*Tirane, me 15.06.2019*

Emri: KRISTI KOTINI Grupi B1 VITI II

## 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

Per zgjidhjen e detyres eshte menduar ndertimi i programit duke krijuar klasa per cdo struktur te dhenash per implementimin e fjalorit. Per secilen klas jan ndertuar metodat e shtimit te fjaleve ne fjalor kur fjala nuk gjendet ne te. Me posht esht kodi per cdo algoritem insertimi me kompleksitetin perkates dhe nje tabele permbledhese me kohen e ekzekutimit per secilin program.

Listat

void shto(const string &s){

koka = new node{s,koka};

i++;

}

void kerko\_shto(const string &s){

node \*tmp = koka;

while(tmp!=NULL){

if(tmp->fjal == s) return; kerkimi ne list eshte linear me kompleksitet

tmp = tmp->next; O(n) ku n eshte nr elementeve te listes.

}

shto(s);}

Pemet Binare Te kerkimit

void kerko\_shto(const string &s){

node\_p \*tmp = rrenja;

while(tmp!=NULL){

if(tmp->fjal == s) return;

if(s<tmp->fjal) {

if (tmp->left == nullptr)

{

tmp->left = new node\_p {s,nullptr,nullptr};

i++;

return;

}else tmp =tmp->left;

}else{

if (tmp->right == nullptr)

{

tmp->right = new node\_p {s,nullptr,nullptr};

i++;

return;

}else tmp =tmp->right;

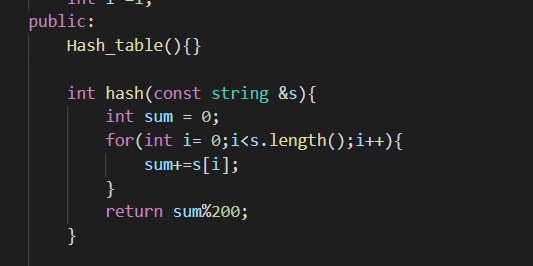
}

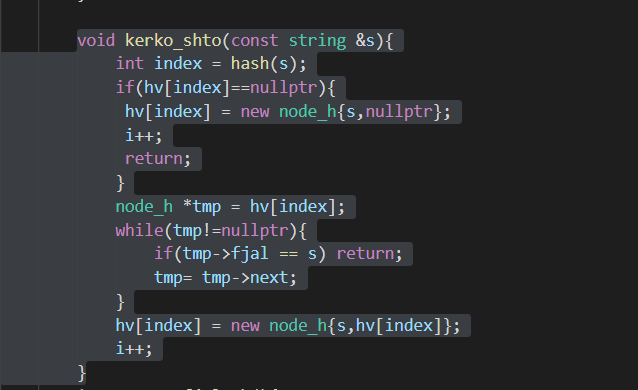
}

Ne rastin e pemeve binare te kerkimit kemi qe gjetja e elementin nese eshte shtuar ne peme ka kompleksitet O(logn) ndersa shtimi esht konstant O(1). Ne total O(1) +O(logn) = O(logn);

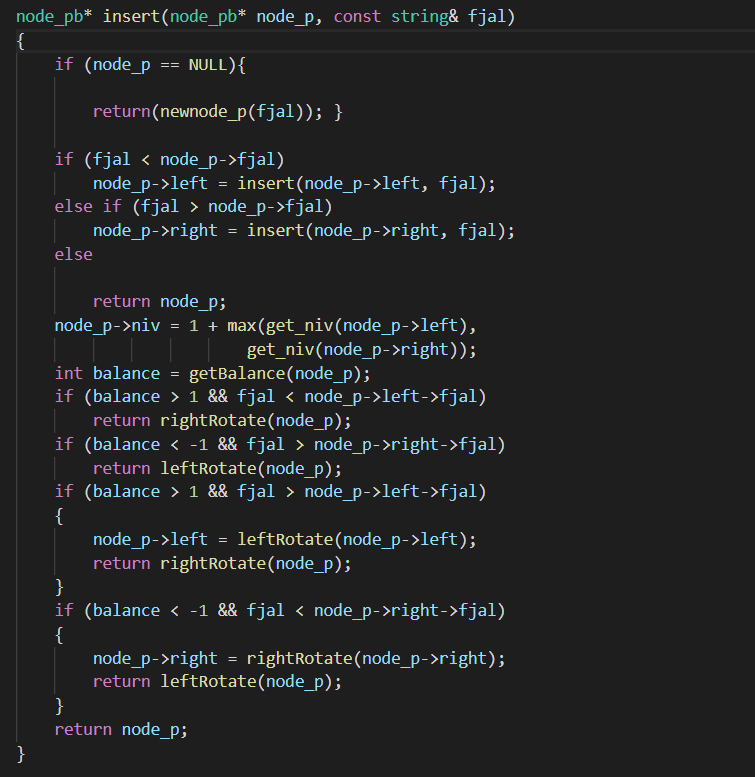
Hash Table

Ne rastin e Hash table kompleksiteti varet shume nga funksioni hash. Sa me i mire funksioni hash aq me afer kompleksitetit te konstantes shkon programi. Implementimi i Tab hash me lista ka O(1) per gjetjen ne tab te fjales dhe O(n) per bredhjen e listes edhe pse funksioni hash i cili implementuar sic duhet con ne nje shperndarje sa me te mire per tiu afruar O(c).





