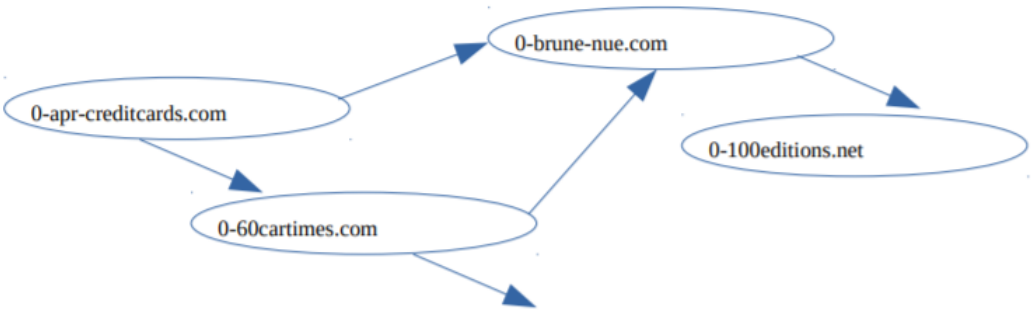
eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Web-a kudeatzeko aplikazioa (4. 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 abenduaren 21

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.

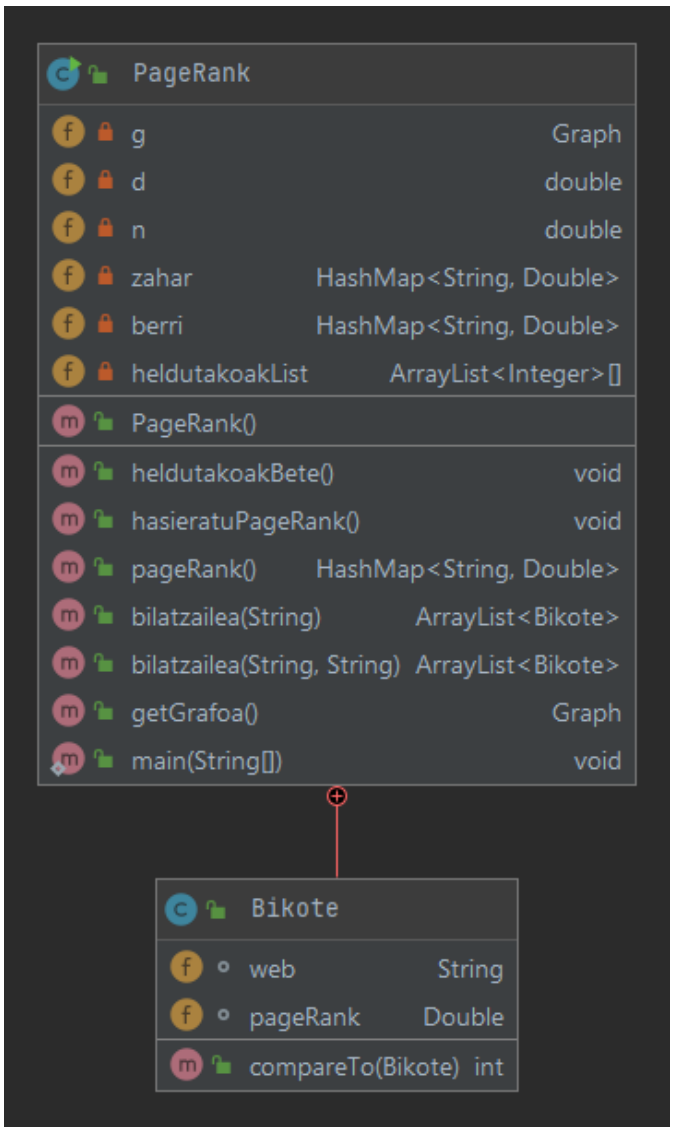
Beraz, aurrean aipatutakoa argi ikusteko, zenbait eginkizun bete ditugu lahuhilekoan zehar.

