SPECIFIKACIJA DRUGOG PROJEKTNOG ZADATKA

Osnovi informacionih sistema i softverskog inženjerstva
2019/2020

Sadržaj

1. Uvod

U okviru projektnog zadatka potrebno je implementirati mašinu za pretraživanje tekstualnih dokumenata (engl. *search engine*).

Program prilikom pokretanja treba da obiđe stablo direktorijuma u datotečkom sistemu počevši od datog korena, da parsira tekstualne datoteke u njima i da izgradi strukture podataka potrebne za efikasno pretraživanje. Nakon toga, program omogućuje korisniku da unosi tekstualne upite koji se sastoje od jedne ili više reči razdvojenih razmakom, pretražuje dokumente koristeći prethodno kreirane strukture podataka i korisniku ispisuje rangirane rezultate pretrage u vidu putanja do dokumenata.

2. Funkcionalni zahtevi

2.1 Parsiranje HTML dokumenata (#parsiranje_skupa_HTML_dokumenata)

Korisniku je potrebno omogućiti odabir korenskog direktorijuma u okviru kojeg želi da pretražuje dokumenate. Dokumenti se pretražuju u zadatom direktorijumu kao i svim poddirektorijumima. Nakon odabira početnog direktorijuma, program treba da parsira samo HTML dokumente koristeći parser koji je priložen uz specifikaciju projektnog zadatka, dok ostali tipovi dokumenata treba da budu ignorisani.

Nakon parisranja, za svaki HTML dokument *X* treba da budu dostupne sledeće informacije:

- dokumenti koji imaju link ka dokumentu *X,*
- dokumenti ka kojima dokument *X* ima link i
- prozvoljne, dodatne informacije koje se prepuštaju studentima da odrede.

2.2 Unos upita za pretragu (#unos_upita)

Korisniku je potrebno omogućiti da unese upit koji može da se sastoji od jedne reči, više reči razdvojenih razmakom ili kombinaciju reči sa logičkim operatorom. Nakon što korisnik unese upit, program ga automatski parsira i pokreće algoritam pretrage. Način formiranja upita i detalji algoritma su objašnjeni dalje u tekstu.

2.3 Pretraga dokumenata (#pretraga_dokumenata)

Algoritam pretrage iz skupa svih HTML dokumenata izdvaja samo one koji odgovaraju upitu koji je korisnik uneo pretragom stabla reči. U poglavlju 2.6 je objašnjeno kako je upit moguće formirati. Na osnovu ovih informacija, potrebno je izdvojiti reči i operatore iz upita koji će biti ulazni parametri algoritma pretrage.

Detalji implementacije algoritma pretrage i parsiranja upita su prepušteni studentima.

2.4 Rangirana pretraga (#rangirana_pretraga)

Implementirati rangiranje rezultujućih stranica pretrage tako da na rang stranice utiče:

- broj pojavljivanja traženih reči na njoj,
- broj linkova iz drugih stranica na pronađenu stranicu i
- broj traženih reči u stranicama koje sadrže link na traženu stranicu.

Ukoliko korisnik unosi upit sastavljen od više reči, rangiranje stranica po svakoj pojedinačnoj reči utiče na sveukupno rangiranje određene stranice. U ovom slučaju, ne treba insistirati na prisustvu svake od reči u rezultatima, ali bi trebalo bolje rangirati rezultate u kojima se pojavljuju sve reči.

2.5 Prikaz rezultata pretrage (#prikaz_rezultata)

Rezultati upita treba da budu putanje do HTML stranica (iz test skupa) sortirane po izračunatom rangu. Odabir algoritma za sortiranje se prepušta studentima.

Napomena: Studenti samostalno implementiraju odabrani algoritam za sortiranje.

2.6 Paginacija rezultata (#paginacija_rezultata)

Neophodno je obezbediti paginaciju stranica rezultujućeg skupa. Ograničiti broj ispisa po stranici na *N* (stranice sa rednim brojevima od *1* do *N*) uz nuđenje opcije za ispis sledećih *N* stranica (od *N+1* do *2N*) itd. Broj stranica koje se u jednom trenutku prikazuju može se dinamički promeniti od strane korisnika.

