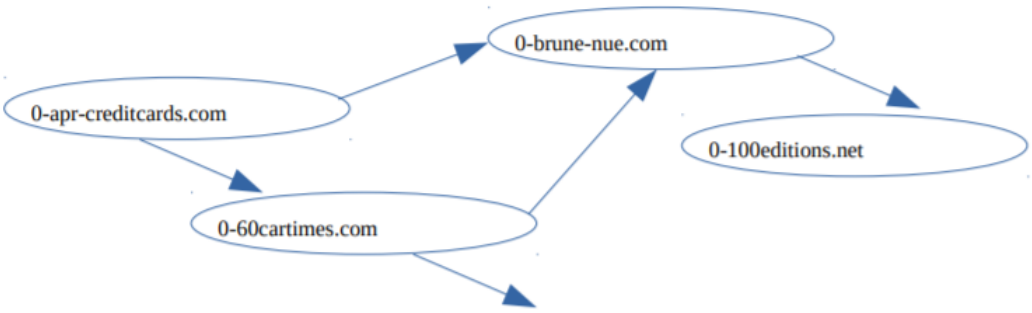
eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Web-a kudeatzeko aplikazioa (3. eginkizuna)



Egileak:
Aitor San José, Martin Amezola, Leire Garcia

Irakasgia:
Datu-Egiturak eta Algoritmoak

Irakasleak:
Iñigo Mendialdua eta Koldobika Gojenola

2. maila

46. taldea

2020.eko azaroaren 29

Aurkibidea

1 Sarrera eta arazoaren aurkezpena

Datu-Egiturak eta Algoritmoak ikasgaiako proiektua Web-orri kopuru handia kudeatuko dituen aplikazioa sortzea da.

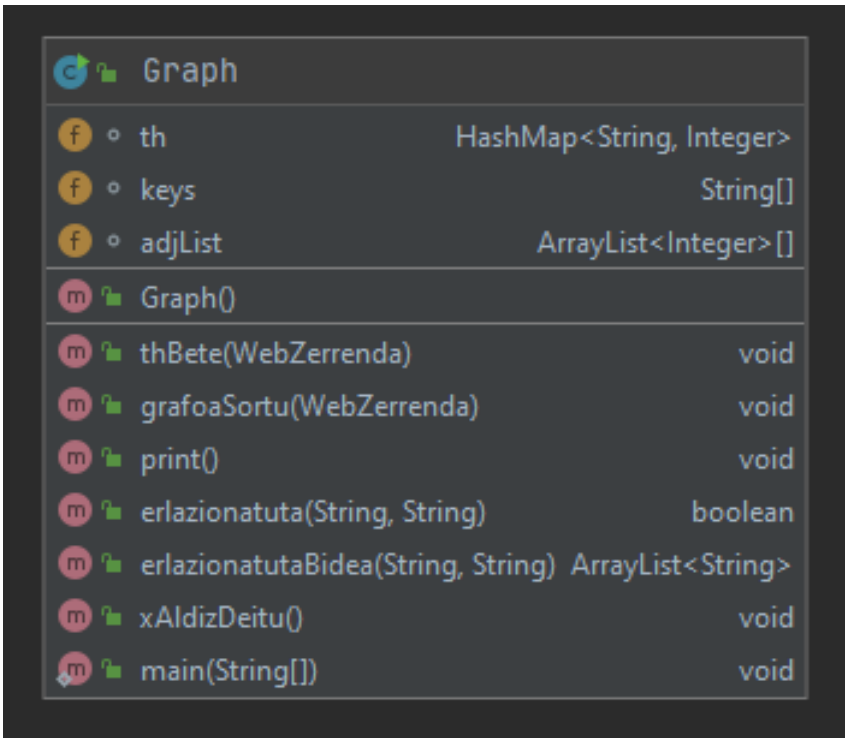
Ikasgai honetan, asko azpimarratzen da programaren kostua, beraz, inplementatzerako orduan datuen maneiatzearen arabera datu egitura bat edo bestea erabili dugu, inplementazioa gero eta eraginkorragoa izatearren.

Beraz, aurrean aipatutakoa argi ikusteko, zenbait eginkizun bete beharko ditugu lahuhilekoan zehar.

