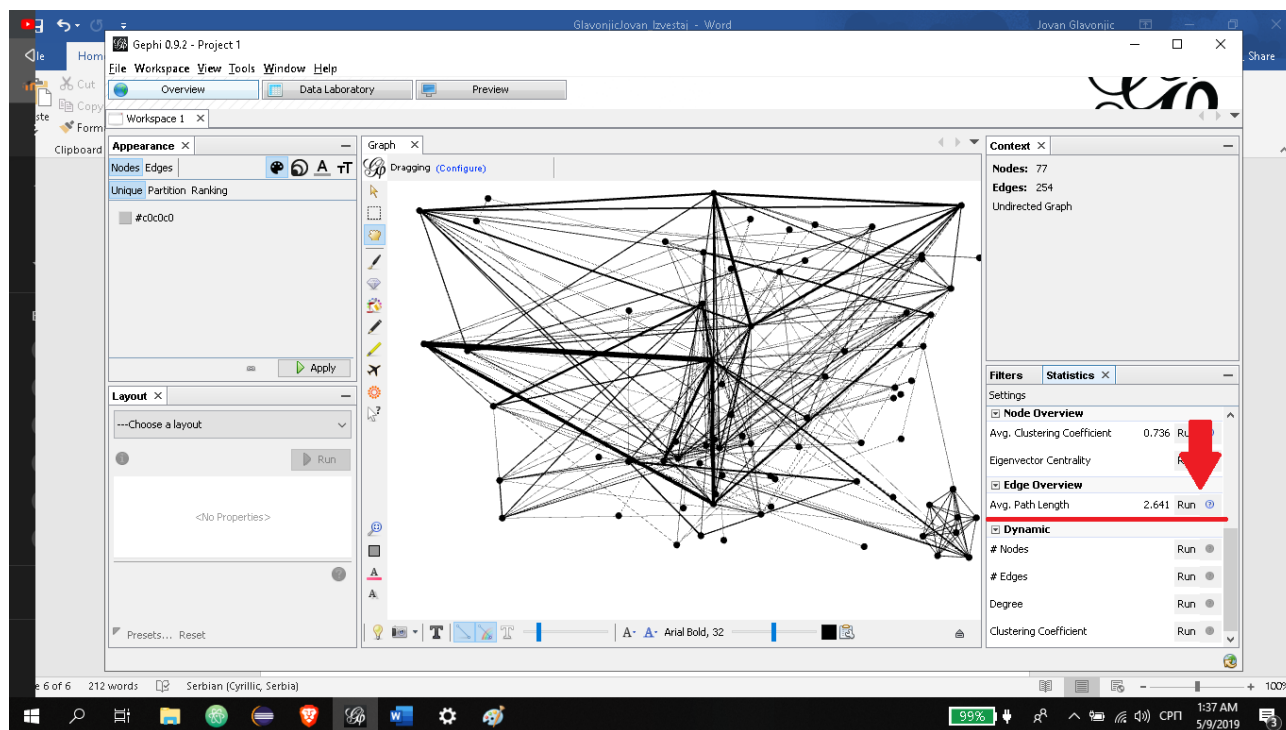
--Просечан коеф. клатерисања мреже износи: **0.736** (слика 1.6)



Слика 1.6

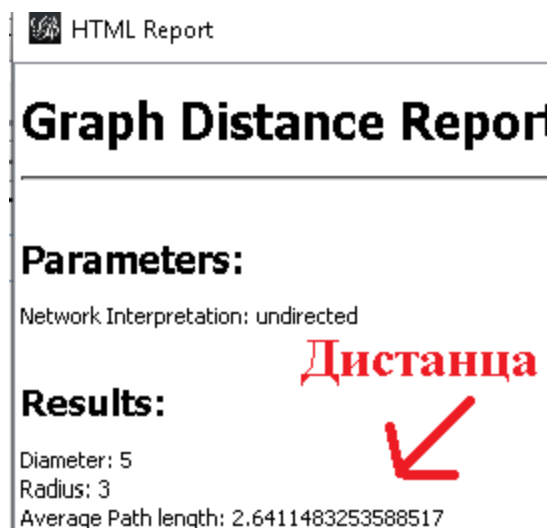
- **Просечна дистанца чворова:**

Просечна дистанца се добија опцијом за приказ просечне дистанце чворова унутар дела за преглед линкова (црвена стрелица показује на ту опцију на слици *слика 1.7*).



Слика 1.7

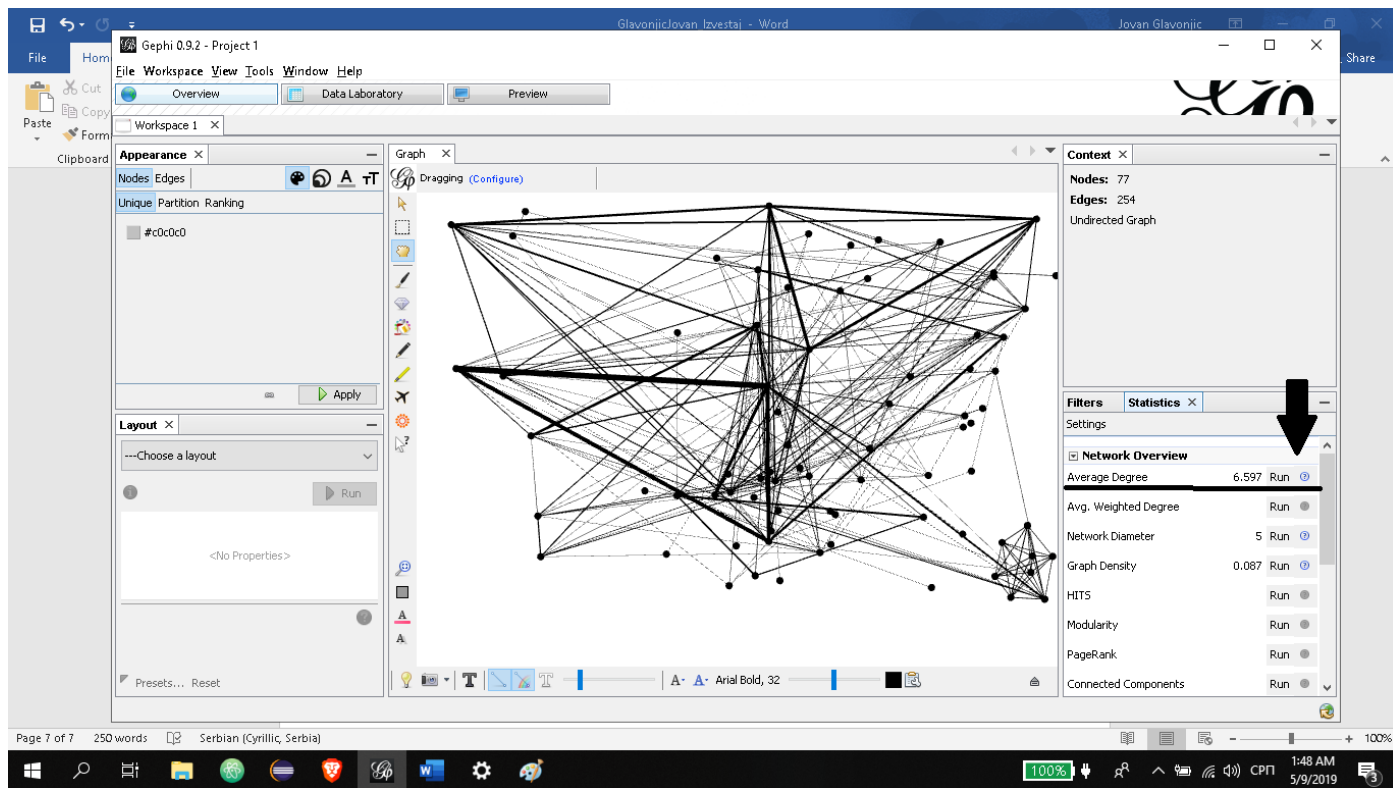
--Процечна дистанца чворова је: **2.641** (слика 1.8)



Слика 1.8

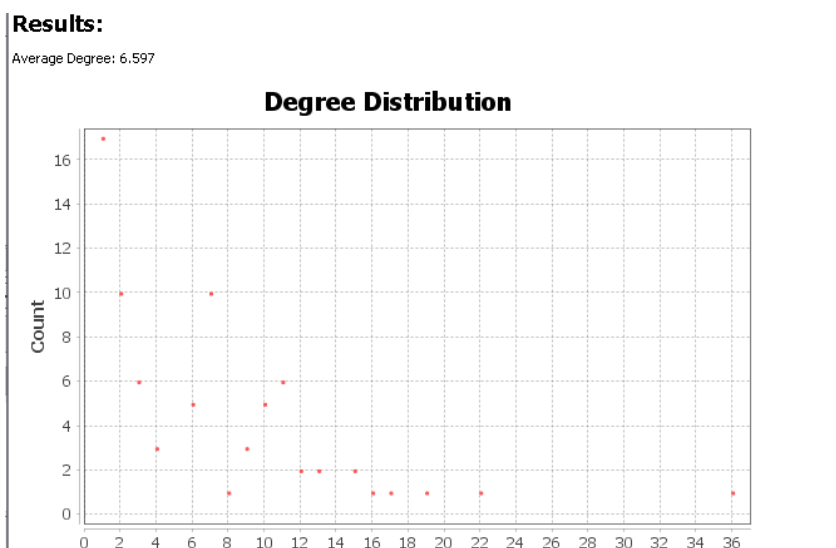
- **Просечан степен чвора:**

Просечан степен чвора се добија опцијом за приказ просечног степена чвора унутар статистика (црна стрелица показује на ту опцију на слици *слика 1.9*).



