

TEMA:

Mašina za pretraživanje tekstualnih dokumenata

Predmet: Osnovi informacionih sistema i softverskog inženjerstva

Studenti na projektu:

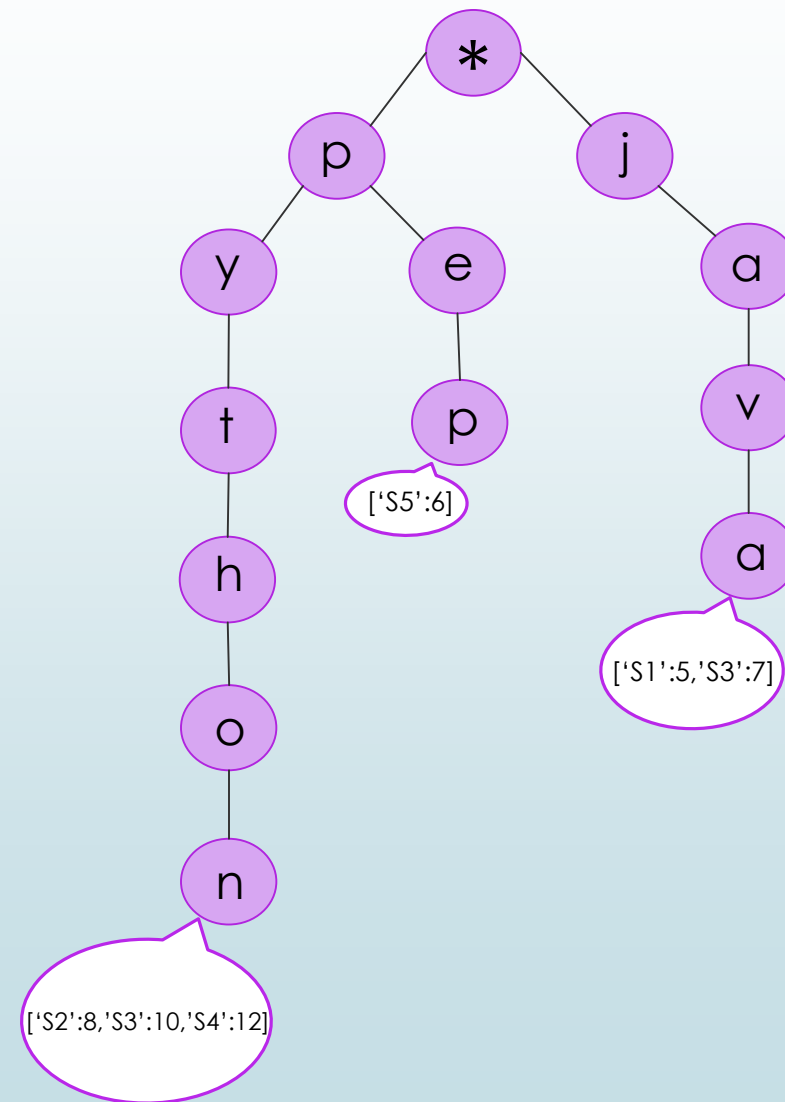
Sladana Savković (student 1)

Dragana Čarapić (student 2)

Strukture podataka

1. #trie

- Svaki čvor stabla sadrži: sadržaj čvora tipa *char*, listu svojih potomaka, *skup(#set)* u formatu *stranica:broj_riječi*, gdje je ključ link stranice koja sadrži zadatu riječ, a vrijednost broj pojavljivanja riječi na stranici
- Čvorovi koji ne predstavljaju kraj riječi sadrže prazan skup.
- Svi karakteri, odnosno riječi se u stablo dodaju malim slovima da bi omogućili *case insensitive* pretragu





➤ Klasa koja implementira stablo definiše metode:

add(self, word, link) dodavanje nove riječi (word) na stranici sa putanjom link

find_word(self, word) pronalazak riječi (word) u stablu

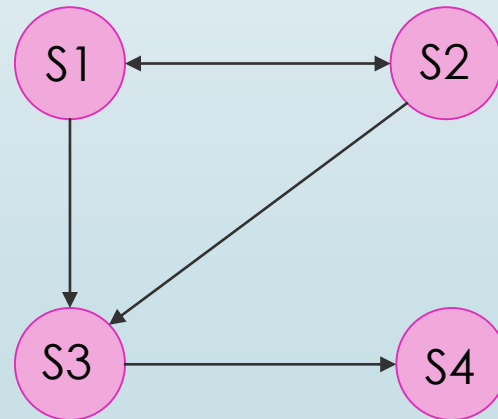
breath_first(self) ispis čvorova stabla sa obilaskom po širini

check_depth(self) provjera da li je stablo prazno

find_word_document(self, word_list, path) broj pojavljivanja svake riječi iz liste word_list na stranici sa putanjom path