Hirugarren eginkizun honetan, web-orrien arteko loturak aztertuko ditugu. Ataza hori betetzeko, grafoak erabiliko ditugu. Aldi berean, bi web-orri erlazionatuta badauden esango digun programa bat garatuko dugu, baita bi web-orri erlazinatuta badaude orri batetik besterako lotura nolakoa den esango diguna (nodo bera iztekotan nodo bera dela, biderik ez badago biderik ez dagoela, eta bidea badago bidea esango duena). Amaitzeko denbora tarte jakin batean (minutu bat, ordu bat...) kalkulatu daitezkeen erlazio kopurua ere lortu dugu.

2 Diseinua

Hirugarren eginkizun honetan grafoekin lan egingo dugu (1. irudia) bai eta lehenengo eginkizuneko WebZerrenda klasearekin (bigarren honetan ez dugu aldaketarik egin, bakarrik datuak grafoan kargatzeko erabiltzen dugu).



1. irudia: Klase diagramaren diseinua.

3 Datu egituren deskribapena

Eginkizun honetan hainbat datu-egitura desberdin erabili ditugu, hala nola, *Grafo*-ak, *ArrayList*-ak, *HashMap*-ak, *Array*-ak, eta *Queue*-ak.

Grafo-a eginkizun honen oinarria da, eta egitura honen inguruan lan egingo dugu. *Grafo*-a egiteko ez dugu honen interfazea implementatu baizik eta 'eskuz' egin dugu. Gure egitura *Web*-en arteko esteken *ArrayList*-en *Array*-a eta *Array*-aren posizioak eta *Web*-en url-ak erlazionatzeko *Array* bat da. Hau sortzeko lehenengo eginkizunaren *WebZerrenda* erabili dugu *GrafoaSortu* metodoan.

ArrayList-ak eginkizun honetan hainbat aldiz erabili ditugu helburu desberdinekin. Lehen aipatu dudan bezala, *Grafo*-aren egituran erabili dugu; horretaz aparte, laguntzaile moduan erabili dugu *Erlazionatuta* metodoan eta emaitza gisa bueltatzen da *ErlazionatutaBidea* metodoan.

HashMap-aren kasuan, lehenengo eginkizuneko *WebMapa* erabiltzen dugu *Grafo*-a sortzerakoan, handik *Web*-en informazioa hartuz eta beste berri bat laguntzaile moduan sortuz *Web*-en url eta indizea erlazionatzeko.

Array-ak gordetako informazioa eskuratzeko kostu handirik ez duten egiturak direnez, asko erabili ditugu gure programan. Lehen esan bezala, *Grafo*-a sortzean erabili ditugu bai *Web* bakoitzaren estekak gordetzeko eta baita hauen url-ak eta posizioak erlazionatzeko. Horretaz aparte, laguntzaile moduan erabili ditugu *Erlazionatuta* eta *ErlazionatutaBidea* metodoetan *Web* bakoitza aztertu den edo ez gordetzeko.

Azkenik, *Queue*-ak oso erabilgarriak izan dira *Grafo*-a aztertzerako orduan. Bai *Erlazionatuta* eta *ErlazionatutaBidea* metodoetan erabili ditugun aztertu gabe zeuden *Web*-ak gordetzeko. Hauek inplementatzeko *LinkedList*-ak erabili ditugu.

4 Metodo nagusien diseinu eta inplementazioa

4.1 HashMap- a bete: *thBete (WebZerrenda lista)*

public void thBete (WebZerrenda lista)
// Aurrebaldintza: WebZerrenda motatako objektua jasotzen du (lista).
// Postbaldintza: WebZerrendako web guztiak th HashMapa datu egituran sartuta daude <Url, Id> formatuan.

- Proba kasuak:
  1. th bete da modu egokian.
  2. th Hash-Mapa betetzean errore bat egon da.
- Algoritmoa:

```
1   WebZerrenda motatako lista, .values() erabiliz Collections batean sartu;
2
3   collections hori arraylist batean bihurtu (l);
4
5   th hash map berria sortu;
6
7   i indize bat sortu;
8           for (arraylistaren web guztiak){
9                   sartu th hashmapean (eguneko webaren url-a, indizea);
10                  indizea eguneratu (gehitu);
11          }
```

- Kostua: *O(n)* non:
  n = l ArrayList-aren luzeera edo web kopurua.