Слика 1.9

--Процечна степен чвора износи: **6.597** (слика 1.10)



Слика 1.10

## 2. Степен и централност

- **Највећи степен**

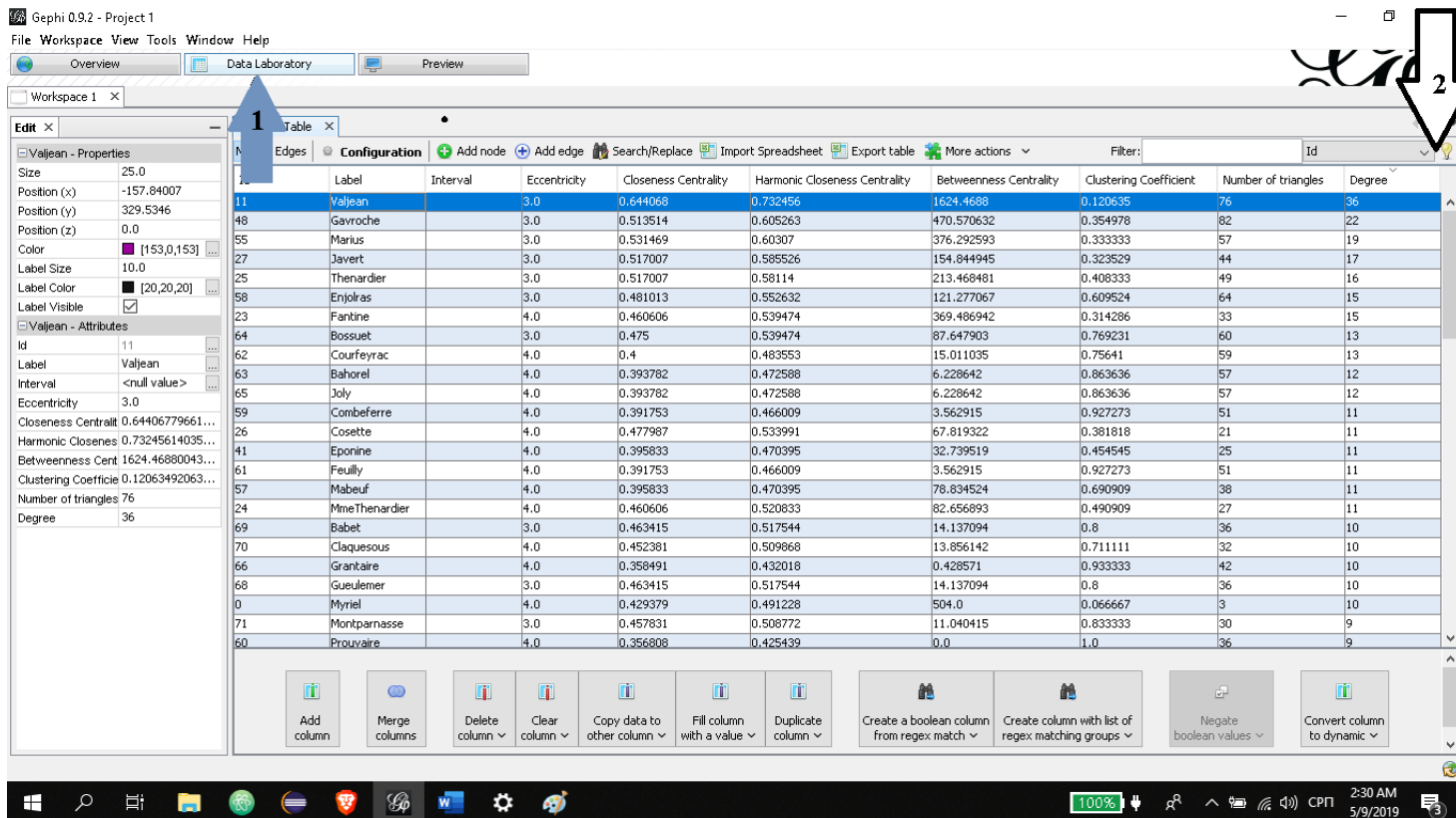
Табела топ десет чворова са највећим степеном:

Ид	Лабела	Степен
11	Valjean	36
48	Gavroche	22
55	Marius	19
27	Javert	17
25	Thenardier	16
23	Fantine	15
58	Enjolras	15
62	Courfeyrac	13
64	Bossuet	13
63	Bahorel	12

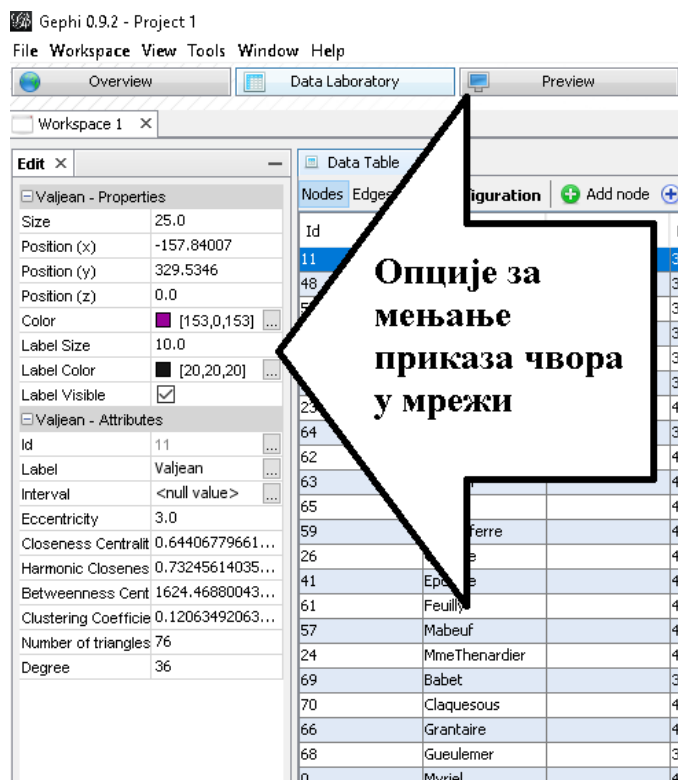
Приступ степенима чворова графа (*слика 2.1* – стрелица број **1** показује на опцију менија која приказује информације о чворовима, стрелица број **2** показује на опцију за сортирање чворова по степенима) омогућује нам да уочимо чворове који имају највећи степен. Као и да изменимо њихов визуелни приказ (*слика 2.2*). Након тога на графу имамо лепши и читкивији приказ информација (*слика 2.3*).

Напомена: Опције за сортирање се налазе у истом реду довољно је кликнути на било коју опцију, тј ону по којој хоћемо да сортирамо чворове мреже (*слика 2.1*).

