*Tirane, me 17.06.2019*

Emri: **Tea Lico** Grupi **A2**

## Ushtrim 1 (5 pike)

Fjaloret mund te implementohen nepermjet listave te lidhura nje drejtimore, pemeve binare, pemeve te balancuara dhe tabelave hash. Megjithate, jo te gjitha keto struktura kane te njejten performance.

Te shkruhet nje program qe lexon nje skedar text me madhesi te konsiderueshme dhe shfaq vetem njehere te gjitha fjalet e vecante qe shfaqen ne te. Gjate leximit te skedarit, programi duhet te krijoje nje fjalor me te gjitha fjalet qe jane shfaqur ne skedar. Sa here qe shfaqet nje fjale e re, dmth qe nuk ndodhet ne fjalor, programi duhet ta shtoje ate ne fjalor. Fjalori te realizohet:

1. nepermjet listes se lidhur nje drejtimore

2. nepermjet pemes binare

3. nepermjet pemes se balancuar

4. nepermjet tabeles hash.

Per secilen nga strukturat e mesiperme, te matet koha e ezkekutimit per:

1. nje skedar me 100 fjale

2. nje skedar me 1000 fjale

3. nje skedar me 100000 fjale

4. nje skedar me 1000000 fjale

Cila eshte varesia e kohes se ekzekutimit te programit (per secilen nga strukturat) nga madhesia e inputit? Cila nga strukturat rezulton me eficente? Po me pak?

### Pergjigje 1

**1. Realizimi i fjalorit nepermjet listes se lidhur nje drejtimore**

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

//percaktimi i struktures se listes 1-Drejtimore

typedef struct nyje {

string vlera;

struct nyje \*pas;

}nyje;

//funksioni i meposhtem shton nje element te ri ne liste

nyje \*shtoElementFillim (nyje \*koka,string fjala){

nyje \*elementRi = new nyje; //krijohet nje element i ri

elementRi->vlera = fjala; //vlera e elementit te ri behet sa fjala

elementRi->pas = koka; // pasardhesi i elementit te ri behet sa koka e listes

koka = elementRi; // elementi shtohet ne fillim te listes, dmth ai behet koka e listes

return koka; //kthejme koken e listes

}

//funksioni i meposhtem afishon listen duke mar si parameter koken e listes

//afishohet vlera per cdo nyje deri sa te kete mbaruar bredhja

void afishoListe (nyje \*koka){

nyje \*tmp = koka;

while (tmp!=NULL){

cout<<tmp->vlera<<endl;

tmp=tmp->pas;

}

}

//funksioni i meposhtem kontrollon nese nje fjal e dhene ndodhet ne liste duke bredhur

//te gjithe listen, nese e gjen kthen vleren booleane true, ne rast te kundert

//shkon deri ne fund te listes dhe kthen false

bool \*ndodhetNeList(nyje \*koka,string fjala){

nyje \*tmp = koka;

while(tmp!=NULL){

if(fjala.compare(tmp->vlera) == 0){

return true;

}

tmp=tmp->pas;

}

return false;

}

int main() {

//deklarojm nje objekt te klases ifstream dhe bejme lidhjen me skedarin

ifstream in;

in.open("100Fjale.txt");

//deklarojm nje strukture te tipit nyje dhe e inicializojme ate me NULL

nyje \*lista = NULL;

if(in.is\_open()){

//deklarojme nje variabel per te numeruar numrin e fjaleve ne fjalor

int nrFjaleshNeFjalor = 0;

while(!in.eof()){

//bejme leximin per cdo fjale

string fjala;

in>>fjala;

//kontrollojme nese fjala ndodhet ne liste

//nese nuk ndodhet e shtojme ne liste nese jo vazhdojme me fjalen tjeter

if(ndodhetNeList(lista,fjala) == false){

//shtojme fjalen ne liste dhe risim me 1 numrin e fjaleve te gjetura

lista = shtoElementFillim(lista,fjala);

nrFjaleshNeFjalor++;