4.2 Grafoaren sorketa: *grafoaSortu (WebZerrenda lista)*

public void grafoaSortu (WebZerrenda lista)
// Aurrebaldintza: WebZerrenda motatako objektua jasotzen du.
// Postbaldintza: "th"HashMapa beteta dago web orriekin, "keys"Array-a beteta dago web orrien url-ekin eta "AdjList"ArrayList-en Array-a bete da.

- Proba kasuak:
  1. Grafoa sortu da.
  2. Grafoa sortzean errore bat egon da.
- Algoritmoa:

```
1
2   // 1. pausua:     th    bete
3       bete th hash mapa;
4
5       // 2. pausua: keys bete
6       keys sortu th hash maparen tamainuarekin;
7
8
9       for (th hash maparen string osoak)
10      {
11          keys[hash maparen dagoeneko string-aren indizea] = hash maparen string-a
                ↪ ;
```

```
12          }
13
14          // 3. pausua: adjList bete
15          adjList sortzen dugu keys-ren luzeerarekin;
16          web objektu bat sortu (w) null balioarekin;
17          web-en arraylist auxiliar bat sortzen dugu (aux);
18          kontagailua 0-z hasi;
19          for(keys-ren string guztiak){
20              w da WebZerrenda motatako lista-n bilatuUrl(keys-ren string-a) egin eta
                    ↪ gero bueltatzen duen weba;
21              aux = web horren esteken zerrenda;
22              berria izeneko arraylist-a sortzen da (integerrekoa);
23          for(aux arraylistaren web guztiak){
24              sartu berria arraylist-ean(th HM-an dagoen value sartzean w.getUrl()
                    ↪ );
25          }
26          adjList[kontagailua] = berria arraylist-a;
27          kontagailua eguneratu;
28          }
```

- Kostua: *O(n + n*m)* non:
  n = WebZerrendako Hash-Maparen luzeera = th-ren luzeera.
  m = web bat duen esteka kopuruaren batazbestekoa.

4.3 Bi nodoen arteko bidea badago: *erlazionatuta(Sting a1, String a2)*

public boolean erlazionatuta(Sting a1, String a2)
// Aurrebaldintza: Bi string jasotzen dira (bi url, konkretuki) eta hauek grafoan daude.
// Postbaldintza: true bi web horien artean bide bat badago (konektatuta badaude) bestela, false.

- Proba Kasuak:

  1. Sartutako bi nodoak nodo bera dira: True
  2. Sartutako nodoak desberdinak dira eta erlazionatuta daude: True
  3. Sartutako nodoak desberdinak dira eta ez daude erlazionatuta: False

- Algoritmoa:

```
1     aztertuGabeak linkedlist (ilara) sortu;
2     pos1 integerra a1-k duen indizearekin (th hashmapan value);
3     pos2 integerra a2-k duen indizearekin (th hashmapan dagoen value);
4     aurkitua boolearra false-n hasieratu;
5     aztertuak boolearren array-a sortu th hashmaparen tamainarekin;
6
7         for(aztertuak arraya errekorritu)
8         {
9             aztertuak[eguneko momentua] = false;
10        }
11
12        aztertuGabeak ilaran gehitu pos1; //aztertuGabeen ilararen bukaeran sartu
13                aztertuak arrayean pos1 indizean true ipini; //aztertuen array-an
                      ↪ true jarri
14                baldin (pos1 eta pos2-ren berdina bada) orduan aurkitua true ipini;
15
```

```
16
17              int unekoa pos1 balioa eman;
18              ArrayList laguntzaile bat sortu, integerrekoa (lag);
19
20              bitartean(ez aukitua && aztertuGabeak ilara ez hutsa){
21                      kendu aztertuGabeak-ko ilarako lehen elementua eta gorde
                            ↪ unekoan;
22                      lag arrayean sartzen ditugu adjList[unekoa]; // Sartzen dugu
                            ↪  uneko nodoaren bizilagunak
23                      //aztertuak arrayan true sartuko dugu nodoen indizeetan,
                            ↪ baita aztertu gabeen ilaran sartu ere:
24                      for (lag arraylist-aren elementu guztiak){
25                          baldin (ez bada tratatua izan, bizilagun hori) {
26                                  aztertuen array-ean true balioa eman;
27                                  sartu aztertuGabeak ilaran;
28                          }
29                          baldin (bizilagun hori pos2 bada){ //amaierako nodoa
                                ↪  unekoaren bizilagunen bat bada, aurkitua
                                ↪ true eta loop-a amaitu
30                              aurkitua true balioa eman;
31                          }
32                      }
33              }
34          bueltatu aurkitua;
```