Laugarren eginkizun honetan, *Web* bakoitzeko *PageRank* balioekin lan egingo dugu. *PageRank* bilaketa-tresna batek indexatutako dokumentuen (edo web orrialdeen) garrantziari balioa emateko erabiltzen den algoritmo sorta da.

Eginkizun honetarako bi ataza ditugu. Hasteko, *PageRank* guztien kalkulua egitea da beharrezkoa. Ondoren, *GakoHitz* baten izena emanda, bere *Web* zerrenda, hurrenez hurren, bueltatuko du. Zerrenda hura ordenatuta egongo da, *PageRank*-aren arabera. Era berean bi *GakoHitz* desberdin sartuta, bi *GakoHitz* horiek dituzten *Web*-en zerrenda bueltatuko du, hau ere *PageRank*-aren arabera ordenatuta.

2 Diseinua

Laugarren eginkizun honetan (1. irudia) aurreko eginkizun guztietan inplementatutako guztia erabiltzen dugu, hau da, *Grafo*-ekin lan egin dugu, horretarako lehenengo eginkizuneko klaseak erabili ditugu datuen kargaketa egiteko, eta bilatzailea(String) metodoa egiteko, bigarren eginkizuneko *OrderedCircularLinkedList*.



1. irudia: Klase diagramaren diseinua.

3 Datu egituren deskribapena

Eginkizun honetan hainbat datu egitura ezberdin erabili ditugu. Hala nola, *grafoak*, *ArrayList*-ak, *OrderedCircularLinkedList*-ak eta *HashMap*ak.

Grafo-a eta *HashMap*-a eginkizun honetan erabiltzen diren datu egitura garrantzitsuenak dira. *Grafo*-aren sorketa, aurreko eginkizunean egin da. Laugarren eginkizun honetan beraz, sortutako grafoa erabiliko da, eta *Web* bakoitzeko, *PageRank* algoritmoa aplikatuko dugu *HashMap* batean bere balioa sartzeko.

Web-en lista ordenatzeko *OrderedCircularLinkedList* egitura erabili dugu; izan ere, egitura honetan elementuren bat sartzean ordenean egiten du, eta kasu honetan *Web*-en *PageRank*-aren arabera.

4 Metodo nagusien diseinu eta inplementazioa

4.1 Heldutakoen Array-a bete: *heldutakoakBete()*

pubic void heldutakoakBete()
// Postbaldintza: Sortzen dugu ArrayList-en Array egitura bat non sartzen dugu nodo bakoitzetik nondik ailegatzen garen. Hau da, adjList-aren alderantzizkoa sortzen du.

- Proba Kasuak:
  1. heldutakoakList bete egin da era egokian.
  2. heldutakoakList betetzean errore bat egon da.
- Algoritmoa:

```
1            bitartean(k integer heldutakoakList guztietarako){
2                heldutakoakList[k]= ArrayList berria sortu;
3            }
4            bitartean(integer i adjList-en indize guztietarako){
5                bitartean (dagokion arraylistaren balioa x aldagai guztietarako){
6                    gehitu heldutakoetan x zenbakiko arraylistean i balioa;
7                }
8            }
```

- Kostua: *O(k + k*m)*
  k = AdjList edo heldutakoakList duten luzeera. Biek luzeera berdina dute.
  m = batazbesteko heltzen diren nodoak.

4.2 Hasieratu PageRank egitura: *hasieratuPageRank()*

public void hasieratuPageRank()
// Postbaldintza: "zahar"eta "berri"izeneko HashMapak sortuta daude, eta heldutakoakList bete egin da. "berri"HashMap-ean sartu dira url guztiak 1/n balioarekin eta "zahar"izeneko HashMap-ean sartu dira url guztiak 0 balioarekin.