2. #graph

- Usmjeni graf je implementiran kao rječnik. Ključevi rječnika (čvorovi grafa) su putanje html stranica tj. linkovi stranica. Vrijednosti u rječniku (grane grafa) su izlazne grane čvorova, tj. putanje html stranica koje linkuje stranica u čvoru grafa. Informaciju o ulaznim granama čvora tj. stranice koje linkuju stranicu koja se nalazi u čvoru, dobijamo tako što za svako pojavljivanje te stranice u granama grafa, registrujemo čvor te grane.



```
'S1' : ['S3', 'S2']  
'S2' : ['S3', 'S1']  
'S3' : ['S4']  
'S4' : []
```



➤ Klasa Graph sadrži sledeće metode:

vertices(self) vraća listu svih čvorova

ret_edge(self, v) - vraća listu svih grana čvora *v*

add_vertex(self, v) metoda za dodavanje novog čvora u graf

add_edge(self, v, e), metode za dodavanje svih grana (*e*) čvora *v*

links_for_rank(g, path) metoda koja vraća sve linkove stranica koje linkuju zadatu stranicu, parametar *path* je putanja zadate stranice

3. #set

- Set je implementiran kao rječnik. Ključevi rječnika su putanje html stranica. Vrijednosti rječnika su inicijalno postavljene na nulu, a zapravo predstavljaju broj traženih riječi koje odgovaraju unesenom upitu.
- Klasa Set sadrži sledeće metode:
 - `__or__(self, other)` metoda u kojoj je implementirana operacija za uniju
 - `__and__(self, other)` metoda u kojoj je implementirana operacija za presjek
 - `__sub__(self, other)` metoda u kojoj je implementirana operacija za razliku
- Klasa Set sadrži i ostale pomoćne metode za dodavanje nove stranice (`add(self, key, value)`), metoda koja vraća sve ključeve (`ret_key(self)`), metoda koja vraća sve vrijednosti (`ret_all_val(self)`) itd.

Funkcionalnosti

#parsiranje_skupa_HTML_stranica

- Sadrži metode:

make_tree_and_graph(f) kreira stablo i graf pozivajući sledeću metodu

recursive_walk(f,p,g,root) rekurzivna metoda koja mijenja svoje neprimitivne tipove parametara praveći stablo i graf i istovremeno prolazi kroz sve foldere i fajlove prosljeđene putanje

#unos_upita

- Koristi se za parsiranje ulaznog upita i prepoznavanje unesenih riječi i operatora uz identifikaciju greški prilikom unosa
- Sadrži metodu:

parsiraj_upit(upit) metoda za parsiranje čije su povratne vrijednosti operator, riječi odnosno None, None u slučaju pogrešnog unosa

#pretraga_dokumenata i #osnovne_skupovne_operacije

- Primjena osnovnih skupovnih operacija (presjek, unija, komplement) i utvrđivanje skupa HTML stranica koje zadovoljavaju uneseni upit
- Sadrži metodu:
pretraga_dokumenta(root, words, operator, graph) vraća rezultujući skup stranica

#paginacija_rezultata

- Sadrži metodu:
paginacija(list, n) sa parametrima: lista stranica iz rangirane pretrage i broj stranica koje trebaju biti prikazane. Broj stranica za prikaz nije fiksni i može se promijeniti u toku paginacije