- Kostua: *O(n + (a*b))*

  n = Aztertuak array-aren luzeera da.

  a = aztertuGabeak ilararen luzeeraren batazbestekoa

  b = web bat dituen bizilagunen batazbeztekoa

4.4 Bi web-en arteko bidea bueltatu: erlazionatutaBidea(String a1, String a2)

public ArrayList<String> erlazionatutaBidea(Sting a1, String a2)
// Aurrebaldintza: Bi string jasotzen dira (bi url, konkretuki) eta hauek grafoan daude.
// Postbaldintza: Web horien artean dagoen bidea bueltatzen du, bide hori badago. Bestela esaten du bide hori ez dela existitzen.

- Proba Kasuak:

  1. Sartutako bi nodoak nodo bera dira: 'Hasierako eta amaierko nodoak berdinak dira'
     Bidea: <a>
  2. Sartutako nodoak desberdinak dira eta erlazionatuta daude:
     Bidea: <a,b,d,e,h>
  3. Sartutako nodoak desberdinak dira eta ez daude erlazionatuta: 'Ez dago biderik'
     Bidea: null

- Algoritmoa:

```
1
2     ema izeneko ArrayList-a sortu;
3     aztertuGabeak linkedlist (ilara) sortu;
4     bidea Array-a sortu th hasmaparen tamainarekin;
5     aurkitua boolearra sortu eta false-n hasieratu;
```

```
6       pos1 integerra a1-k duen indizearekin (th hashmapan value);
7       pos2 integerra a2-k duen indizearekin (th hashmapan dagoen value);
8
9       aztertuak boolearren array-a sortu th hashmaparen tamainarekin;
10
11          for(aztertuak arraya errekorritu)
12          {
13              aztertuak[eguneko momentua] = false;
14          }
15
16          aztertuGabeak ilaran gehitu pos1; //aztertuGabeen ilararen bukaeran sartu
17                  aztertuak arrayean pos1 indizean true ipini; //aztertuen array-an
                        ↪ true jarri
18                  baldin (pos1 eta pos2-ren berdina bada) orduan aurkitua true ipini;
19
20
21                  int unekoa pos1 balioa eman;
22                  ArrayList laguntzaile bat sortu, integerrekoa (lag);
23
24                  bitartean(ez aukitua && aztertuGabeak ilara ez hutsa){
25                          kendu aztertuGabeak-ko ilarako lehen elementua eta gorde
                                ↪ unekoan;
26                          lag arrayean sartzen ditugu adjList[unekoa]; // Sartzen dugu
                                ↪  uneko nodoaren bizilagunak
27                          //aztertuak arrayan true sartuko dugu nodoen indizeetan,
                                ↪ baita aztertu gabeen ilaran sartu ere:
28                          for (lag arraylist-aren elementu guztiak){
29                                  baldin (ez bada tratatua izan, bizilagun hori) {
30                                          aztertuen array-ean true balioa eman;
31                                          sartu aztertuGabeak ilaran;
32                                          bidea array-ean sartzen dugu unekoa;
33                                  }
34                                  baldin (bizilagun hori pos2 bada){ //amaierako nodoa
                                        ↪  unekoaren bizilagunen bat bada, aurkitua
                                        ↪ true eta loop-a amaitu
35                                          aurkitua true balioa eman;
36                                  }
37                          }
38
39                          baldin (aurkitua true da){
40                              baldin(pos1 eta pos2 berdinak dira){
41                                      ema arraylistean gehitu keys[pos1] // pos1 duen
                                            ↪ url-a.
42                                      Printeatu("Hasierako eta amaierko nodoak
                                            ↪ berdinak dira ");
43                              }
44                          bestela{
45                                  sortu pila bat pila izenekoa;
46                                  oraingoa integer bat sortu pos2 balioarekin;
47                                  sartu pilan keys[oraingoa] // oraingoaren url-a
48
49                                  bitartean (oraingoa eta pos1 ezberdinak){
50                                      oraingoa da bidea[oraingoa];
51                                      sartu pilan keys[oraingoa];
52                                  }
```

```
53                                  Printeatu("Bidea hau da: ");
54                                  while(pila ez hutsa den bitartean){
55                                          Printeatu(pilaren goiko elementua);
56                                          ema-n gehitu pilako goiko elementua eta
                                                ↪ pilatik elementu hori kendu;
57                                  }
58                          }
59                  }
60                  bestela{
61                          Printeatu("Ez dago biderik");
62                  }
63                  bueltatu ema;
64                  }
```