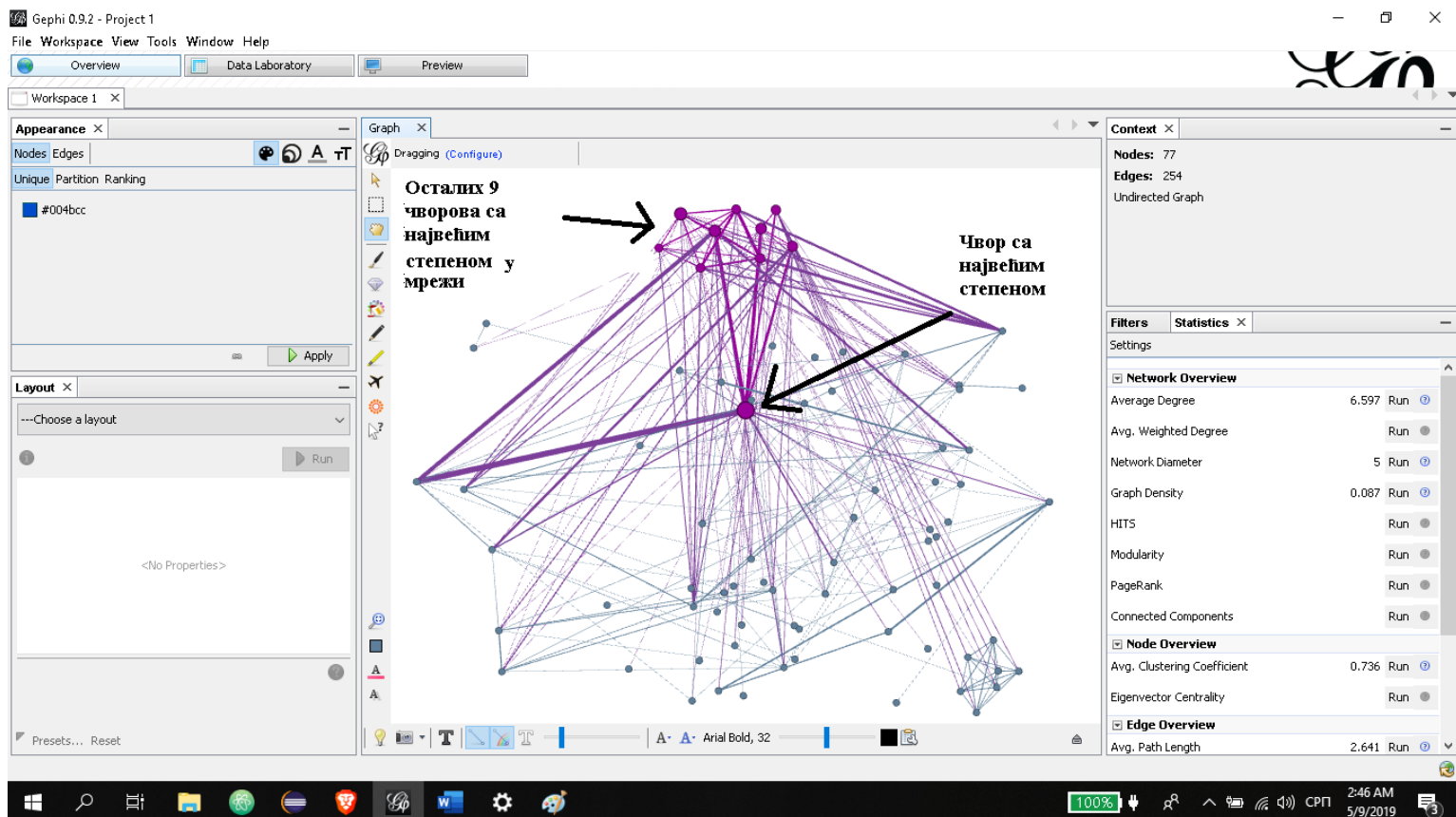




Слика 2.1



Слика 2.2



Слика 2.3

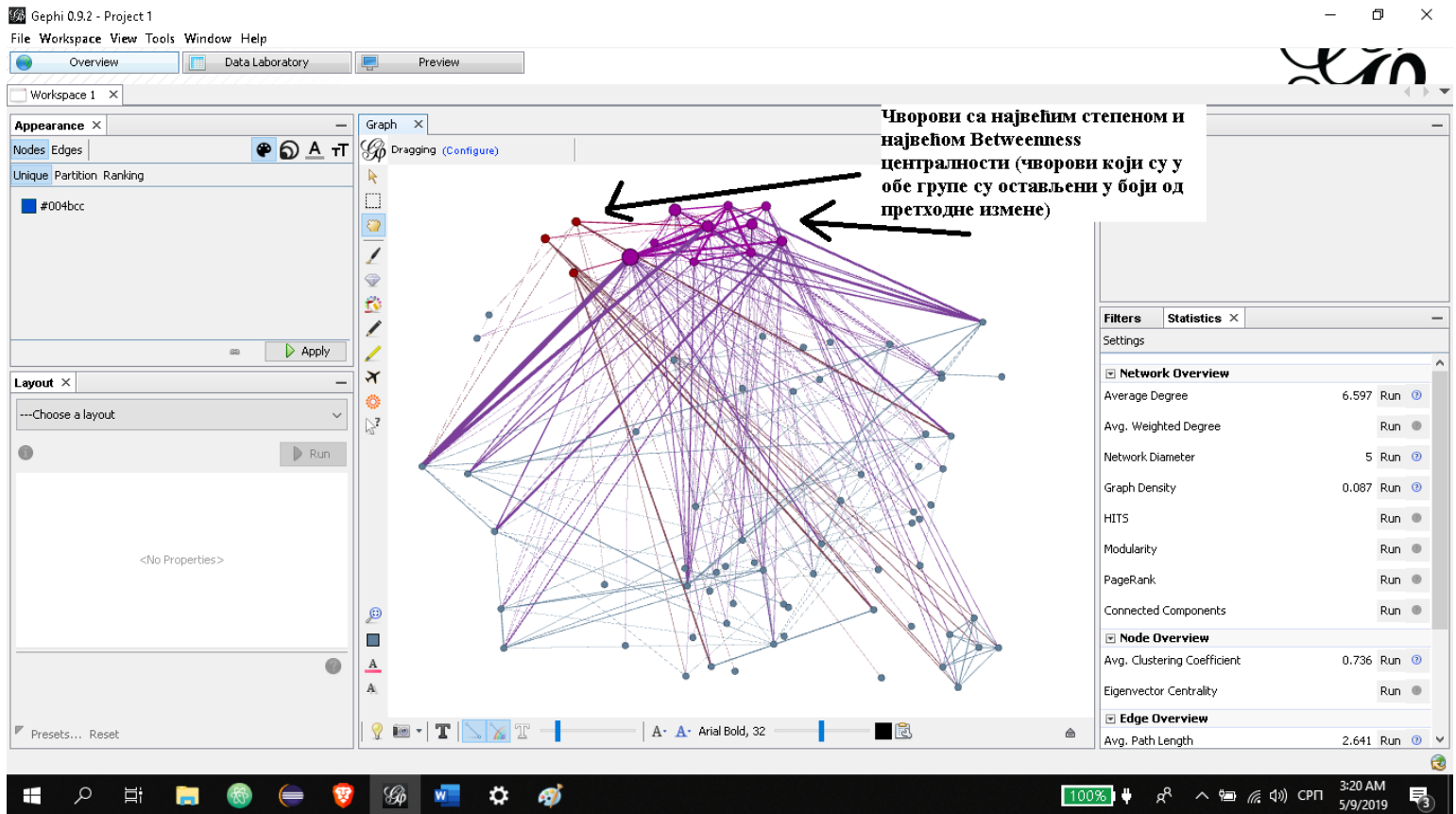
- Централности

- ❖ **Betweenness:**

Табела топ десет чворова:

Ид	Лабела	Betweenness
11	Valjean	1624.4688004333127
0	Myriel	504.0
48	Gavroche	470.57063191366586
55	Marius	376.2925925725461
23	Fantine	369.48694181635364
25	Thenardier	213.46848051759042
27	Javert	154.84494504463547
51	MlleGillenormand	135.65694444444445
58	Enjolras	121.27706694320474
16	Tholomyes	115.793642305407

Поново вршимо измену визуализације графа (слика 2.4).



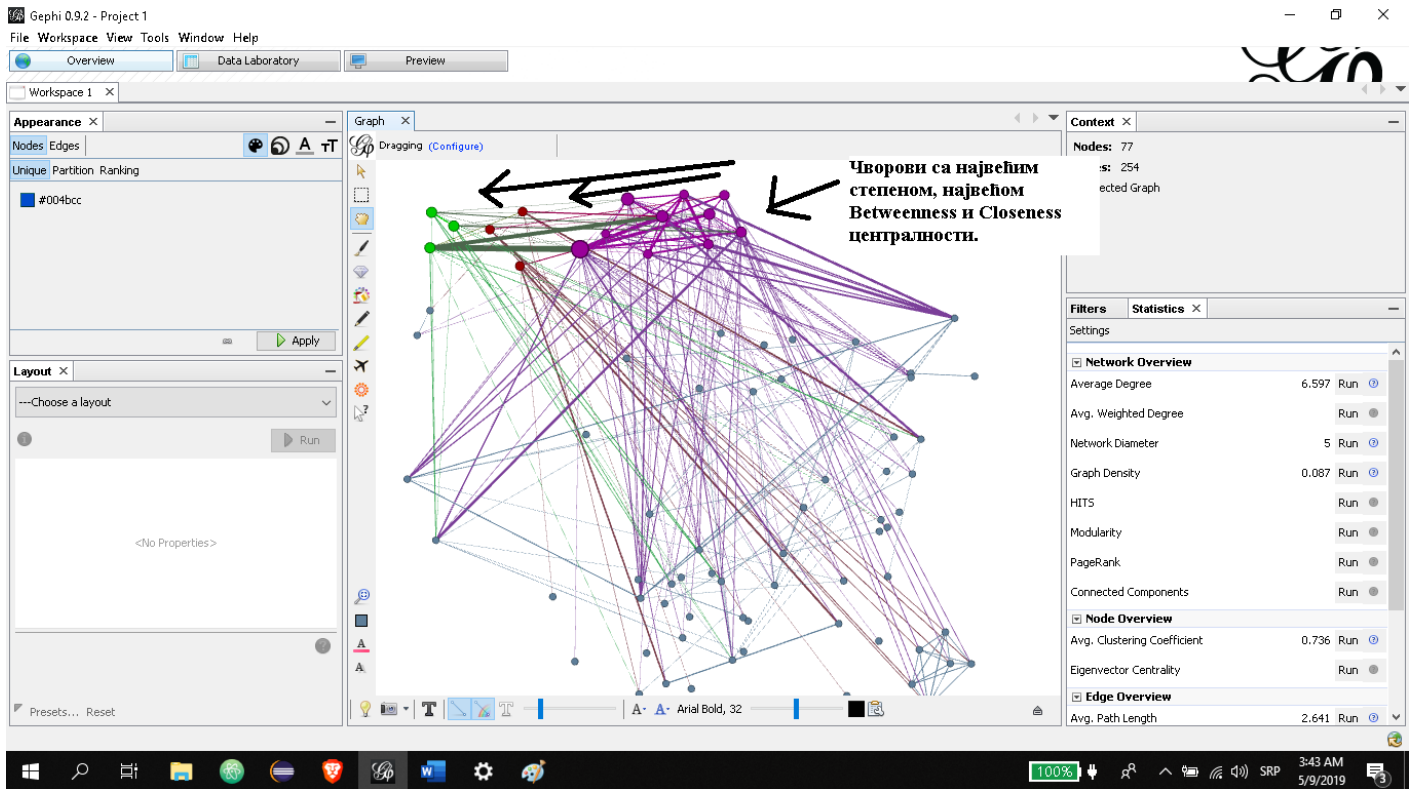
Слика 2.4

## ❖ Closeness

Табела топ десет чворова:

Ид	Лабела	Closeness
11	Valjean	0.6440677966101694
55	Marius	0.5314685314685315
25	Thenardier	0.5170068027210885
27	Javert	0.5170068027210885
48	Gavroche	0.5135135135135135
58	Enjolras	0.4810126582278481
26	Cosette	0.4779874213836478
64	Bossuet	0.475
68	Gueulemer	0.4634146341463415
69	Babet	0.4634146341463415

Поново вршимо измену визуализације графа (слика 2.5).



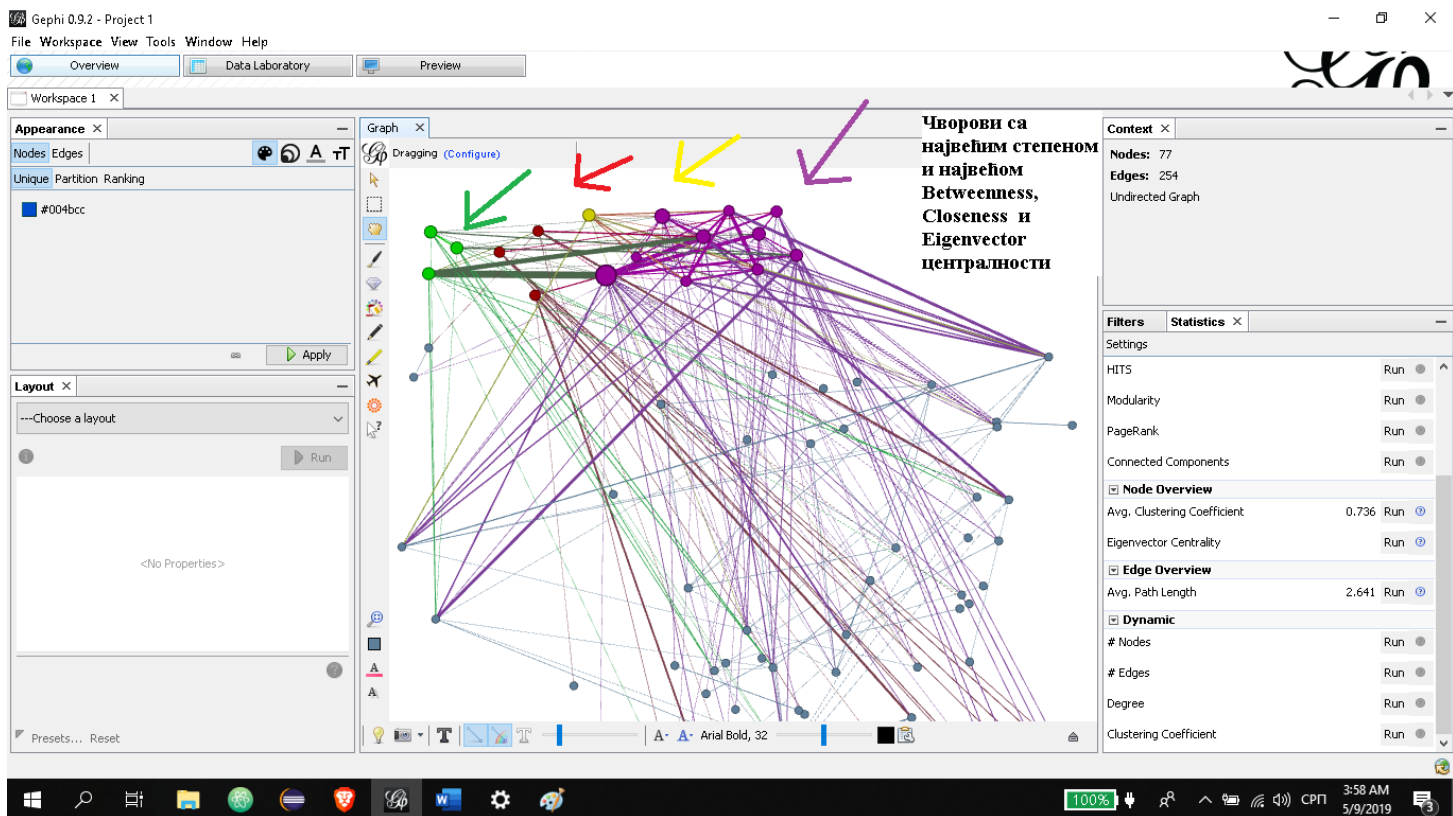
Слика 2.5

## ❖ Eigenvector

Табела топ десет чворова:

Ид	Лабела	Eigenvector
11	Valjean	1.0
48	Gavroche	0.9959418749764172
55	Marius	0.8289654718930208
58	Enjolras	0.816258816987052
64	Bossuet	0.7241400839567756
62	Courfeyrac	0.6803721982859918
27	Javert	0.6765355896405881
25	Thenardier	0.6764947218451572
65	Joly	0.6436060005440667
63	Bahorel	0.6436060005440666

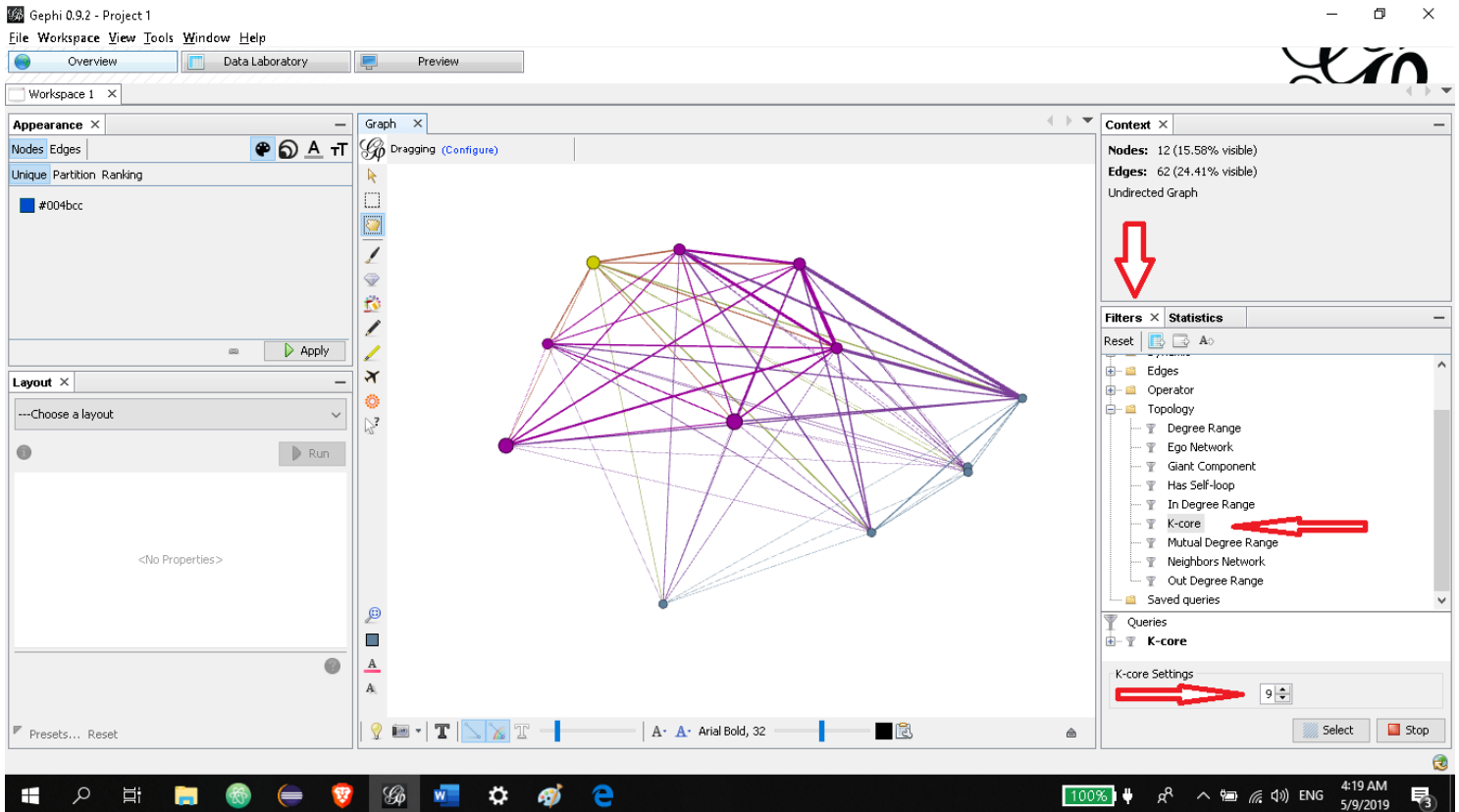
Вршимо измену визуализације графа (слика 2.6).