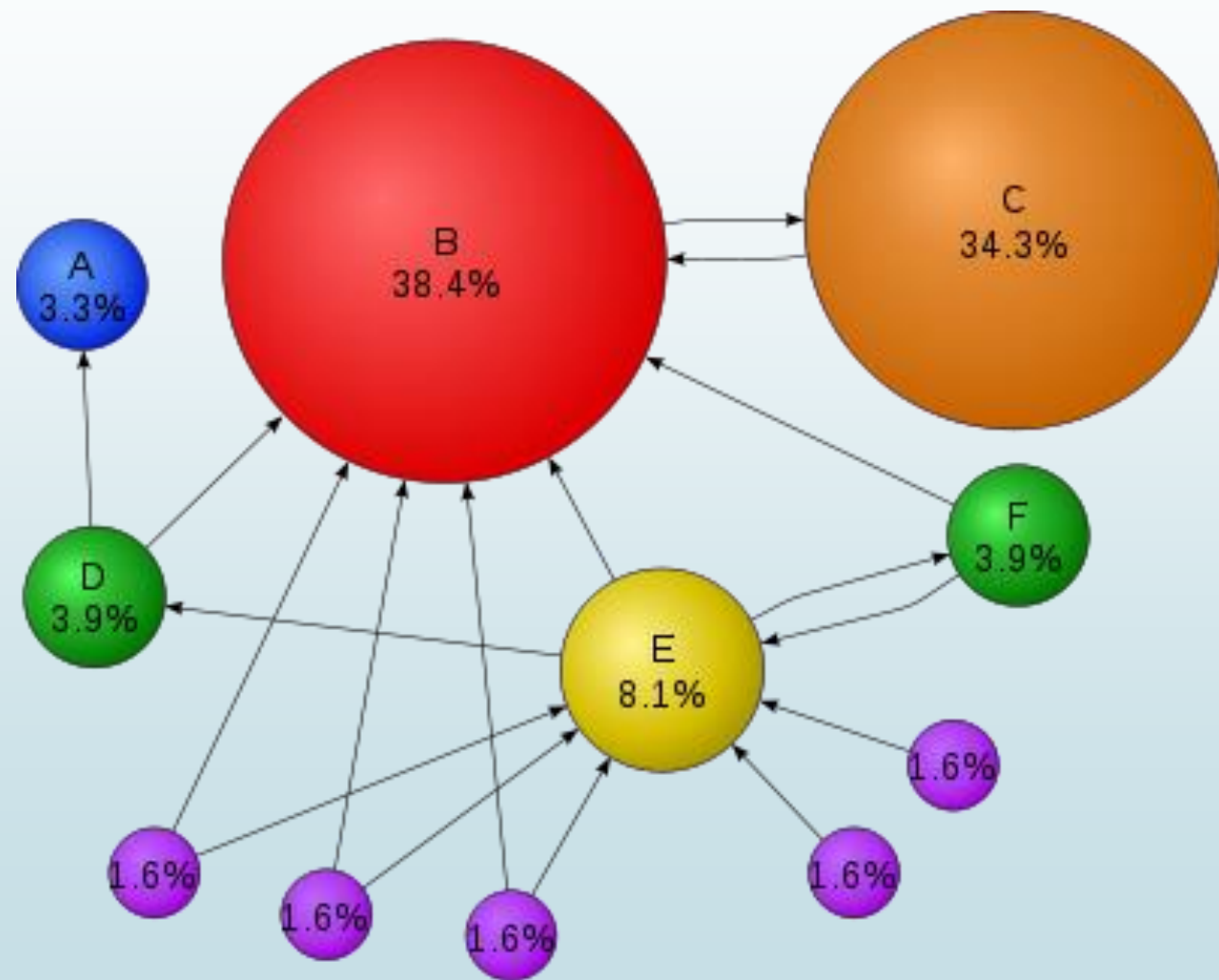
#prikaz_rezultata

- Korisniku je dozvoljeno da izabere način sortiranja - po opadajućem i rastućem redoslijedu ranga. Za sortiranje je korišten algoritam *Quick sort* zbog brzine izvršavanja $T(n)=O(n)$.

#rangirana_pretraga

Algoritam se zasniva na tri koraka koja određuju rang stranice (značaj). Stranica na koju pokazuje dosta značajnih stranica je i sama značajna i raste njen rang.

Na slici je stranica prikazana čvorom, a njeng rang procentom. Stranica B je najznačajnija jer na nju pokazuju takođe značajne stranice, za razliku od stranice E na koju pokazuje svega 1 stranica manje ali njihov značaj je mali.