- Kostua: *O(n + (a*b) + 2p)* non:
  n = Aztertuak array-aren luzeera da.
  a = aztertuGabeak ilararen luzeeraren batazbestekoa
  b = web bat dituen bizilagunen batazbeztekoa
  p = bidearen luzeera.

5 Kodea

5.1 Hirugarren eginkizuneko kodea

```
1  package packlEnuntziatu3;
2  import packlEnuntziatu1.Web;
3  import packlEnuntziatu1.WebZerrenda;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.text.DecimalFormat;
7  import java.text.NumberFormat;
8  import java.util.*;
9
10 public class Graph {
11       HashMap<String, Integer> th; //HashMap, non Key = url eta Value = indizea
12       String[] keys;
13       ArrayList<Integer>[] adjList;
14
15         public Graph() {}
16
17         public void thBete (WebZerrenda lista){
18                 Collection<Web> c= lista.getHM().values();
19                 ArrayList<Web> l= new ArrayList<>(c);
20                 this.th = new HashMap<>();
21                 int i =0;
22                 for (Web w :l){
23                         this.th.put(w.getUrlWeb(),i);
24                         i++;
25                   }
26           }
27
28           public void grafoaSortu(WebZerrenda lista){
29                 // Post: web-en zerrendatik grafoa sortu, Nodoak web-en url-ak dira
30             // 1. pausua:     th    bete
31                   this.thBete(lista);
32             // 2. pausua:    keys    bete
33                 keys = new String[th.size()];
34                 for (String k: th.keySet()) keys[th.get(k)] = k;
35             // 3. pausua:    adjList    bete
36                   adjList = new ArrayList[keys.length];
37                   Web w= null;
38                   ArrayList<Web> aux = new ArrayList<>();
39                   int kont=0;
40                   for(String s: keys){
41                         w = lista.bilatuUrl(s);
42                         aux = w.getEstekenZerrenda();
43                         ArrayList<Integer> berria = new ArrayList<>();
44                         for(Web w1 : aux){
45                                 berria.add(th.get(w1.getUrlWeb()));
46                         }
47                         adjList[kont]=berria;
48                         kont++;
49                   }
50         }
51
```

```
52      public void print(){
53          for (int i = 0; i < adjList.length; i++){
54              System.out.print("Element: " + i + " " + keys[i] + " --> ");
55              for (int k: adjList[i])  System.out.print(keys[k] + " ### ");
56
57              System.out.println();
58          }
59      }
60
61        public boolean erlazionatuta(String a1, String a2){
62                Queue<Integer> aztertuGabeak = new LinkedList<>();
63                int pos1 = th.get(a1);
64                int pos2 = th.get(a2);
65                boolean aurkitua = false;
66                boolean[] aztertuak = new boolean[th.size()];
67                for(int k=0; k<aztertuak.length;k++) aztertuak[k] = false;
68                aztertuGabeak.add(pos1); //aztertuGabeen ilararen bukaeran sartu
69                aztertuak[pos1]= true; //aztertuen array-an true jarri
70                if (pos1 == pos2) aurkitua = true;
71                int unekoa= pos1;
72                ArrayList<Integer> lag = null;
73                while(!aurkitua && !aztertuGabeak.isEmpty()){
74                        //aurkitu ez dugun bitartean edo pila hutsa den bitartean loop-
                              ↪ ean sartu
75                        unekoa= aztertuGabeak.remove();
76                        lag = adjList[unekoa]; //aztertzen ari garen nodoaren
                              ↪ bizilagunak lortu
77                        //aztertuak arrayan true sartuko dugu nodoen indizeetan, baita
                              ↪ aztertu gabeen ilaran sartu ere:
78                        for (int biziLag : lag){
79                            if (!aztertuak[biziLag]) {//ez bada tratatua izan
80                                    aztertuak[biziLag] = true;
81                                    aztertuGabeak.add(biziLag);
82                            }
83                            if(biziLag==pos2){ //amaierako nodoa unekoaren
                                  ↪ bizilagunen bat bada, aurkitua eta loop handia
                                  ↪ amaitu
84                                    aurkitua= true;
85                            }
86                        }
87                }
88                return aurkitua;
89        }
90
91        public  ArrayList<String> erlazionatutaBidea(String a1, String a2){
92                ArrayList<String> ema= new  ArrayList<String>();
93                Queue<Integer> aztertuGabeak = new LinkedList<>();
94                Integer[] bidea = new Integer[th.size()];
95                int pos1 = th.get(a1);
96                int pos2 = th.get(a2);
97                boolean aurkitua = false;
98                boolean[] aztertuak = new boolean[th.size()];
99                for(int k=0; k<aztertuak.length;k++){
100                       aztertuak[k]=false;
101               }
```

```
102                 aztertuGabeak.add(pos1); //aztertuGabeen ilararen bukaeran sartu
103                 aztertuak[pos1]= true; //aztertuen array-an true jarri
104                 if (pos1 == pos2) aurkitua = true;
105                 int unekoa= pos1;
106                 ArrayList<Integer> lag = null;
107                 while(!aurkitua && !aztertuGabeak.isEmpty()){
108                         //aurkitu ez dugun bitartean edo pila hutsa den bitartean loop-