Слика 2.6

### 3. Чворови у максималаном кору

Унутар филтера постоји поље за филтрирање по кору (слика 3.1).



Слика 3.1

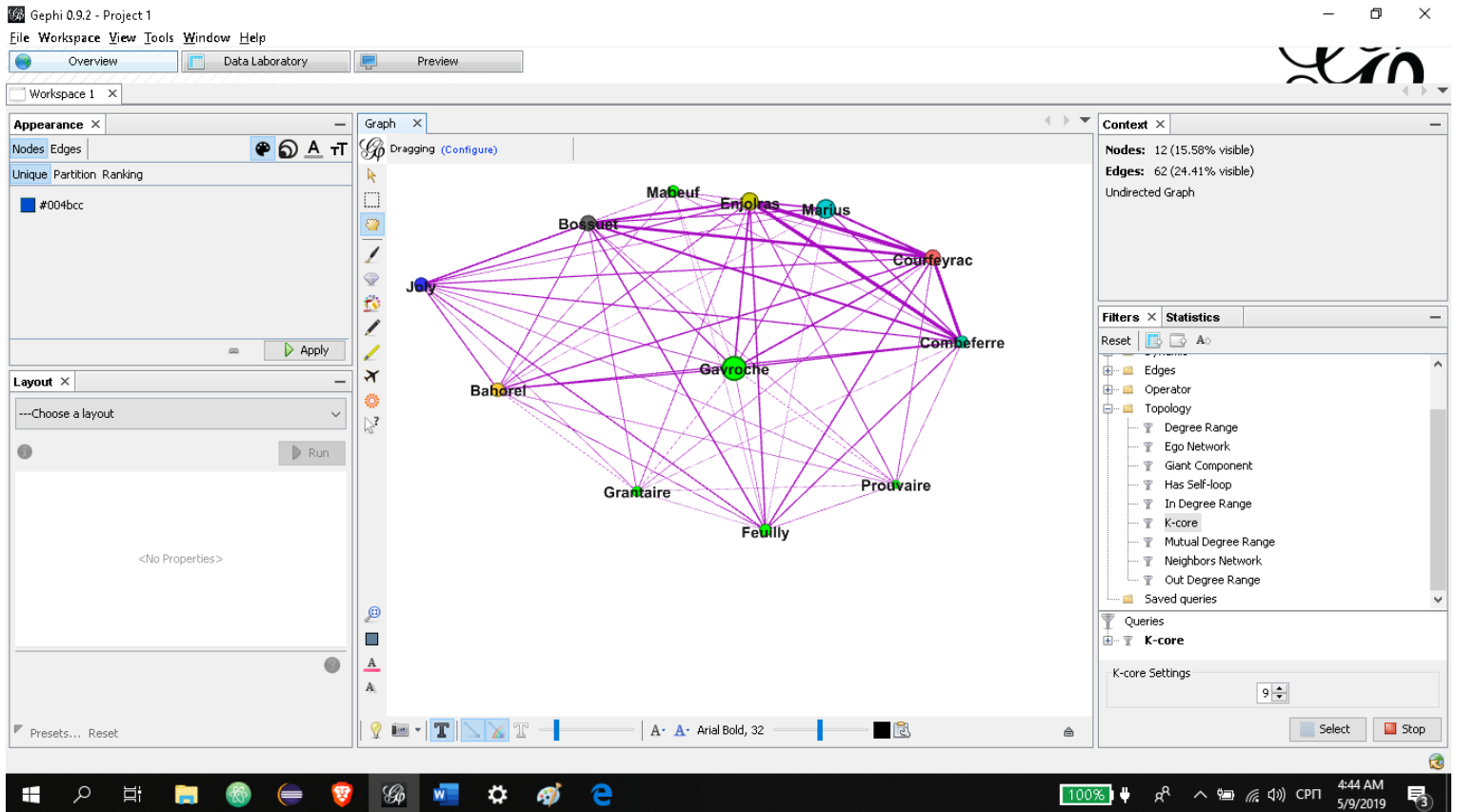
Максимални кор ове мреже је: 9

Списак чворова који припадају максималном кору (слика 3.2)

Визуелни приказ максималног кора (слика 3.3)

Id	Label
48	Gavroche
55	Marius
58	Enjolras
64	Bossuet
62	Courfeyrac
65	Joly
63	Bahorel
59	Combeferre
61	Feuilly
57	Mabeuf
66	Grantaire
60	Prouvaire

Слика 3.2



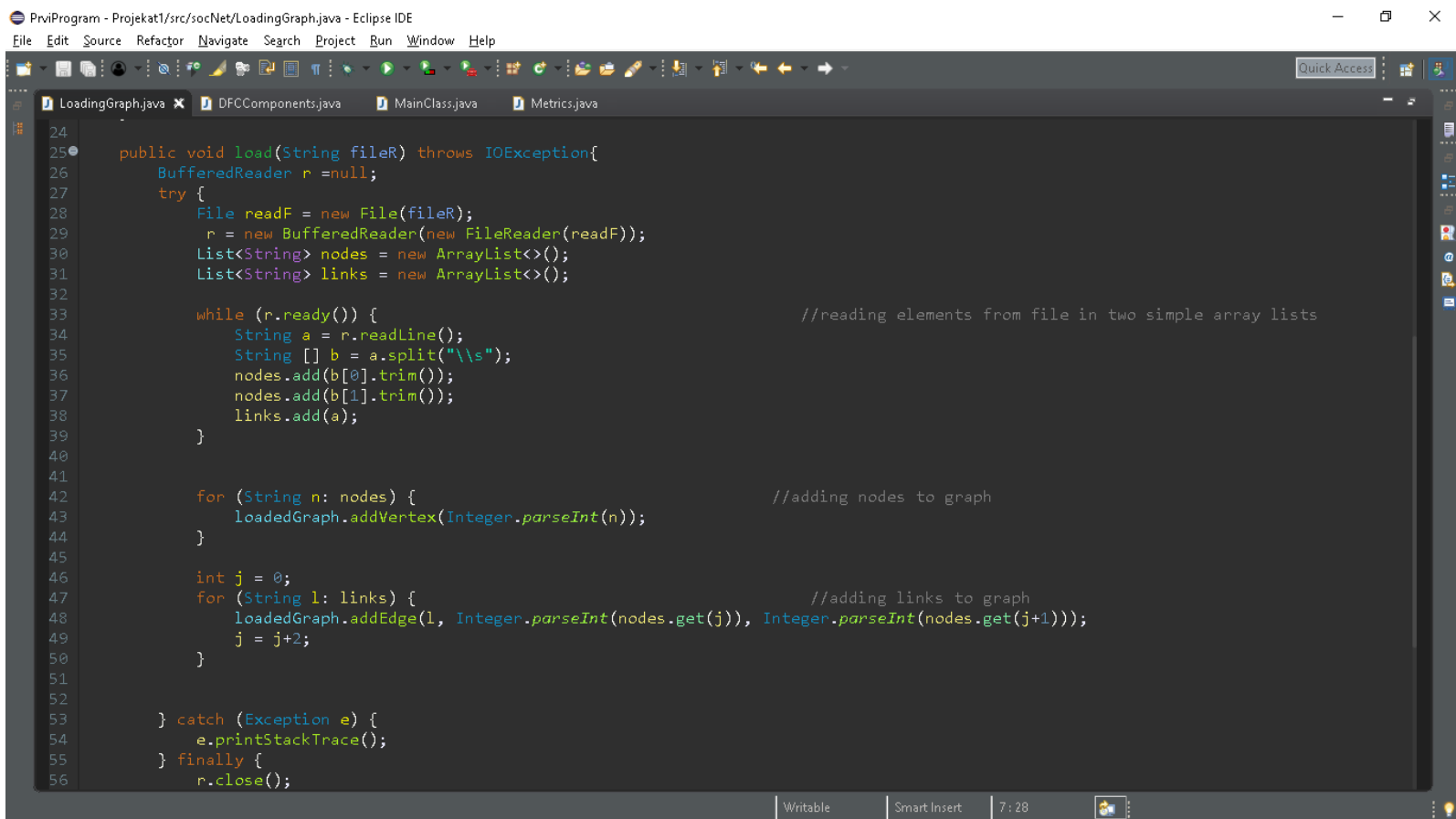
Слика 3.3



## 4. Програмски део задатка

- **Учитаваше и детектовање**

На самоме почетку креирана је класа под називом учитавање графа енг- (LoadingGraph). Приликом креирања инстанце класе за учитавање покреће се метода за учитавање која узиме информације из фајла датог у домаћем. Унутар саме методе креира се обичан ридер и две листа које примају линкове и чворове и распоређују их у неусмерен граф (*слика 4.1*)