}

}

//afishojme listen dhe numrin e fjaleve te saj

afishoListe(lista);

cout<<endl<<"Numri i fjaleve te ndryshme te fjalorit eshte: "<<nrFjaleshNeFjalor<<endl;

}

else{

cout<<"Gabim ne hapjen e skedarit";

}

return 0;

}

**2. Realizimi i fjalorit nepermjet pemes binare**

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

//percaktimi i struktures se pemes binare

typedef struct peme{

string fjala;

struct peme \*majtas;

struct peme \*djathtas;

struct peme \*prind;

} peme;

//funksioni i meposhtem krijon nje kulm per nje fjale te dhene

peme \* krijoKulm(string fjala){

peme \* p = new peme;

p->fjala = fjala;

p->majtas = NULL;

p->djathtas = NULL;

p->prind = NULL;

return p;

}

//funksioni ben kthimin ne numer te plote te fjales

//duke perdorur nje algoritem qe kthen cdo karakter ne numer

//te plote dhe ben shumen pas cdo iteracioni e shumezon me 10

int ktheInteger(string fjala){

int s = 0;

for(int i=0;i<fjala.length();i++){

s = s\*10 + fjala.at(i);

}

return s;

}

//funksioni shton kulme ne nje peme binar,

//p - rrenja e pemes, z - kulmi qe do te shtohet

peme \* shtoNePeme(peme \*p, peme \*z){

//perdorim 2 shenjuesa ndihmes njeri shenjon NULL dhe tjetri ne rrenjen e pemes p

peme \* y = NULL;

peme \* x = p;

int fjalaRe = ktheInteger(z->fjala); //bejme kthimin e fjales ne numer te plote

//per sa kohe qe nuk jemi ne fund te pemes

while(x != NULL){

y = x; //y shenjon tek kulmi x qe fillimisht eshte rrenja e pemes

//bejme kthimin e fjales qe ndodhtet tek kulmi x ne numer te plote

int xFjala = ktheInteger(x->fjala);

//nqs vlera e fjales qe duam te shtojme eshte me e vogel se vlera e fjales se gjetur

//levizim shenjuesin majtas, nqs eshte me e madhe e levizim shenjuesin djathtas

//nqs eshte e barabarte dmth kemi gjetur te njejten fjale dhe nuk e shtojme ne peme

//por kthejme pemen pa ndryshuar asgje, kjo shmang perseritjen e fjaleve

if(fjalaRe < xFjala)

x = x->majtas;

else if(fjalaRe > xFjala)

x = x->djathtas;

else

return p;

}

//nqs nuk eshte gjetur fjala atehere prindi i fjales qe duam te shtojme behet sa kulmi i fundit qe

//kemi pasi kemi bredhur pemen

z->prind = y;

//y eshte NULL nqs pema ka qen bosh dhe rrenja e pemes behet sa z

if(y==NULL){

p = z;

}

//nqs y nuk ka qen NULL atehere

else{

//kthejme vleren e fjales ne kulmin e fundit ne numer te plote

int yFjala = ktheInteger(y->fjala);

// nqs vlera e fjales qe duam te shtojme eshte me e vogel se vlera e kulmit ku duam ta

//shtojme, fjalen e shtojme majtas dhe nqs eshte me e madhe se vlera e kulmit ku duam ta

//shtojme, fjalen e shtojme djathtas

if (fjalaRe < yFjala)

y->majtas = z;

else if(fjalaRe > yFjala)

y->djathtas = z;

}

//kthejme pemen me kulmin e shtuar

return p;

}

//funksioni kthen numrin e kulmeve te pemes

int nrKulme(peme \* p){

if(p == NULL)

return 0;

else if(p->majtas == NULL && p->djathtas == NULL)

return 1;

else

return 1 + nrKulme(p->majtas) + nrKulme(p->djathtas);

}

//funksioni afishon pemen sipas bredhjes pararendore

//ku trajtohet rrenja, me pas nenpema e majte

//dhe nenpema e djathte

void afishoPeme(peme \*p) {

if(p != NULL) {

cout << p->fjala << endl;

afishoPeme(p->majtas);

afishoPeme(p->djathtas);