- Proba Kasuak:
  1. "zahar"eta "berri"HashMapak sortu eta bete dira dagokion balioekin eta heldutakoakList betetzeko funtzioa deitzean ez da arazorik egon, bete egin da.
  2. "zahar"eta "berri"HashMapa-k sortzean eta betetzean arazo bat egon da edo heldutakoakBete funtzioa deitzean arazo bat egon da. Ez da behar den moduan bete.
- Algoritmoa:

```
1            zahar = hashmapa berria;
2            berri = hashmapa berria;
3            heldutakoakBete() deitu;
4            balio = 1/n;
5            bitartean (String s keys en balio guztiak){
6                zahar.sartu(s, zero);
7                berri.sartu(s, balio);
8            }
```

- Kostua: *O(n)*
  n = Keys Array-aren luzeera.

4.3 PageRank algoritmoa aplikatu: *pageRank()*

public HashMap<String, Double> pageRank() // Postbaldintza: HashMapa bueltatzen du, url bakoitzarekin eta bere PageRank-aren balioarekin.

- Proba Kasuak:
  1. PageRank algoritmoa aplikatu da behar den bezala eta bueltatutako HashMapan url bakoitzak duen PageRank balioa lortu dugu.
  2. Algoritmoa aplikatzean arazo bat egon da (Bukle infinitua adibidez, edo HasMapean txarto sartu egin da).
- Algoritmoa:

```
1        pagerank hasieratu; // Aurreko metodoa
2        dif aldagaia 1-era hasieratu;
3        String lag aldagaia sortu;
4        integer aux sortu;
5        balioa aldagaia sortu;
6        bitartean (dif handiago 0.0001){
7            dif aldagaia 0 ra hasi;
8            zahar hash mapa berriaren balioekin ipini;
9            berri hash mapa berria sortu;
10           bitartean (i integerra o tik keys.length-1 arte){
11               balioa 0ra hasi;
12               lag da keys[i] ren balioa;
13               bitartean (k balioa heldutakoaklist[i] dagokion arraylistaren size
                     ↪ -1 ra arte doa){
14                   aux=dagokion arraylistaren k indizean dagoen balioa;
15                   balioa formula aplikatuz eguneratu;
16               }
17               balioa-ri gehitu ezkerreko zatia; ((1-d)/n)
18               berri.gehitu(lag, balioa);
19               dif aldagaian gorde berri - zaharraren balio absolutua
20           }
21       }
22       bueltatu berri;
```

- Kostua: *O(n + m)*

  n = grafoaren elementu kopurua.

  m = elementu bakoitzaren heldutako listaren elementu kopurua.

4.4 "Google"bilatzailea hitz batekin: *bilatzailea()*

public ArrayList<Bikote> bilatzailea(String gakoHitz)
// Postbaldintza: Bueltatzen du ArrayList bat ordenatuta, PageRank-aren balioaren arabera, sartutako gakoHitza duten url-ekin.

- Proba Kasuak:
  1. Sartutako hitza badago gako hitzen listan. Hitz hori duten url-ak sartu dira ArrayList-ean ordenatuta haien PageRank balioaren arabera.
  2. Sartutako hitza badago gako hitzen listan. Baina errore bat egon da ArrayList-en eraketan eta ordenaketan.
  3. Sartutako hitza ez dago gako hitzen listan. Beraz ArrayList hutsa bueltatzen da.

- Algoritmoa:

```
1           bikote motatako OrderedCircularLinkedList sortu;
2           bikoteen arraylist bat sortu ema izenekoa;
3           Hitza motatako h aldagaian gordetzen dugu GakoHitzZerrendan bilatutako
                ↪ string-aren emaitza;
4           bitartean(integer i 0 h Hitzaren weblistaren size-1 ra arte doa)
5           {
6           Web motatako w objektuan gordetzen dugu h hitzaren weblistaren "i"
                ↪ elementua;
7           "b" sortzen dugu Bikote motatakoa;
8           b ren web atributuan, w-ren url-a gordetzen dugu;
9           b ren pagerank atributuan berri hashmapan duen url horren balioa
                ↪ gordetzen dugu;
10          b orderedcircularlinkedlistean sartu; (lista deitzen da)
11      }
12      lista (OrderedCircularLinkedList motatakoa dena) -ren iteradorea sortzen
            ↪ dugu itr aldagaian;
13      bitartean (iteradoreak balioak ditu){
14          ema arraylistean gehitu listako lehen elementua;
15      }
16      bueltatu ema;
```

- Kostua: $O(h*(\frac{h}{2}) + l)$

  h/2 = *OrderedCircularLinkedList*-ak add() egitean ematen duen denboraren batazbestekoa

  h = hitz bat batazbeste dituen weblista elementu kopurua.

  l = "lista"OrderedCircularLinkedList duen luzeera.