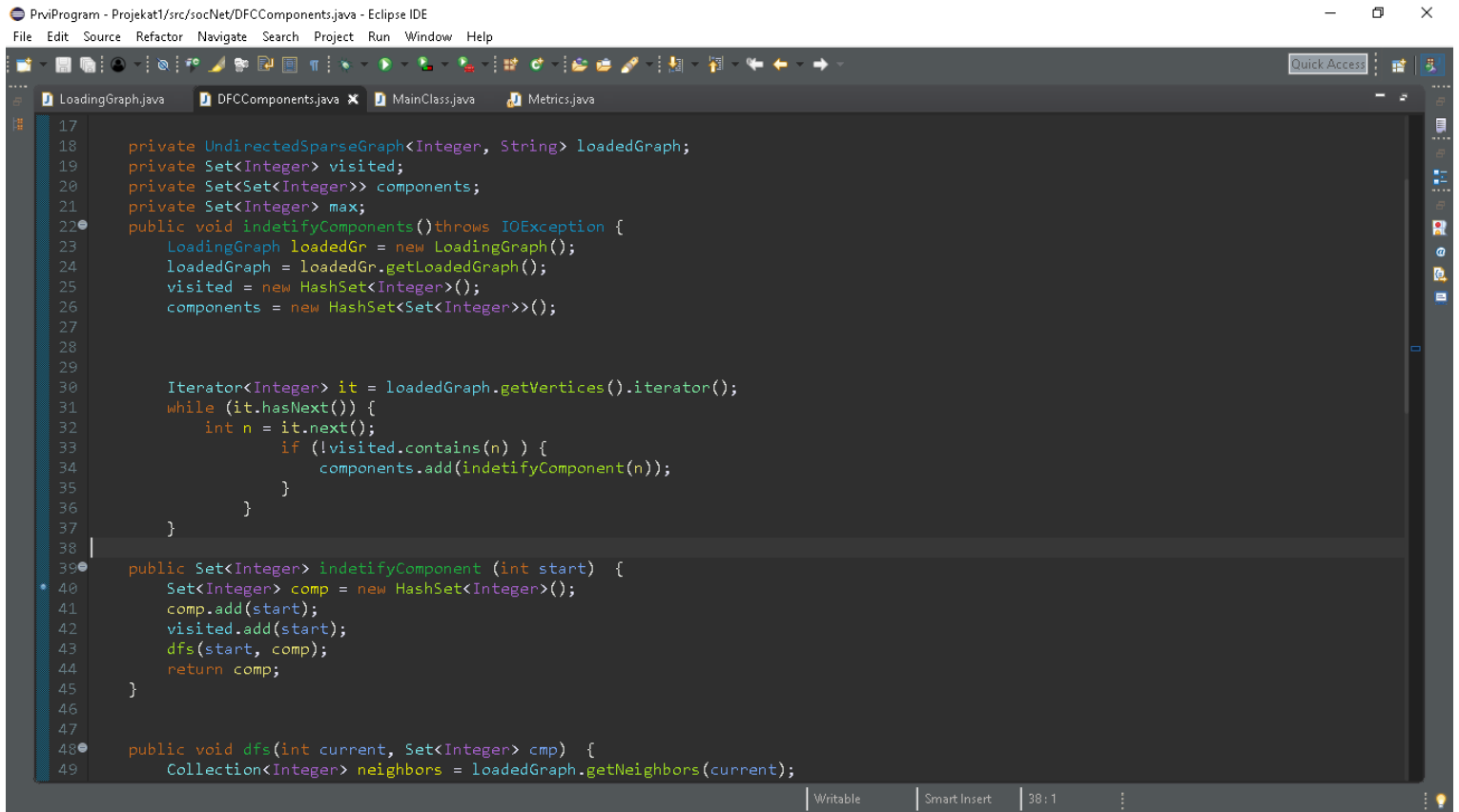


Слика 4.1

Након тога креирана је класа за пролазак кроз краф дфс алогоритмом и тражење гигантске компоненте. ДФс-ом се пролази графом у дубину, гледа се

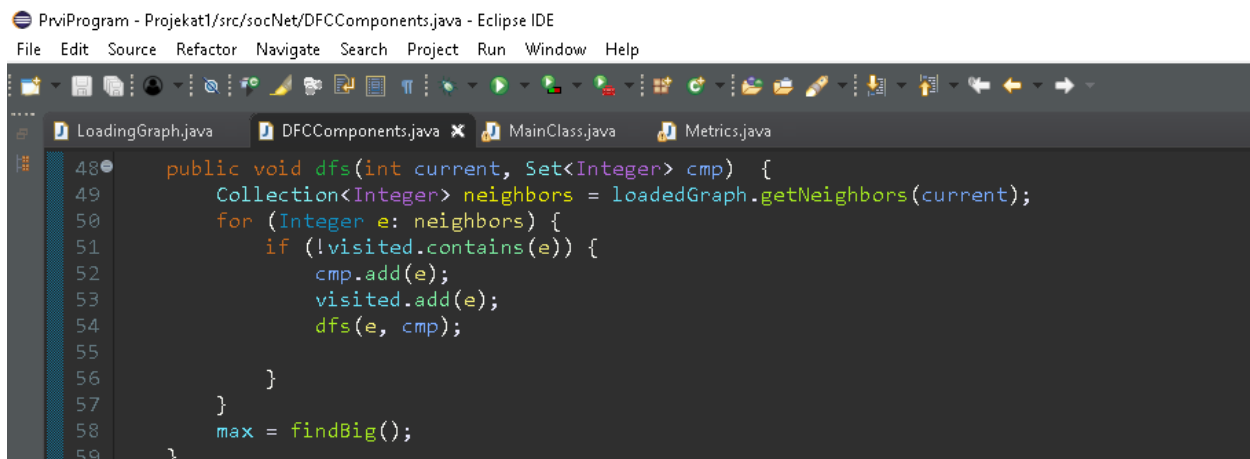


да ли смо на посећеном чвору , nakon тога гледамо све суседе тј. све чворове до којих можемо доћи од тога чвора.(слика 4.2 и слика 4.3)



```
17 private UndirectedSparseGraph<Integer, String> loadedGraph;
18 private Set<Integer> visited;
19 private Set<Set<Integer>> components;
20 private Set<Integer> max;
21 public void identifyComponents() throws IOException {
22     LoadingGraph loadedGr = new LoadingGraph();
23     loadedGraph = loadedGr.getLoadedGraph();
24     visited = new HashSet<Integer>();
25     components = new HashSet<Set<Integer>>();
26
27     Iterator<Integer> it = loadedGraph.getVertices().iterator();
28     while (it.hasNext()) {
29         int n = it.next();
30         if (!visited.contains(n)) {
31             components.add(identifyComponent(n));
32         }
33     }
34 }
35
36 public Set<Integer> identifyComponent (int start) {
37     Set<Integer> comp = new HashSet<Integer>();
38     comp.add(start);
39     visited.add(start);
40     dfs(start, comp);
41     return comp;
42 }
43
44 public void dfs(int current, Set<Integer> cmp) {
45     Collection<Integer> neighbors = loadedGraph.getNeighbors(current);
46     for (Integer e: neighbors) {
47         if (!visited.contains(e)) {
48             cmp.add(e);
49             visited.add(e);
50             dfs(e, cmp);
51         }
52     }
53     max = findBig();
54 }
```

Слика 4.2



```
48 public void dfs(int current, Set<Integer> cmp) {
49     Collection<Integer> neighbors = loadedGraph.getNeighbors(current);
50     for (Integer e: neighbors) {
51         if (!visited.contains(e)) {
52             cmp.add(e);
53             visited.add(e);
54             dfs(e, cmp);
55         }
56     }
57     max = findBig();
58 }
59 }
```