}

}

int main(){

//deklarojm nje objekt te klases ifstream dhe bejme lidhjen me skedarin

ifstream in;

in.open("1\_000Fjale.txt");

//deklarojme nje strukture te tipit peme dhe e inicializojme ate me NULL

peme \* p = NULL;

if(in.is\_open()){

while (!in.eof()) {

//bejme leximin per cdo fjale

string fjale;

in>>fjale;

//shtojme ne pemen p, nje kulm te krijuar nga funksioni krijoKulm()

//duke vendosur si parameter fjalen e lexuar

p = shtoNePeme(p,krijoKulm(fjale));

}

//afishojme pemen dhe numrin e fjaleve te fjalorit

afishoPeme(p);

cout<<endl<<"Numri i fjaleve te fjalorit eshte: "<<nrKulme(p)<<endl;

}

else{

cout<<"Gabim ne hapjen e skedarit";

}

return 0;

}

**3. Realizimi i fjalorit nepermjet pemes se balancuar**

**4. Realizimi i fjalorit nepermjet tabeles hash**

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

//percaktimi i struktures se listes 1-Drejtimore

typedef struct nyje {

string vlera;

struct nyje \*pas;

}nyje;

//funksioni i meposhtem shton nje element te ri ne fillim te nje liste

nyje \*shtoElement (nyje \*koka,string fjala){

nyje \*elementRi = new nyje; //krijohet nje element i ri

elementRi->vlera = fjala; //vlera e elementit te ri behet sa fjala

elementRi->pas = koka; // pasardhesi i elementit te ri behet sa koka e listes

koka = elementRi; // elementi shtohet ne fillim te listes, d.m.th. ai behet koka e listes

return koka; //kthejme koken e listes

}

//funksioni i meposhtem kontrollon nese nje fjal e dhene ndodhet ne liste duke bredhur

//te gjithe listen, nese e gjen kthen true, ne rast te kundert

//jemi ne fund te listes dhe do te kthehet false

bool ndodhetNeList(nyje \* l,string fjala){

nyje \* tmp = l;

while(tmp!=NULL){

if(tmp->vlera==fjala){

return true;

}

tmp=tmp->pas;

}

return false;

}

//funksioni llogarit vleren e funksionit hash per nje fjale te dhene,

// duke perdorur funksionin h(k)=k mod N

//konvertimi i fjales ne numer na sherben per te llogaritur vleren hash

int llogaritVlerenHash(string fjala,int N){

int s=0;

for(int i=0;i<fjala.length();i++){

s = s + fjala.at(i);

}

return s%N;

}

//funksioni afishon listat e tabeles hash duke bredhur

//secilen prej tyre deri ne fund

void afishoListat (nyje \* T[],int nrEl){

for(int i=0;i<nrEl;i++){

nyje \*tmp = T[i];

cout<<"Elementet ne pozicionin "<<i<<" jane: "<<endl;

while(tmp!=NULL){

cout<<tmp->vlera<<endl;

tmp=tmp->pas;

}

cout<<endl;

}

}

int main(){

ifstream in; //deklarojm nje objekt te klases ifstream

in.open("100Fjale.txt"); // dhe bejme lidhjen me skedarin

const int nrEl=10;

nyje \*tabelaHash[nrEl]; //deklarojme nje tabele te tipit nyje

//inicializojme secilen nyje te tabeles me vleren NULL

for(int i=0;i<nrEl;i++){

tabelaHash[i]=NULL;

}

if(in.is\_open()){

int nrFjaleshNeFjalor = 0; //deklarojme nje variabel per te numeruar numrin e fjaleve

while(!in.eof()){

string fjala;

in>>fjala; //bejme leximin per cdo fjale

int hash = llogaritVlerenHash(fjala,nrEl); //llogarisim vleren hash per fjalen e lexuar

//kontrollojme nese fjala ndodhet ne listen me koke te vendosur

//ne indeksin hash te tabeles

if(ndodhetNeList(tabelaHash[hash],fjala)==false){

//shtojme fjalen ne listen me indeks hash te tabeles

//dhe rrisim me 1 numrin e fjaleve te gjetura

tabelaHash[hash]=shtoElement(tabelaHash[hash],fjala);

nrFjaleshNeFjalor++;

