**Objasnjenje koriscenih algoritama**

**Opis strukture**

Pretrazivac se sastoji od 5 fajlova, 4 modula i glavnog main fajla:

-main.py (sadrzi glavnu logiku interakcije sa korisnicima)

-parser.py (predefinisan)

-graph.py (sadrzi klasu i funkcije za rad sa grafom)

-trie.py (sadrzi klasu i funkcije sa rad sa trie-em)

-set.py (sadrzi klasu koja predstavlja skup stranica)

**#Trie**

Fajl koji sadrzi dve klase- Trie i TrieNode.

TrieNode sadrzi promenljivu slovo, niz svih TrieNode-ova.

Trie klasa nam sluzi za cuvanje dictionary-ja.

Ubacuje reci u drvo stavljajuci slovo po slovo u odgovarajuci node (mapa- slovo, bool), trazi reci od root to true (whole) tj. moraju biti cele reci, krece od korena i ide sve dok ne nadje node sa true(space) vrednosti, ako ne nadje taj nod, racuna se da rec nije pronadjena, jer nije cela i ne prepoznaje.

**#Pretraga\_dokumenata**

Za pretragu koristimo klasu set, koja ima definisan all\_files(lista svih apsolutnih vrednosti koje su nadjene da sadrze trazenu rec) i logic\_operator promenljivu (koja sadrzi vrednost za logicke operatore definisane u pretrazi). Nakon sto korisnik unese string za pretragu, sa jednom ili vise reci, ili sa operatorima OR AND i NOT, pokrece se metoda generate, koja prepoznaje logicke operatore, ako ih ima, i postavlja vrednost u logic\_operator, i ostatak stringa deli u elemente niza. Za svaki element tog niza se vrsi pretraga tako da u all\_files kao key vrednost se koristi trazena rec, a value je niz svih apsolutnih putanji gde se ta rec nalazi. Zatim se na all\_files primenljuje funkcija custom\_set, koja grupise sve iste apsolutne putanje, tako da imamo vezu rec-putanje-i broj ponavljanja te reci u tom fajlu.

**#rangirana pretraga**

Rang smo racunali pomocu sledece formule:

rang= broj\_ponavaljanja\_trazene\_reci\_u\_fajlu / broj\_reci\_u\_tom\_fajlu \* broj\_grana(linkova)\_koje\_pokazuju\_ka\_tom\_fajlu\_u\_grafu

Zatim ga zaokruzujemo na 10 decimala, i sortiramo.

**#Parsiranje\_skupa\_HTML\_dokumenata**

Dobija se apsolutna putanja od korisnika ili se bira test folder, zatim se poziva look\_file\_walker, koja prima putanju,prazan trie,prazan raw graph i prazan file\_content\_size (direktorijum u koji se smesta putanja ka html fajlu i koliko taj fajl ima reci ). Zatim se ide kroz direktorijume, u dubinu, i traze se reci koje se zavrsavaju na 'html', a svaki html fajl se definise njegova putanja koja se prosledjuje u parser. Parser vraca links (niz svih linkova koji izlaze iz tog html fajla) i words (cuva sve reci u tom fajlu). Popunjavamo direktorijum raw\_graph po principu key-apsolutna putanja tog fajla, valuesniz izlaznih linkova (grana tog cvora). Za svaki karakter iz reci se ubacuje u trie.

**#Unos\_upita**

Korisniku se nudi meni gde odgovarajucim izborom dobija opciju unosa upita koji ce se dalje pretrazivati.

**#Graph**

Sluzi da bi se odredio stepen cvora(broj dolaznih linkova), koji sluzi za rangiranje