2.7 Osnovna upotreba logičkih operatora (#osnovne_skupovne_operacije)

Potrebno je podržati ispravnu upotrebu logičkih operatora AND, OR i NOT prilikom formiranja upita:

- Logički operator **AND** zahteva prisustvo obe navedene reči u stranicama koje su u rezultujućem skupu. Obe reči treba u jednakoj meri da utiču na ukupan rang.
- Logički operator **OR** zahteva prisustvo bar jedne reči u stranicama koje čine rezultujući skup.
- Logički operator **NOT** se u ovom slučaju smatra **BINARNIM**. Dakle, predviđa se upotreba u obliku *'reč1 NOT reč2'* gde se pojavljivanje reči *reč1* zahteva u stranicama rezultujućeg skupa, dok se *reč2* u rešenju ne sme pojaviti.

Primeri:

- python AND sequence
- set OR graph
- dictionary NOT list

Napomena: Upit može imati samo jedan logički operator. Kombinovanje logičkih operatora je objašnjeno u narednom poglavlju.

2.8 Napredna upotrebna logičkih operatora (opciono)

Naprednu pretragu korisnik mora posebno aktivirati (npr. izborom posebne stavke menija). U tom slučaju se umesto AND, OR i NOT ključnih reči koriste operatori iz C-olikih programskih jezika: **&&**, **||** i **!**. Reči AND, OR i NOT se u tom slučaju tretiraju kao bilo koje druge reči. Logičke operatore treba posmatrati kao presek, uniju, odnosno komplement skupa i u skladu sa tim ih implementirati. Prethodno pomenuti operatori se mogu proizvoljno kombinovati u jednom upitu. Sledi nekoliko primera upita praćenih objašnjenjima:

- *! (python && java && clojure)* – prvo je neophodno odrediti presek skupova koji sadrže reči *python*, *java* odnosno *clojure*, a nakon toga odrediti komplement ovog skupa.
- *(dictionary || list || set) && ! tree* – neka skup A predstavlja uniju tri skupa od kojih svaki sadrži po jednu od reči *dictionary*, *list* odnosno *set*. Neka je B skup koji sadrži reč *tree*. Potrebno je odrediti presek skupa A i komplementa skupa B.
- *(dictionary list || set) && ! tree* – ekvivalent prethodnom upitu
- *graph || trie && ! set* - Neka je A komplemet skupa stranica koje sadrže reč *set*. Neka je B skup koji je jednak preseku skupa A i skupa stranica koje sadrže reč *trie*. U tom slučaju, rezultujući skup stranica predstavlja uniju skupa stranica koje sadrže reč *graph* i skupa B.

Napomene:

- Neophodno je voditi računa o priritetima logičkih operator (**!** je većeg prioriteta od **&&**, koji je pak većeg prioriteta od **||**).
- Omogućiti proizvoljno ugnježdavanje zagrada i logičkih operatora.

Da bi se omogućila napredna upotreba logičkih operatora, potrebno je implementirati parser za upite, čije će rezultat biti stablo parsiranja (#parser). Ovo stablo je neophodno evaluirati kako bi se dobio rezultujući skup stranica, nad koijm je potrebno odraditi goreobjašnjeno rangiranje (#evaluacija_stabla_parsiranja).

3. Nefunkcionalni zahtevi

3.1 Detalji implementacije

Mašina za pretraživanje tekstualnih dokumenata treba da bude konzolna aplikacija napisana u programskom jeziku *Python*.

3.2 Ulazni podaci

Posmatraju se samo HTML stranice (datoteke) unutar test skupa.

3.3 Validacija ulaznih podataka

Program ni u jednom trenutku ne sme neočekivano da prestane da radi (na primer, za korenski direktorijum se odabere direktorijum koji sadrži slike). Voditi računa da se korisnik obavesti ukoliko unese upit u formatu koji ovom specifikacijom nije definisan kao validan.