4.5 "Google"bilatzailea bi hitzekin: *bilatzailea()*

public ArrayList<Bikote> bilatzailea(String gakoHitz1, String gakoHitz2)
// Postbaldintza: Bueltatzen du ArrayList bat ordenatuta, PageRank-aren balioaren arabera, sartutako gakoHitza1 eta gakoHitza2 duten url-ekin.

- Proba Kasuak:

  1. Sartutako hitzak badaude gako hitzen listan. Hitz horiek duten url-ak sartu dira ArrayList-ean ordenatuta haien PageRank balioaren arabera.
  2. Sartutako hitzak badaude gako hitzen listan. Baina errore bat egon da ArrayList-en eraketan eta ordenaketan.
  3. Sartutako hitzak ez daude gako hitzen listan. Beraz ArrayList hutsa bueltatzen da.

- Algoritmoa:

```
1           bikote motatako OrderedCircularLinkedList sortu;
2           lag1<Web> eta lag2<Web> arraylistak sortu gakohitz1 eta gakohitz2 ren
                ↪ weblistenkin;
3           bikoteen arraylist bat sortu ema izenekoa;
4           kont integerrra sortu eta 0ra hasieratu;
5           aurkitua izeneko boolearra falsera hasieratu;
6           bitartean (lag1-eko web bakoitzeko (w aldagaian)){
7               kont 0ra hasieratu;
8               aurkitua falsera hasieratu;
9               bitartean (aurkitua false eta kont txikiago lag2 ren luzeera){
```

```
10                  i integerrean gorde w-ren urlaren eta lag2.get(kont)-ren
                        ↪ konparazioa;
11                  baldin(i == 0 hau da, berdinak badira) {
12                      b bikote bat sortu;
13                      b-ko web atributuan w-ren urlWeb sartu;
14                      b-ko pagerank atributuan berri hashmapean gordeta dagoen
                            ↪ pageranka gorde;
15                      lista (OrderedCircularLinkedList motatakoa)-n b sartu;
16                      aurkitua boolearra true-ra jarri;
17                  }
18              }
19          }
20          Nodo bat sortu bikote motatakoa unekoa deituta;
21          unekoan gorde lista-ko azken elementua;
22          bitartean(unekoa.hurrengoa ezberdin lista-ko azken elementua) {
23              unekoa aurreratu;
24              ema-n gehitu unekoak daukan elementua;
25          }
26          eman gehitu listako azken nodoak duen elementua; // Buklean ez delako
                ↪ azkena sartu
27          bueltatu ema;
```

- Kostua: $O((n*m) + (h*(\frac{h}{2}) + l))$

  Oharra: l eta h kostuak aurreko bilatzailean azalduta.

  n = lehen gako hitzaren web zerrendaren tamaina.

  m = bigarren gako hitzaren web zerrendaren tamaina.

5 Kodea

```java
package packlEnuntziatu4;

import packlEnuntziatu1.GakoHitzZerrenda;
import packlEnuntziatu1.Hitza;
import packlEnuntziatu1.Web;
import packlEnuntziatu1.WebZerrenda;
import packlEnuntziatu2.Node;
import packlEnuntziatu2.OrderedCircularLinkedList;
import packlEnuntziatu3.Graph;

import java.io.File;
import java.io.FileNotFoundException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

public class PageRank {

    private Graph g;
    private double d;
    private double n;
    private HashMap<String, Double> zahar;
    private HashMap<String, Double> berri;
    private ArrayList<Integer>[] heldutakoakList;

    public PageRank() {
        this.d = 0.85;
        this.g = new Graph();
        g.grafoaSortu();
        this.n = g.getTh().size();
        this.heldutakoakList = new ArrayList[g.getAdjList().length];
        HashMap<String, Double> zahar = new HashMap<>();
        HashMap<String, Double> berri = new HashMap<>();
    }

    public void heldutakoakBete(){
        /*
        * post: heldutakoakList hasieratu eta bete du.
        * heldutakoakList atributuan gordeko da nondik heltzen garen nodo bakoitzera.
        */
        for(int k=0; k<heldutakoakList.length ; k++){
            heldutakoakList[k]= new ArrayList<>();
        }
        for (int i=0; i<g.getAdjList().length; i++ ){
            for (int x:g.getAdjList()[i]){
                heldutakoakList[x].add(i);
            }
        }
    }

