

# TEMA:

# Mašina za pretraživanje tekstualnih dokumenata

Predmet: Osnovi informacionih sistema i softverskog inženjerstva

## **Studenti na projektu:**

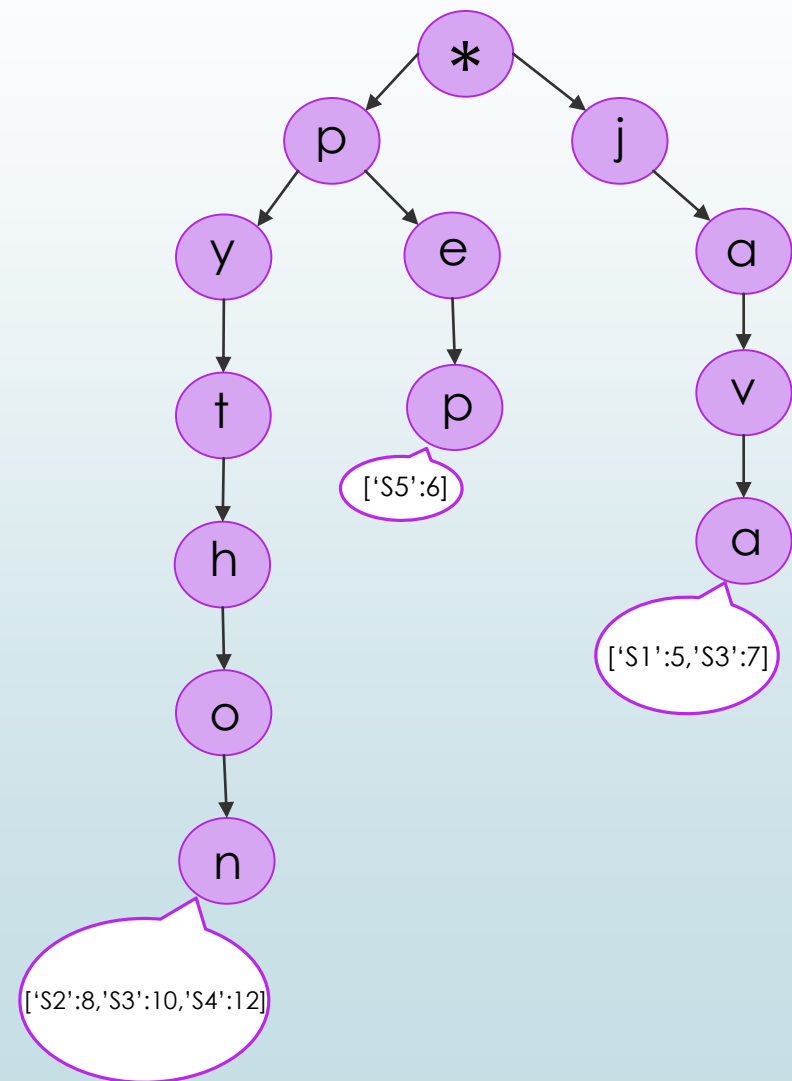
Sladana Savković (student 1)

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

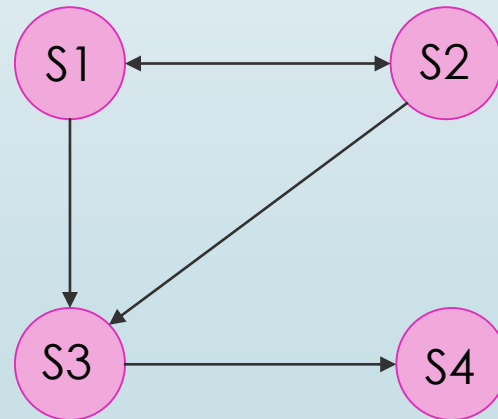
*breath\_first(self)* ispis čvorova stabla sa obilaskom po širini