}

}

//afishojme listat per cdo element te tabeles dhe numrin e fjaleve te fjalorit

afishoListat(tabelaHash,nrEl);

cout<<endl<<"Numri i fjaleve te fjalorit eshte: "<<nrFjaleshNeFjalor<<endl;

}

else{

cout<<"Gabim ne hapjen e skedarit";

} return 0;

}

**Varesia e kohes se ekzekutimit te programit (per secilen nga strukturat)**

**Lista 1 Drejtimore, Koha e ekzekutimit per:**

1. 100 Fjale = 0.1543 sekonda
2. 1000 Fjale = 0.9344 sekonda
3. 100000 Fjale = 23.17 sekonda
4. 1000000 Fjale = 97.7 sekonda

**Pema Binare, Koha e ekzekutimit per:**

1. 100 Fjale = 0.1437 sekonda
2. 1000 Fjale = 0.7177 sekonda
3. 100000 Fjale = 6.17 sekonda
4. 1000000 Fjale = 11.38 sekonda

**Tabela Hash, Koha e ekzekutimit per:**

1. 100 Fjale = 0.1618 sekonda
2. 1000 Fjale = 0.9923 sekonda
3. 100000 Fjale = 7.137 sekonda
4. 1000000 Fjale = 17.3 sekonda

Me eficente rezulton **Pema binare** nuk kontrollojme gjith pemen pasi ne varesi te vleres se vjales fjales kontrollohet ose nenpema e majte ose nenpema e djathte per cdo kulm, kjo ul ndjeshem kompleksitetin e algoritmit.

Me pak eficente rezulton **Lista 1Drejtimore** sepse sa here kontrollojme nese nje element ndodhet ne Liste duhet te bredhim gjith listen nga fillimi

**Kompleksiteti i algoritmit per:**

**1. rastin e perdorimit te listave te lidhura eshte:**

Algoritmi shtoElementFillim -> O(1)  
 Algoritmi ndodhetNeListe -> O(n)  
 Algoritmi afishoListe -> O(n)

Main -> O(4 + n(n + 2) + n + 2) = O(n2)

**2. rastin e perdorimit te pemeve binare eshte:**

Algoritmi krijoKulm -> O(1)

Algoritmi shtoNePeme -> O(h) | h-lartesia e pemes

Algoritmi nrKulme -> O(n)

Algoritmi afishoPeme -> O(n)

Main -> O(4 + n(3 + h) + n + 2) = O(nh) | h-lartesia e pemes

**3. rastin e perdorimit te pemeve te balancuara eshte:**

**4. rastin e perdorimit te tabelave hash eshte:**

Algoritmi shtoElement -> O(1)  
 Algoritmi ndodhetNeListe -> O(n)

Algoritmi afishoListat -> O(n)

Main -> O(16 + n(3 + n 3) + 4) = O(n2)

## Ushtrim 2 (5 pike)

*Web crawler* eshte lloj software i cili sherben per te zbuluar faqe qe jane online te publikuara. Nje *web crawler* kontrollon faqet e internetit per adresa URL. Nese gjenden te tilla ne permbajtjen e faqeve, atehere faqet e identifikuara nga keto adresa jane te arritshme prej faqes aktuale. *Web crawler* fillon nga me nje liste adresash faqesh, te cilat kontrollohen te parat. Pas tyre, vizitohen faqet adresat e se cilave gjendeshin ne faqet paraardhese. Ky proces mund te vazhdoje deri per nje nivel te caktuar thellesie.

Te ndertohet nje program i cili realizon nje *web crawler* te thjeshte. Programi duhet te perdore *command- line* per te marre 3 argumenta:

* Adresen e faqes prej nga do te filloje kontrolli (psh. [www.fshn.edu.al](http://www.fshn.edu.al))
* Nivelin e thellesise se kontrollit (psh. nese thellesia eshte 4 atehere duke nisur nga faqja e pare, kontrollohen faqet e gjetura ne faqen e pare (niveli pare), pastaj faqet e gjetura ne faqet e nivelit te pare (niveli dyte), pastaj faqet e gjetura ne faqet e nivelit te dyte (niveli trete) dhe se fundmi faqet e gjetura ne faqet e nivelit te trete.
* Direktorine ku do te ruhen rezultatet e kontrollit (dmth te gjitha adresat e faqeve te arritshme nga faqja me adresen e dhene si argumenti i pare).