3.4 Strukture podataka

Prilikom implementacije, potrebno je koristiti sledeće strukture podataka:

- HTML stranice test skupa podataka potrebno je organizovati u obliku usmerenog grafa (#graph).
- Sve reči svih HTML stranica iz test skupa podataka potrebno je čuvati u trie stablu kako bi se omogućila efikasna pretraga (#trie).
- Rezultat pretrage jedne ili više reči treba da bude skup HTML stranica (#set).

Napomena: Sve navedene strukture podataka se moraju samostalno implementirati.

4. Raspodela zadataka po studentima

4.1 Student 1

Funkcionalnosti:

1. #parsiranje_skupa_HTML_dokumenata - izgraditi trie stablo u koje će biti smeštene sve reči iz svih HTML dokumenata.
2. #unos_upita - parsiranje upita, određivanje reči i utvrđivanje postojanja nekog od logičkih operatora AND, OR, odnosno NOT.
3. #pretraga_dokumenata - određivanje skupova HTML stranica koje sadrže pojedinačne reči upita.
4. #osnovne_skupovne_operacije - primena osnovnih skupovnih operacija preseka, unije, odnosno komplementa i utvrđivanje rezultujućeg skupa stranica.

Strukture podataka:

1. #trie - pored odgovarajućih klasa kojima će se predstaviti *trie* stablo, potrebno je implementirati funkcije za dodavanje reči u stablo, odnosno pretragu reči u stablu.

4.2 Student 2

Funkcionalnosti:

1. #parsiranje_skupa_HTML_dokumenata - izgraditi graf u kome su čvorovi predstavljeni HTML stranicama, a grane predstavljene linkovima između HTML stranica.
2. #rangirana_pretraga – za sve stranice rezultujućeg skupa HTML stranica odrediti rang.
3. #prikaz_rezultata - na osnovu prethodno određenog ranga, neophodno je sortirati stranice samostalno implementiranim algoritmom za sortiranje. Prikazati putanje do ovako sortiranih stranica i navesti rang svake.
4. #paginacija_rezultata – prikaz *N* stranica odjednom, kao i mogućnost kretanja napred odnosno nazad za *N* stranica. Korisniku dati mogućnost da dinamički promeni *N*.

Strukture podataka:

1. #graph - implementirati operacije za dodavanje HTML stranice u graf, nakon čega je neophodno povezati odgovarajuće čvorove grafa u skladu sa ulaznim, odnosno izlaznim linkovima.
2. #set - implementirati operacija za uniju, presek i komplement skupa

Napomena: Studenti treba da objasne svaki algoritam koji je implementiran. Objašnjenje treba da bude u vidu prikaza pseudokoda, formule ili kratke prezentacije.

4.3 Dodatni poeni

Za dodatne poene potrebno je implementirati složene skupovne operacije (tačka 2.8). Prvi student treba da implementira #parser, a drugi #evaluacija_stabla_parsiranja.

Napomena: Ukoliko se samo jedan od studenata odluči da radi naprednu upotrebu logičkih operatora, tada treba da implementira stavku #parser.

5. Upotreba Git-a

Obavezno je korišćenje Sistema za kontrolu verzija *git*. Kao udaljeni repoziorijum, može se koristiti Github ili Gitlab. Neophodno je da projekat bude u privatnom repozitorijumu, na koji će predmetni asistent biti dodat kao *collaborator*/*reporter*. Prilikom *commit*-ovanja izvornog koda, neophodno je na početku poruke navesti oznaku stavke koju je student delimično ili u potpunosti implementirao, npr.

git commit -m “#trie Implementacija trie-a”

U okviru ovog dokumenta, pored svake funkcionalnosti, nalaze se i oznake koje treba koristiti prilikom *commit*-ovanja.

Commit-i čija poruka ne počinje oznakom stavke, biće ignorisani.