Слика 4.3

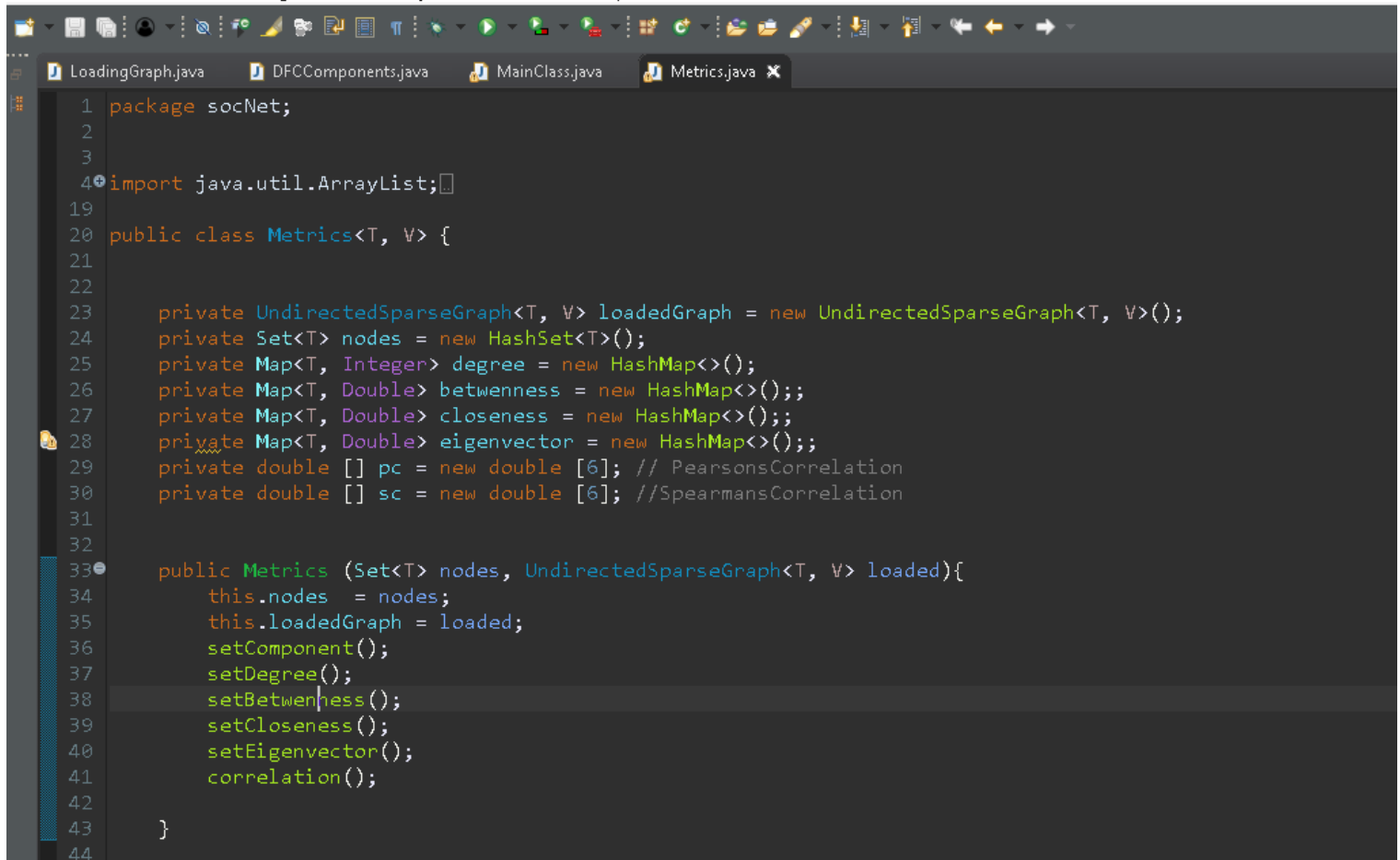
После креирања скупа компоненти правимо методу која само пролази кроз скуп и тражи онај највећи, тј. гигантск компоненту(*слика 4.4*)

```
60
61 public Set<Integer> findBig(){
62     Iterator<Set<Integer>> it = components.iterator();
63     if (components.isEmpty()) {
64         return Collections.emptySet();
65     }
66     Set<Integer> max1 = it.next();
67     while (it.hasNext()) {
68         Set<Integer> s = it.next();
69         if (s.size() > max1.size()) {
70             max1 = s;
71         }
72     }
73     return max1;
74 }
75
76
```

*Слика 4.4*

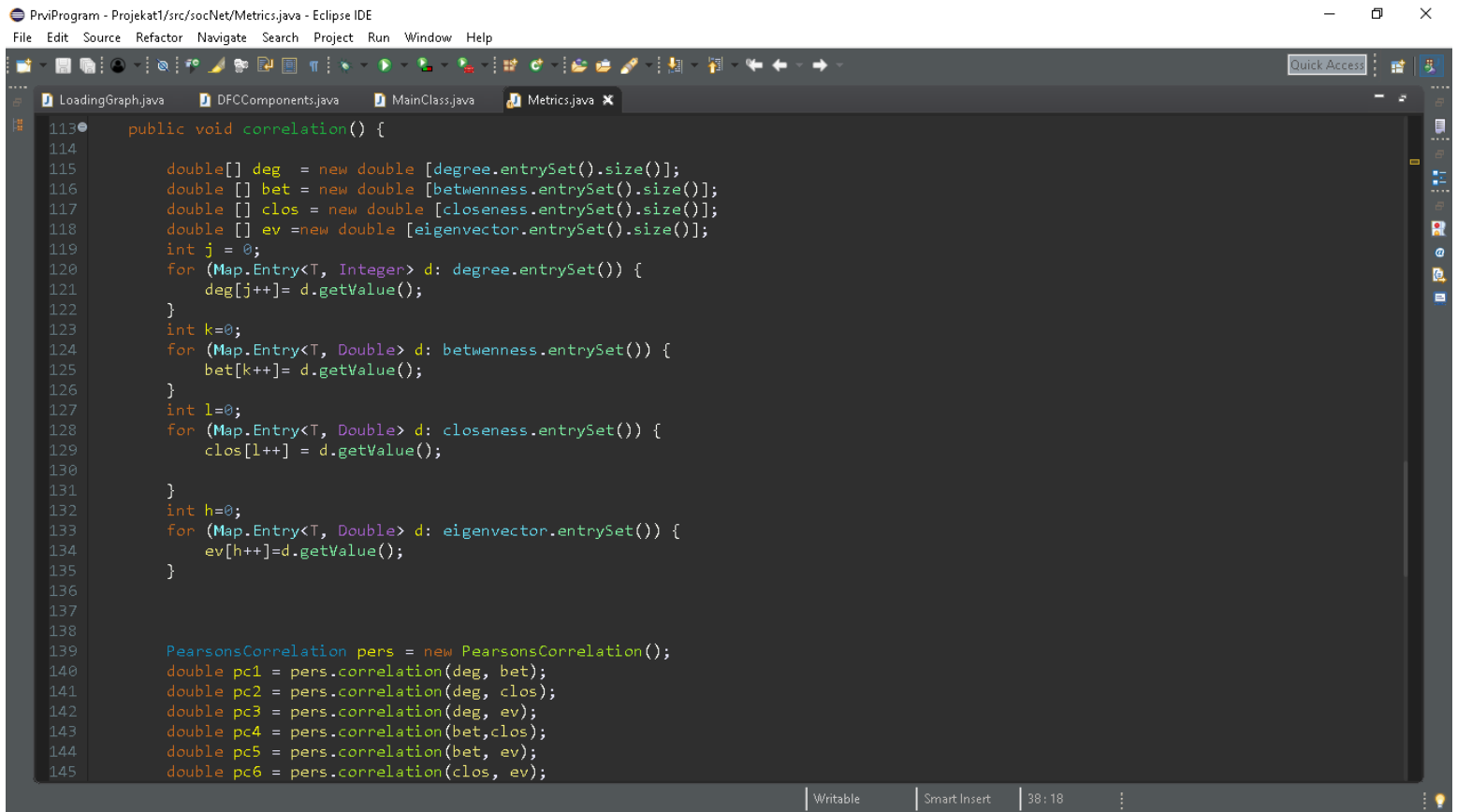
- **Метрике**

Класа метрике у себи садржи четири мапе, где су кључеви сами чворови а вредности њихове централности(*слика4.5*).