*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

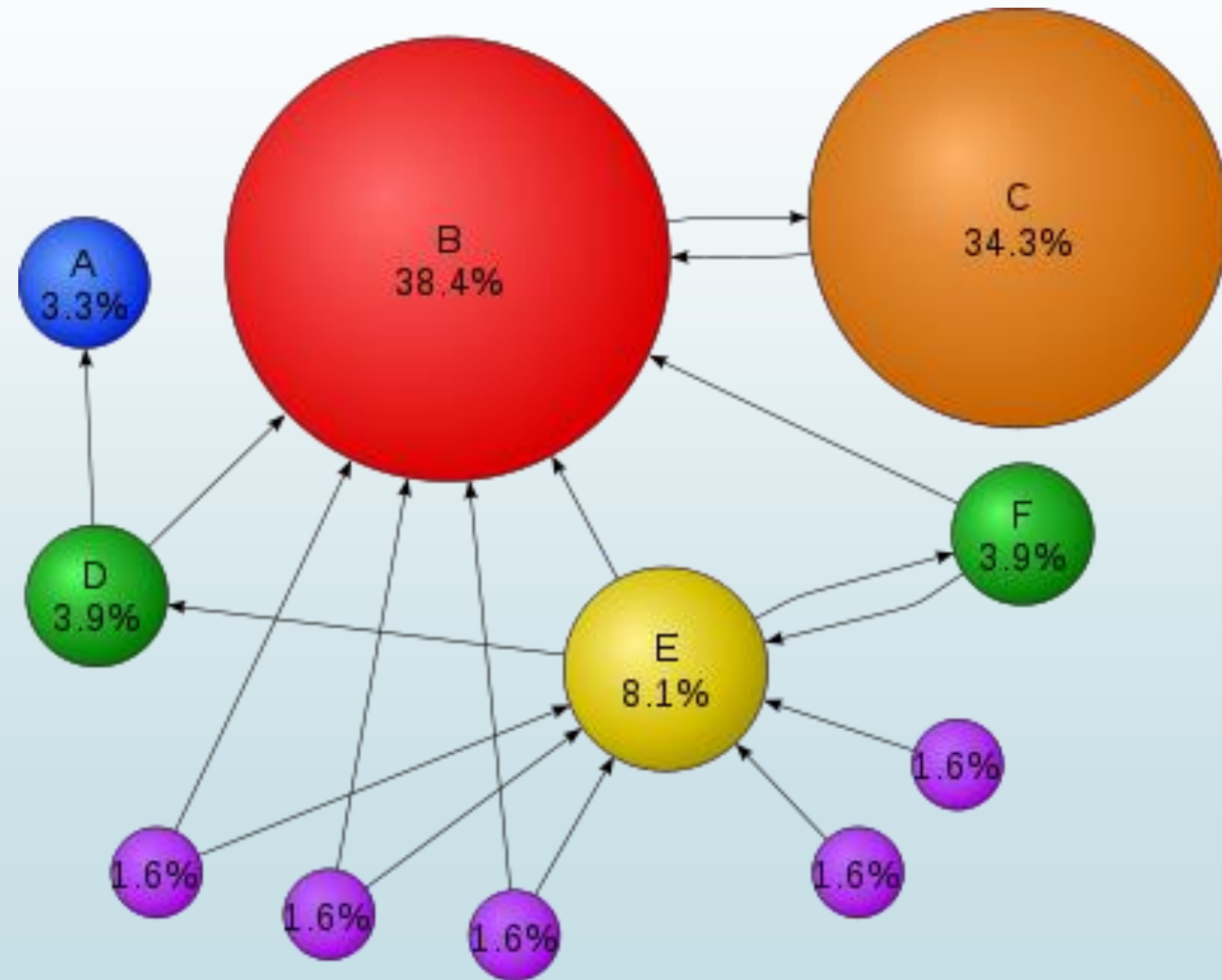
- Korisniku je dozvoljeno da izabere način sortiranja - po opadajućem i rastućem redoslijedu ranga. Za sortiranje je korišten algoritam *Quick sort* zbog brzine izvršavanja  $T(n)=O(n)$ .



# #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.



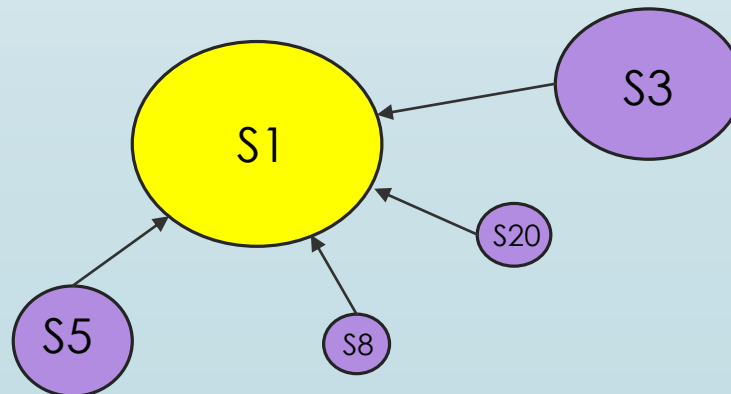
## Kriterijumi rangiranja po značaju na rang:

### 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 ( $r_3$  u kodu) predstavlja zbir riječi iz upita koje se nalaze u značajnim stranicama koje linkuju traženu stranicu.

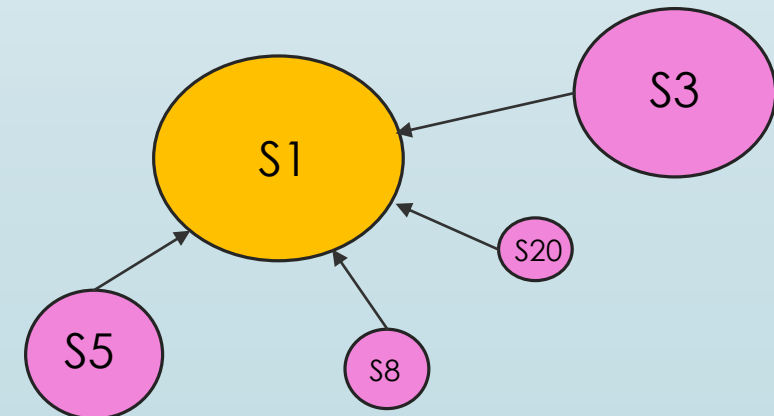
#### Java AND Python

- 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
- $r_3 = (\text{broj riječi u S3}) + (\text{broj riječi u S5})$**



#### 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
- $n_i = \text{const} * \text{broj\_riječi\_stranice\_}S_i$
- $r_3 = n_3 + n_5 + n_8 + n_{20}$**



## 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:

$$r = r_1 + r_2 + r_3$$