                            ↪ ean sartu
109                         unekoa= aztertuGabeak.remove();
110                         lag = adjList[unekoa]; //aztertzen ari garen nodoaren
                            ↪ bizilagunak lortu ditugu
111                         //aztertuak arrayan true sartuko dugu nodoen indizeetan, baita
                            ↪ aztertu gabeen ilaran sartu ere:
112                         for (int biziLag : lag){
113                                 //bidean jarri ze nodotik heldu garen
114                                 if (!aztertuak[biziLag]) {//ez bada tratatua izan
115                                         aztertuak[biziLag] = true;
116                                         aztertuGabeak.add(biziLag);
117                                         bidea[biziLag]=unekoa;
118                                 }
119                                 if(biziLag==pos2){ //amaierako nodoa unekoaren
                                    ↪ bizilagunen bat bada, aurkitua eta loop handia
                                    ↪ amaitu
120                                         aurkitua= true;
121                                 }
122                         }
123                 }
124                 if (aurkitua){
125                         if(pos1==pos2){//nodo bera badira
126                                 ema.add(keys[pos1]);
127                                 System.out.println("Hasierako eta amaierko nodoak
                                    ↪ berdinak dira ");
128                         }
129                         else{//nodo desberdinak badira
130                                 Stack<String> pila= new Stack<String>();
131                                 int oraingoa=pos2;
132                                 pila.push(keys[oraingoa]);
133                                 while(oraingoa!=pos1){
134                                         oraingoa= bidea[oraingoa];
135                                         pila.push(keys[oraingoa]);
136                                 }
137                                 System.out.println("Bidea hau da: ");
138                                 while(!pila.isEmpty()){
139                                         System.out.println(pila.peek());
140                                         ema.add(pila.pop());
141                                 }
142                         }
143                 }
144                 else{//biderik ez badago
145                         System.out.println("Ez dago biderik");
146                 }
147                 return ema;
148         }
149
150
```

```
151         public void xAldizDeitu(){
152                 Random r= new Random();
153                 int random1= 0;
154                 int random2= 0;
155                 long start = System.currentTimeMillis();
156                 for (int i=0; i<100; i++){
157                         random1= r.nextInt(keys.length-1);
158                         random2= r.nextInt(keys.length-1);
159                         erlazionatuta(keys[random1],keys[random2]);
160                 }
161                 long end = System.currentTimeMillis();
162                 NumberFormat formatter = new DecimalFormat("#0.00000");
163                 System.out.print("Execution time is " + formatter.format((end - start) /
                        ↪ 1000d) + " seconds");
164         }
165 
166         public static void main(String[] args) {
167                 WebZerrenda w = WebZerrenda.getNireWebZerrenda();
168                 File webIndexFitxeroa = null;
169                 File webEstekaFitxeroa = null;
170                 webIndexFitxeroa = new File ("resources\\index.txt");
171                 webEstekaFitxeroa = new File ("resources\\pld-arcs-1-N.txt");
172                 try {
173                         w.indexFitxeroaKargatu(webIndexFitxeroa);
174                         w.arcFitxeroaKargatu(webEstekaFitxeroa);
175                 } catch (FileNotFoundException e) {
176                         e.printStackTrace();
177                 }
178                 Graph g = new Graph();
179                 g.grafoaSortu(w);
180                 System.out.println("Erlazionatuta:");
181                 System.out.println(g.erlazionatuta(g.keys[10], g.keys[0]));
182                 System.out.println(" ");
183                 System.out.println("ErlazionatutaBidea biderik gabe:");
184                 g.erlazionatutaBidea("0-5.co.il", g.keys[0]);
185                 System.out.println(" ");
186                 System.out.println("ErlazionatutaBidea bide zuzenarekin:");
187                 g.erlazionatutaBidea("0-chat.com", "0-deai.com");
188                 System.out.println(" ");
189                 System.out.println("Erlazionatuta bidea random:");
190                 g.erlazionatutaBidea(g.keys[20], g.keys[10]);
191                 System.out.println(" ");
192                 System.out.println("Erlazionatuta bidea nodo berarekin");
193                 g.erlazionatutaBidea(g.keys[20],g.keys[20]);
194                 System.out.println(" ");
195                 System.out.println("Erlazionatuta bidea random:");
196                 g.erlazionatutaBidea(g.keys[234565], g.keys[17660]);
197                 System.out.println(" ");
198                 System.out.println("Erlazionatuta 100 aldiz exekutatzeko behar duen
                        ↪ denbora");
199                 g.xAldizDeitu();
200         }
201 
202 }
```