Programi duhet te lexoje permbajtjen e faqes me adresen e dhene si argument i pare dhe te kerkoje aty per adresa URL. Keto adresa duhet te shkruhen ne direktorine e dhene si parameter i trete dhe duhet te sherbejne per ciklin tjeter te kontrollit. Ky proces duhet te vazhdoje derisa te arrihet niveli i dhene si argument i dyte.

Programi duhet te shmange ciklet e pafundme, dmth viziten e nje faqeje qe eshte vizituar dhe kontrolluar me pare. Psh. nese faqja [www.fshn.edu.al](http://www.fshn.edu.al) permban link per tek faqja [www.unitir.edu.al](http://www.unitir.edu.al) dhe kjo e fundit permban link per tek faqja [www.fshn.edu.al](http://www.fshn.edu.al) atehere jemi ne kushtet e nje cikli te pafundem.

### Pergjigje 2

*// Krijojme nje klas Item e cila na mban per cdo element  
// nje vlere e tipit numer i plote per thellesine dhe   
// nje String per url***public class** Item {  
 **private int depth**;  
 **private** String **url**;  
  
 **public** Item(**int** depth, String url) {  
 **this**.**depth** = depth;  
 **this**.**url** = url;  
 }  
  
 **public int** getDepth() {  
 **return depth**;  
 }  
  
 **public** String getUrl() {  
 **return url**;  
 }  
  
 *//Metoda toString() afishon vleren e thellesiste per kete element   
 //dhe permbajtjen e URL* **public** String toString() {  
 String s = **""**;  
 **for**(**int** i = 0; i < **depth**; i++)  
 s += **" "**;  
 s = s + **"Thellsesia "** + **depth** + **" >> "** +**url**;  
 **return** s;  
 }  
}

*//Krijojme nje klas ItemQueue e cila do te mbaje nje  
//rradhe me Objekte te tipit Item, pra do te mbaje  
//nje rradhe me elemente ku secili element ka  
//thellesine dhe url-ne e tij***import** java.util.ArrayList;  
  
**public class** ItemQueue {  
 **private** ArrayList<Item> **items**;  
  
 **public** ItemQueue() {  
 **items** = **new** ArrayList<>();  
 }  
  
 */\*\*  
 \* Metoda enqueue(Item item) Shton nje Item ne rradhe  
 \* ne qofte se eshte e mundur, eshte e mundur ne qofte se rradha  
 \* nuk e permban ate Item, kthen true nqs perfundon me sukses.  
 \* kthen false nqs Item ndodhet ne rradhe, pra ai nuk mund te shtohet  
 \* \*/* **public boolean** enqueue(Item item) {  
 **if**(**items**.contains(item))  
 **return false**;  
 **items**.add(**items**.size(),item);  
 **return true**;  
 }  
  
 */\*\*  
 \* Heq koken e rradhes nese rradha nuk eshte bosh  
 \* dhe e kthen ate ose kthen null ne qofte se rradha eshte bosh  
 \* \*/* **public** Item dequeue() {  
 **if**(**items**.size() != 0) {  
 Item item = **items**.get(0);  
 **items**.remove(item);  
 **return** item;  
 }  
 **return null**;  
 }  
  
 */\*\*  
 \* Metoda getSize() kthen numrin e elementeve qe permban rradha  
 \* \*/* **public int** getSize() {  
 **return items**.size();  
 }  
  
  
  
}

*//Ndertojme klasen per te realizuar WebCrawler-in  
//e cila do te mbaje thelelsine dhe url-ne nga do filloj kerkimi  
//dhe nje rradhe me elemente***import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.net.URL;  
**import** java.util.Scanner;  
  