```

```
53      public void hasieratuPageRank(){
54          zahar = new HashMap<>();
55          berri = new HashMap<>();
56          heldutakoakBete();
57          double balio = 1/n;
58          for (String s:g.getKeys()){
59              zahar.put(s, 0.0);
60              berri.put(s,balio);
61          }
62
63      }
64
65      public HashMap<String, Double> pageRank() {
66          //POST: emaitza web-orri zerrendaren web-orri bakoitzaren PageRank algoritmoaren
                ↪ balioa da
67          this.hasieratuPageRank();
68          double dif=1;
69          String lag;
70          int aux;
71          double balioa;
72          while(dif>0.0001){
73              long start = System.currentTimeMillis();
74              dif=0;
75              zahar=berri;
76              berri=new HashMap<String, Double>();
77              for (int i=0; i<g.getKeys().length; i++){
78                  balioa=0;
79                  lag=g.getKeys()[i];
80                  for (int k=0; k<heldutakoakList[i].size(); k++){
81                      aux=heldutakoakList[i].get(k);
82                      balioa=balioa+ ((zahar.get(g.getKeys()[aux])/g.getAdjList()[aux].size
                            ↪ ())*d);
83                      //lag=heldutakoakList[i].get(k);
84                  }
85                  balioa=balioa+((1-d)/n);
86                  berri.put(lag,balioa);
87                  dif=dif+Math.abs(berri.get(lag)-zahar.get(lag));
88              }
89              System.out.println("Diferentzia "+dif);
90              long end = System.currentTimeMillis();
91              NumberFormat formatter = new DecimalFormat("#0.00000");
92              System.out.println("Execution time is " + formatter.format((end - start) /
                    ↪ 1000d) + " seconds");
93          }
94          return berri;
95      }
96
97      public class Bikote implements Comparable<Bikote> {
98          String web;
99          Double pageRank;
100
101         @Override
102         public int compareTo(Bikote o) {
103             if(pageRank==o.pageRank) {
104                 return 0;
```

```
105             }else {
106                 if (pageRank<o.pageRank){
107                     return -1;
108                 }else {
109                     return 1;
110                 }
111             }
112         }
113     }
114 
115     public ArrayList<Bikote> bilatzailea(String gakoHitz) {
116         // Post: Emaitza emandako gako-hitza duten web-orrien zerrenda da, bere pagerank
                ↪ - aren arabera handienetik
117         // txikienera ordenatuta (hau da, lehenengo posizioetan pagerank handiena duten
                ↪ web - orriak agertuko dira)
118         OrderedCircularLinkedList<Bikote> lista= new OrderedCircularLinkedList<>("Page
                ↪ ranken lista");
119         ArrayList<Bikote> ema= new ArrayList<>();
120         Hitza h= GakoHitzZerrenda.getNireGakoHitzZerrenda().bilatuHitza(gakoHitz);
121         for(int i=0; i<h.getWebLista().size(); i++){
122             Web w= h.getWebLista().get(i);
123             Bikote b= new Bikote();
124             b.web=w.getUrlWeb();
125             b.pageRank= berri.get(w.getUrlWeb());
126             lista.add(b);
127         }
128         Iterator<Bikote> itr= lista.iterator();
129         while (itr.hasNext()){
130             ema.add(lista.removeFirst());
131         }
132         return ema;
133     }
134 
135     public ArrayList<Bikote> bilatzailea(String gakoHitz1, String gakoHitz2){