Pemet e Balancuar



Pemet e Balancuara jane ne vetevete peme te kerkimit binar por me nje kusht me teper ruajtaj e balances ndermjet niveleve. Kjo gje arrihet nepermjet rrotullimev te cilat ndodhin pas shtimit te nje elementi qe prish balancen. Edhe pse jan algoritma shume kompleks ato bejne te mundur qe kjo peme te kete nje kompleksitet kohor si per shtim dhe kerkim O(log n)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Implementimi | 100 fjale | 1000 fjale | 100 000 fjale | 1 000 000 fjale | Fjalet e Gjetura | Kompleksiteti ne Total |
| Lista | 7 milisek | 27 milisek | 5,5 sekonda | 3,7 min | 73/436/10790/37589 | O(n^2) |
| Pemet binare | 0-1 milisek | 1 milisek | 55 milisek | 0.6 sek | 73/436/10790/37589 | O(nlogn) |
| Pemet e balancuara | 1 milisek | 2 milisek | 68 milisek | 5 sek | 73/436/10790/37589 | O(nlogn) |
| Hash Table | 0 milisek | 0 milisek | 68 milisek | 2,5 sek | 73/436/10790/37589 | O(n) |

## 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:

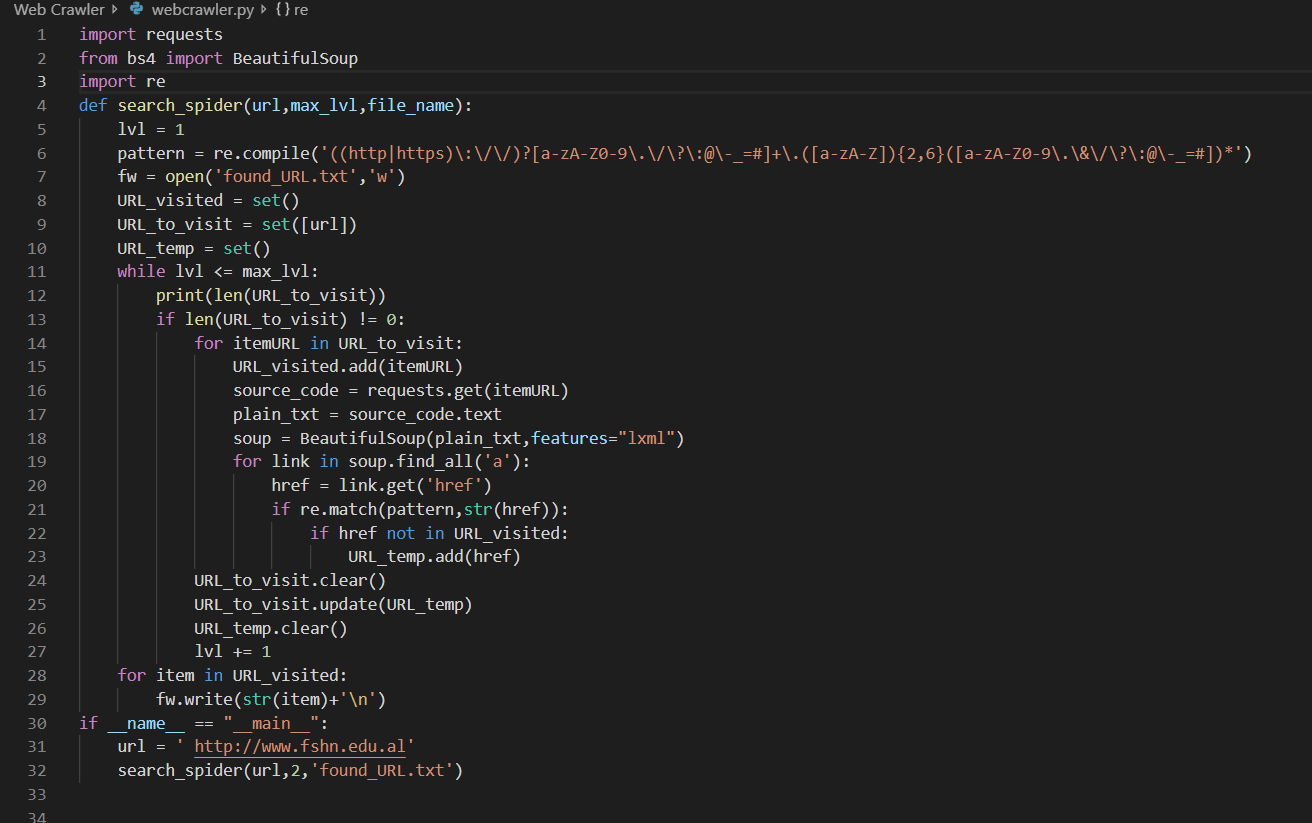
* Adresen e faqes prej nga do te filloje kontrolli (psh. [www.fshn.edu.al](http://www.fshn.edu.al))
* Nivelin e thellesise se kontrollit (psh. nese thellesia eshte 4 atehere duke nisur nga faqja e pare, kontrollohen faqet e gjetura ne faqen e pare (niveli pare), pastaj faqet e gjetura ne faqet e nivelit 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

WebCrawler i implementuar ne Python.



Per sa i perket webcrawlerit kompleksiteti i ketij programi eshte me kompleks. Nje optimizim sa me i mir i tij, duke specializuar dhe ngushtuar kerkimin e tij per asete te caktuar con ne nje performanc me te lart. Per sa i perket kohes se ekzekutimit webcrawleri varet nga 2 faktor. Niveli qe kalojm per te kerkuar si dhe numri i linqeve qe gjenden ne secilin nivel. Per kete arsye mund te themi se kompleksiteti i nje webcrawleri eshte O(n x m) ku n eshte niv dhe m nr i linqeve.