6 JUnit

```java
package packlEnuntziatu3;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import packlEnuntziatu1.Web;
import packlEnuntziatu1.WebZerrenda;

public class GraphTest {
    private WebZerrenda wz;
    private Web w1, w2, w3, w4, w5, w6, w0;
    private Graph g;

    @Before
    public void setUp() throws Exception {  //hasieraketak
        wz= WebZerrenda.getNireWebZerrenda();
        w0= new Web(0, "a.com");
        w1= new Web(1, "b.com");
        w2= new Web(2, "c.com");
        w3= new Web(3, "d.com");
        w4= new Web(4, "e.com");
        w5= new Web(5, "f.com");
        w6= new Web(6, "g.com");
        //web-en esteken zerrendak bete:
            //a norekin lotuta
        w0.getEstekenZerrenda().add(w1); //b
        w0.getEstekenZerrenda().add(w2); //c
        w0.getEstekenZerrenda().add(w4); //e
            //b norekin lotuta
        w1.getEstekenZerrenda().add(w3);//d
            //c norekin lotuta
        w2.getEstekenZerrenda().add(w0);//a
            //d norekin lotuta
        w3.getEstekenZerrenda().add(w5);//f
            //e norekin lotuta
        w4.getEstekenZerrenda().add(w3);//d
        w4.getEstekenZerrenda().add(w6);//g
            //f norekin lotuta
        w5.getEstekenZerrenda().add(w6);//g
            //g norekin lotuta
        w6.getEstekenZerrenda().add(w5);//f
        //web-ak webZerrendan sartu
        wz.gehitu(w0);
        wz.gehitu(w1);
        wz.gehitu(w2);
        wz.gehitu(w3);
        wz.gehitu(w4);
        wz.gehitu(w5);
        wz.gehitu(w6);
        //grafoa
        g = new Graph();
        g.grafoaSortu(wz);
    }
```

```
53
54      @After
55      public void tearDown() throws Exception {
56          wz= null;
57          w1= null;
58          w2= null;
59          w3= null;
60          w4= null;
61          w5= null;
62          w6= null;
63          w0= null;
64          g= null;
65      }
66
67      @Test
68      public void Test2Erlazionatuta() {
69          System.out.println("  ");
70          System.out.println("  ");
71          System.out.println(" TestErlazionatuta ");
72          boolean ema;
73          System.out.println("  ");
74          System.out.println(" GRAFOA ");
75          g.print();
76          System.out.println("  ");
77          System.out.println("  ");
78
79          System.out.println(" "+w0.getUrlWeb() +" "+w0.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true ");
80          ema=g.erlazionatuta(w0.getUrlWeb(), w0.getUrlWeb());
81          System.out.println(ema);
82
83          System.out.println("  ");
84          System.out.println("  ");
85
86          System.out.println(" "+w0.getUrlWeb() +" "+w4.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true ");
87          ema=g.erlazionatuta(w0.getUrlWeb(), w4.getUrlWeb());
88          System.out.println(ema);
89
90          System.out.println("  ");
91          System.out.println("  ");
92
93          System.out.println(" "+w0.getUrlWeb() +" "+w5.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true ");