**public class** WebCrawler {  
 **private int depth**;  
 **private** ItemQueue **queue**;  
 **private** String **url**;  
  
 **public** WebCrawler(String url, **int** depth)  
 {  
 **this**.**url** = url;  
 **queue** = **new** ItemQueue();  
 **this**.**depth** = depth;  
 }  
  
 */\*  
 \* Metoda getPageLinks mer si parameter nje element dhe ne qofte se thellesia e elementit  
 \* eshte me e madhe se thellesia e percaktuar nuk mund te kerkohet per url, ky eshte dhe kushti i stopimit  
 \* Ne rast te kundert krijojme nje rradhe te re dhe nqs elementi qe duam te shtojme nuk ndodhet  
 \* ne rradhen me url atehere lexojme url per ate element,rritim thellesine me 1 per kete element,  
 \* lexojme cdo string qe permban url-ja dhe nqs  
 \* ndodhet nje element qe permban tekstin href="https:// atehere kemi gjetur nje url  
 \* dhe shtojme ne rradhen e krijuar ne fillim vetem nqs nuk ndodhet  
 \* Me pas bredhim gjith Rradhen e krijuar,heqim elementin e pare dhe me pas per kete element therasim metoden getPageLinks rekursivisht e cila do te kerkoje perseri per url  
 \* vetem nqs nuk e tejkalone thellesine qe ka dhen perdoruesi  
 \* Kur arrihet thellesia perfundon ekzekutimi i metodes  
 \*/* **private void** getPageLinks(Item item)  
 {  
 **int** currentDepth = item.getDepth();  
 **if**(currentDepth > **depth**)  
 **return**;  
 **try** {  
 ItemQueue links = **new** ItemQueue();  
 **if**(**queue**.enqueue(item)){  
 currentDepth++;  
 URL url = **new** URL(item.getUrl());  
 Scanner in = **new** Scanner(url.openStream());  
 **while** (in.hasNext()){  
 String next = in.next();  
 **if** (next.contains(**"href=\"https://"**)) {  
 String foundURL = **""**;  
 **int** start = next.indexOf(**"\""**) + 1;  
 **int** end = next.lastIndexOf(**"\""**);  
 foundURL = foundURL + next.substring(start,end);  
 links.enqueue(**new** Item(currentDepth,foundURL));  
 }  
 }  
 in.close();  
 **while**(links.getSize() != 0){  
 Item newItem = links.dequeue();  
 getPageLinks(newItem);  
 }  
 }  
 }  
 **catch** (IOException e) {  
 **return**;  
 }  
 }  
  
 */\*  
 \* Metoda saveLinks merr si parameter emrin e skedarit ku duam te ruajm URL e gjetura  
 \* Ajo theret metoden getPageLinks e cila si fillim mer si parameter nje element te ri  
 \* me thellesi 0(thellesia minimale per te filluar kerkimin)  
 \* dhe url sa url e percaktuar nga perdoruesi per te filluar kerkimin dhe shkruan ne skedar  
 \* elementet e rradhes duke hequr nga rradha te parin dhe duke e afishuar ne skedar  
 \* me ane te metodes toString() afishon secilin ne baze te url dhe thellesise perkatese  
 \* Kjo perseritet deri sa rradhe te behet boshe   
 \*/* **public void** saveLinks(String fileName) **throws** IOException  
 {  
 getPageLinks(**new** Item(0,**url**));  
 FileWriter out = **new** FileWriter(fileName, **false**);  
 **while**(**queue**.getSize()!=0)  
 {  
 Item item = **queue**.dequeue();  
 out.write(item.toString()+**"\n"**);  
 }  
 out.close();  
 }

**public static void** main(String[] args)  
 {  
 **try** {  
 String pageURL = args[0];  
 **int** depth = Integer.*parseInt*(args[1]);  
 String fileName = args[2];  
  
 WebCrawler webCrawler = **new** WebCrawler(pageURL,depth);  
 webCrawler.saveLinks(fileName);  
 }  
 **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **catch** (NumberFormatException e){  
 e.printStackTrace();  
 }  
 }  
}