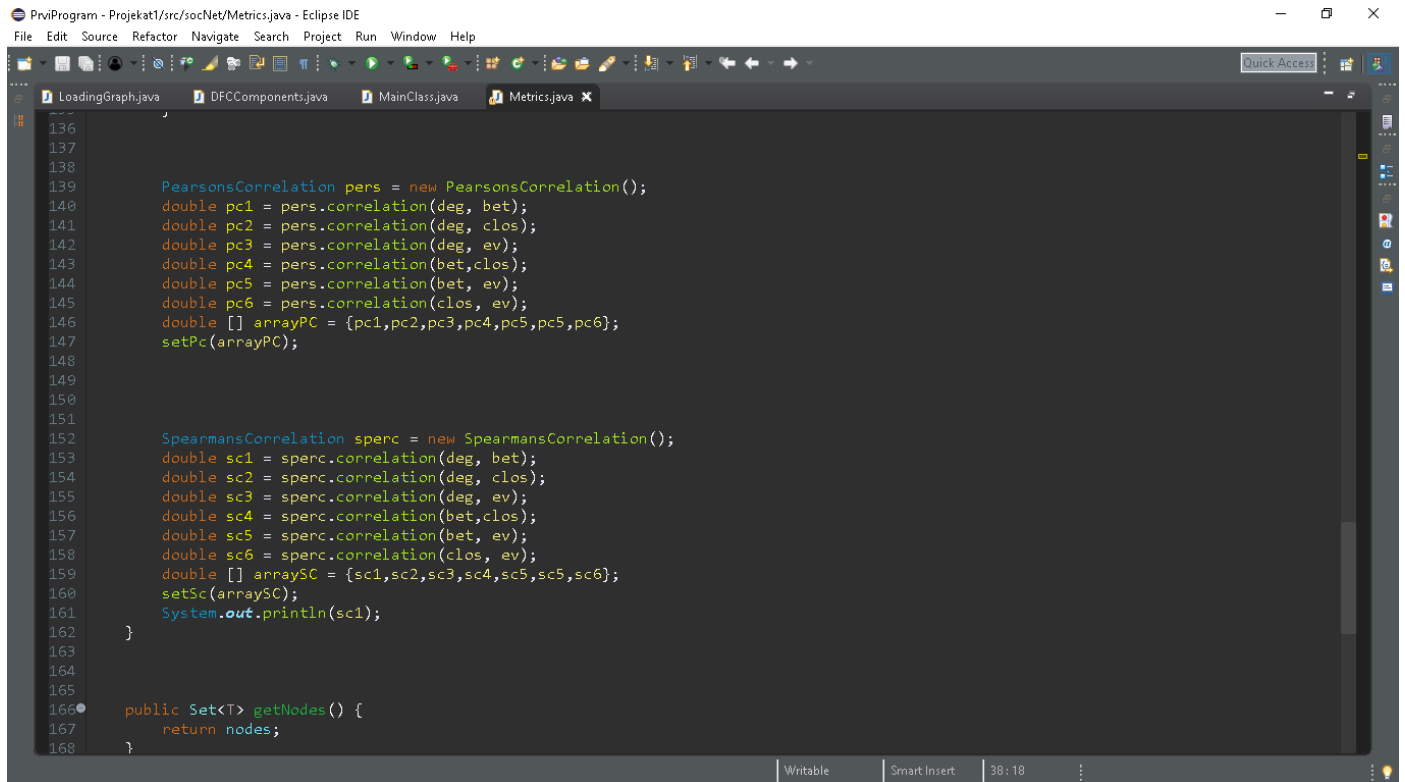
Слика 4.5

Такође у себи садржи сперманову и персонову колерацију за све метрике централности(слика 4.6 и слика 4.7). Приликом креирања инстанце класе Метрике (која је генеричка), прослеђујемо јој као параметар граф и гигантску компоненту са којом ћемо да радимо. Све методе се позивају приликом позива конструктора (слика 4.5).



```
113 public void correlation() {
114
115     double[] deg = new double [degree.entrySet().size()];
116     double [] bet = new double [betweenness.entrySet().size()];
117     double [] clos = new double [closeness.entrySet().size()];
118     double [] ev = new double [eigenvector.entrySet().size()];
119     int j = 0;
120     for (Map.Entry<T, Integer> d: degree.entrySet()) {
121         deg[j++] = d.getValue();
122     }
123     int k=0;
124     for (Map.Entry<T, Double> d: betweenness.entrySet()) {
125         bet[k++] = d.getValue();
126     }
127     int l=0;
128     for (Map.Entry<T, Double> d: closeness.entrySet()) {
129         clos[l++] = d.getValue();
130     }
131
132     int h=0;
133     for (Map.Entry<T, Double> d: eigenvector.entrySet()) {
134         ev[h++] = d.getValue();
135     }
136
137
138
139     PearsonCorrelation pers = new PearsonCorrelation();
140     double pc1 = pers.correlation(deg, bet);
141     double pc2 = pers.correlation(deg, clos);
142     double pc3 = pers.correlation(deg, ev);
143     double pc4 = pers.correlation(bet, clos);
144     double pc5 = pers.correlation(bet, ev);
145     double pc6 = pers.correlation(clos, ev);
```

Слика 4.6

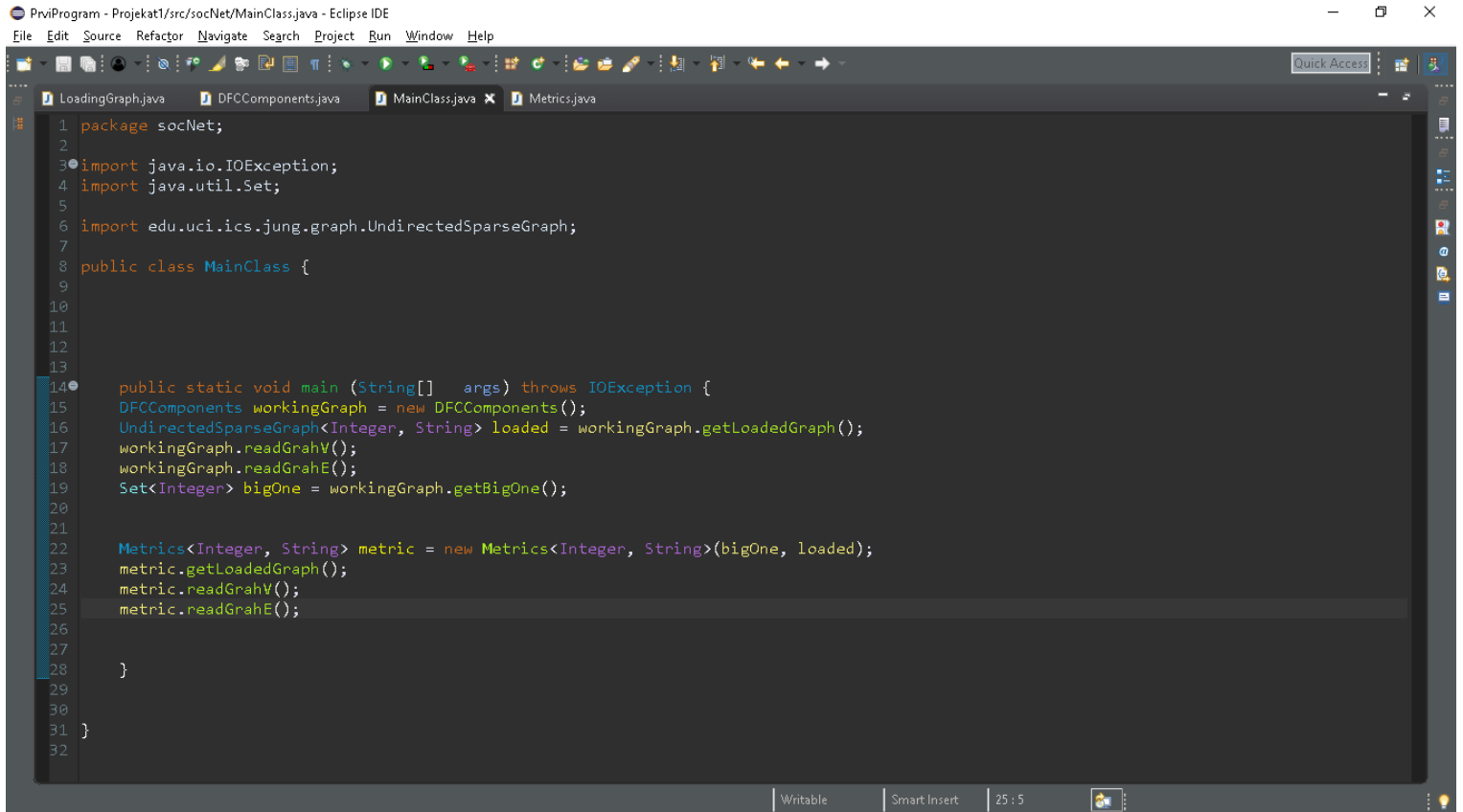


```
136
137
138
139     PearsonCorrelation pers = new PearsonCorrelation();
140     double pc1 = pers.correlation(deg, bet);
141     double pc2 = pers.correlation(deg, clos);
142     double pc3 = pers.correlation(deg, ev);
143     double pc4 = pers.correlation(bet, clos);
144     double pc5 = pers.correlation(bet, ev);
145     double pc6 = pers.correlation(clos, ev);
146     double [] arrayPC = {pc1, pc2, pc3, pc4, pc5, pc6};
147     setPc(arrayPC);
148
149
150
151
152     SpearmanCorrelation sperc = new SpearmanCorrelation();
153     double sc1 = sperc.correlation(deg, bet);
154     double sc2 = sperc.correlation(deg, clos);
155     double sc3 = sperc.correlation(deg, ev);
156     double sc4 = sperc.correlation(bet, clos);
157     double sc5 = sperc.correlation(bet, ev);
158     double sc6 = sperc.correlation(clos, ev);
159     double [] arraySC = {sc1, sc2, sc3, sc4, sc5, sc6};
160     setSc(arraySC);
161     System.out.println(sc1);
162 }
163
164
165
166 public Set<T> getNodes() {
167     return nodes;
168 }
```

Слика 4.7

- **Мејн класа**

Мејн класа служи за креирање инстанци потребних да претходно описане опције и методе одраде посао који желимо (*слика 4.8*)



*Слика 4.8*