136         // Post: Emaitza emandako gako-hitzak dituzten web-orrien zerrenda da, bere
                ↪ pagerank-aren arabera handienetik
137         //  txikienera ordenatuta(hau da, lehenengo posizioetan pagerank handiena duten
                ↪ web-orriak agertuko dira
138         OrderedCircularLinkedList<Bikote> lista= new OrderedCircularLinkedList<>("Page
                ↪ ranken lista");
139         ArrayList<Web> lag1= GakoHitzZerrenda.getNireGakoHitzZerrenda().bilatuHitza(
                ↪ gakoHitz1).getWebLista();
140         ArrayList<Web> lag2= GakoHitzZerrenda.getNireGakoHitzZerrenda().bilatuHitza(
                ↪ gakoHitz2).getWebLista();
141         ArrayList<Bikote> ema= new ArrayList<>();
142         int kont=0;
143         boolean aurkitua=false;
144         for (Web w:lag1){
145             kont=0;
146             aurkitua=false;
147             while (kont< lag2.size() && !aurkitua){
148                 int i= w.getUrlWeb().compareTo(lag2.get(kont).getUrlWeb());
149                 if(i==0) {
150                     Bikote b = new Bikote();
151                     b.web = w.getUrlWeb();
```

```
152                     b.pageRank = berri.get(w.getUrlWeb());
153                     lista.add(b);
154                     aurkitua=true;
155                 }
156             }
157         }
158         Node<Bikote> unekoa=lista.getLast();
159         while(unekoa.next!=lista.getLast()) {
160             unekoa=unekoa.next;
161             ema.add(unekoa.elem);
162         }
163         ema.add(lista.getLast().elem);
164         return ema;
165     }
166 
167     public Graph getGrafoa(){
168         return this.g;
169     }
170 
171     public static void main(String[] args) {
172 
173         File wordsFitxeroa = null;
174         File webIndexFitxeroa = null;
175         File webEstekaFitxeroa = null;
176 
177         wordsFitxeroa = new File ("resources\\words.txt");
178         webIndexFitxeroa = new File ("resources\\index.txt");
179         webEstekaFitxeroa = new File ("resources\\pld-arcs-1-N.txt");
180 
181         GakoHitzZerrenda ghz = GakoHitzZerrenda.getNireGakoHitzZerrenda();
182         WebZerrenda wz = WebZerrenda.getNireWebZerrenda();
183 
184         try { // Lehenik fitxeroen karga egiten dugu.
185             ghz.fitxeroaKargatu(wordsFitxeroa);
186             wz.indexFitxeroaKargatu(webIndexFitxeroa);
187             wz.arcFitxeroaKargatu(webEstekaFitxeroa);
188 
189             System.out.println("");
190             System.out.println("");
191             System.out.println("");
192 
193         } catch (FileNotFoundException e) {
194             e.printStackTrace();
195         }
196 
197         PageRank p =new PageRank();
198         p.pageRank();
199         ArrayList<PageRank.Bikote> ema= p.bilatzailea("casino");
200         for (int i=0; i<ema.size();i++){
201             System.out.println("Web url: " + ema.get(i).web + " PageRank: " + ema.get(i).
                    ↪ pageRank);
202         }
203     }
204 }
```

6 Ondorioak

Lauhilabete honetan datu egitura ezberdinei buruzko aplikazio praktikoak ikusi ditugu; izan ere, duela pare bat hilabetetik hona asko hobetu dugu gure maila.

Azkenengo eginkizun honetan praktikan jarri dugu aurretik egindako guztia, eta ikusi dugu aurreko eginkizun guztiek haien artean duten lotura.

7 Erreferentziak

- **Mendialdua, Iñigo. Datu-Egiturak eta Algoritmoak:**
  Grafo eta PageRank laborategia.
  URL:
  https://egela.ehu.eus/pluginfile.php/4395397/mod_resource/content/2/Praktika%202020-2021%20Eginkizuna4.pdf