94          ema=g.erlazionatuta(w0.getUrlWeb(), w5.getUrlWeb());
95          System.out.println(ema);
96
97          System.out.println("  ");
98          System.out.println("  ");
99
100         System.out.println(" "+w1.getUrlWeb() +" "+w5.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true ");
101         ema=g.erlazionatuta(w1.getUrlWeb(), w5.getUrlWeb());
102         System.out.println(ema);
103
```

```
104         System.out.println("  ");
105         System.out.println("  ");
106
107         System.out.println(" "+w2.getUrlWeb() +" "+w4.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true ");
108         ema=g.erlazionatuta(w2.getUrlWeb(), w4.getUrlWeb());
109         System.out.println(ema);
110
111         System.out.println("  ");
112         System.out.println("  ");
113
114        System.out.println(" "+w4.getUrlWeb() +" "+w2.getUrlWeb()+"-rekin:    EMAITZA:
               ↪ false ");
115        ema=g.erlazionatuta(w4.getUrlWeb(), w2.getUrlWeb());
116        System.out.println(ema);
117
118         System.out.println("  ");
119         System.out.println("  ");
120
121         System.out.println(" "+w5.getUrlWeb() +" "+w0.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ false ");
122         ema=g.erlazionatuta(w5.getUrlWeb(), w0.getUrlWeb());
123         System.out.println(ema);
124     }
125
126     @Test
127     public void Test1ErlazionatutaBidea() {
128         System.out.println(" ");
129         System.out.println(" TestErlazionatutaBidea ");
130         System.out.println("  ");
131
132         System.out.println(" "+w1.getUrlWeb() +" "+w5.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true, bidea-> b.com, d.com, f.com ");
133         g.erlazionatutaBidea(w1.getUrlWeb(), w5.getUrlWeb());
134
135
136         System.out.println("  ");
137         System.out.println("  ");
138         System.out.println(" "+w0.getUrlWeb() +" "+w0.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ true, Hasierako eta amaierko nodoak berdinak dira ");
139         g.erlazionatutaBidea(w0.getUrlWeb(), w0.getUrlWeb());
140
141         System.out.println("  ");
142         System.out.println("  ");
143         System.out.println(" "+w4.getUrlWeb() +" "+w0.getUrlWeb()+"-rekin:    EMAITZA:
                ↪ false, Ez dago biderik");
144         g.erlazionatutaBidea(w4.getUrlWeb(), w0.getUrlWeb());
145     }
146 }
```

7 Ondorioak

Hasteko, eginkizun honetan *Grafo*, *ArrayList*, *HashMap*, *Array*, eta *Queue*-ekin lan egin dugu, eta horren ondorioz haien erabilera eta funtzionamendua argi geratu zaigu.

Eginkizun honen helburua grafo bat sortzea eta horrekin bilaketak egitea izan da, horregatik beti bezala eraginkortasunari garrantzia handia eman diogu.

8 Erreferentziak

- **Mendialdua, Iñigo. Datu-Egiturak eta Algoritmoak:**
  Grafoen laborategia.
  URL:
  https://egela.ehu.eus/pluginfile.php/4346026/mod_resource/content/1/Praktika%202020-2021%20Eginkizuna3.pdf