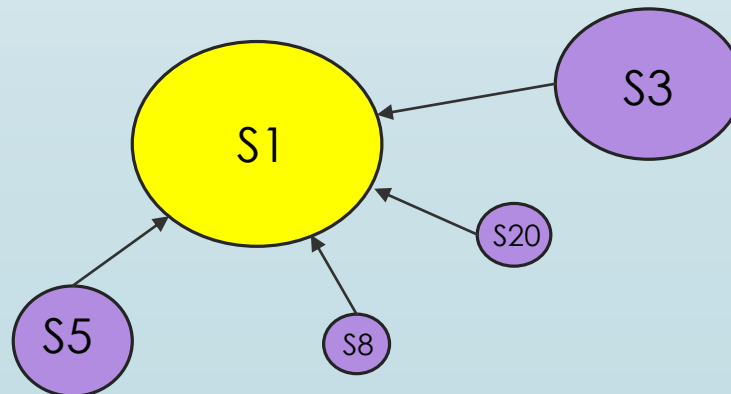
Kriterijumi rangiranja po značaju na rang:

1. Broj traženih riječi u stranicama koje sadrže link na traženu stranicu

- Stranica koja linkuje traženu stranicu je značajna ukoliko odgovara unesenom upitu. Međurang po prvom kriterijumu ($r3$ u kodu) predstavlja zbir riječi iz upita koje se nalaze u značajnim stranicama koje linkuju traženu stranicu.

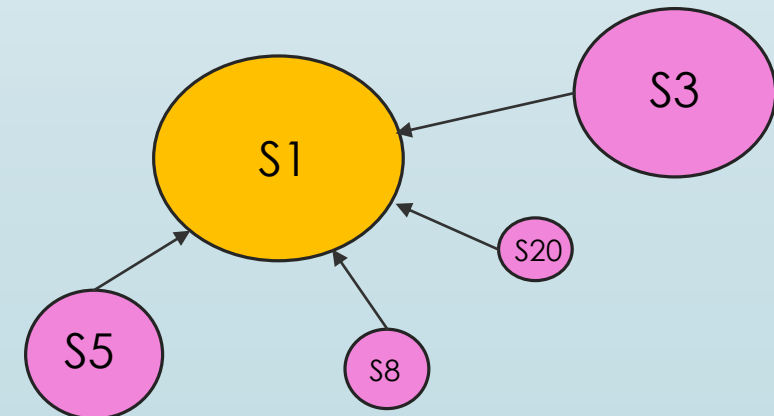
Java AND Python

- riječi: Java, Python
 - posmatrana stranica: S1
 - stranice koje linkuju S1: S3, S5, S8, S20
 - stranice koje sadrže obe riječi upita (**značajne**): S3, S5
 - stranice koje neodgovaraju upitu (bezznačajne): S8, S20
- $r3 = (\text{broj riječi u S3}) + (\text{broj riječi u S5})$**



Python programming language

- riječi: Python, programming, language
 - posmatrana stranica: S1
 - stranice koje linkuju S1: S3(3 riječi upita), S5(2 riječi), S8(1 riječ), S20(0 riječi)
 - Množenjem sa konstantom povećava značaj stranica koje sadrže više riječi upita
- $n_i = \text{const} * \text{broj_riječi_stranice_}S_i$
- $r3 = n3 + n5 + n8 + n20$**



2. Broj pojavljivanja traženih riječi na stranici

- Broj riječi se računa u toku primjene skupovnih operacija u zavisnosti od vrste operatora u upitu. (*r1* u kodu)

AND (*presjek*): **$r1 = \text{br_pojavljivanja_prve_riječi} + \text{br_pojavljivanja_druge_riječi}$**

NOT (*razlika*): **$r1 = \text{br_pojavljivanja_prve_riječi}$**

OR / unos bez operatora (*unija*):

$r1 = \text{ukupan_br_pojavljivanja_svih_riječi_upita} * \text{broj_nenultih} / \text{broj_riječi}$

broj_nenultih – broj riječi upita koje se pojavljuju barem 1 na određenoj stranici


broj_riječi – broj riječi koje sadrži upit

Na ovaj način se bolje rangiraju stranice koje sadrže sve riječi upita.

3. Broj linkova iz drugih stranica na pronađenu stranicu

- Primjena metode *links_for_rank(g, path)* koja vraća skup linkova stranica koje linkuju zadatu stranicu. (*r2* u kodu)

$r2 = \text{broj_pronađenih_linkova}$

- 
- Međurangovi r_1 , r_2 , r_3 su skalirani prema važnosti u sledećem opsegu:
 - opseg r_1 : 0 – 30
 - opseg r_2 : 0 – 20
 - opseg r_3 : 0 – 50
 - Rang *jedne* stranice iz rezultujućeg skupa stranica se računa kao:

$$r = r_1 + r_2 + r_3$$