Projekat 2 iz algoritama i struktura podataka

Strukture koje su koriscene u projektu su: **grafovi, Trie** i algoritam za sortiranje **Merge sort**.

Pocetak programa se zasniva na tome da mozemo da biramo putanju od kog direktorijuma (foldera) zelimo da da izvrsimo pretragu reci u html file-ovima. Zbog toga, pravimo listu putanja kako dolazimo do tih file-ova. Nakon toga se parsiraju ti fajlovi i te fajlove stavljamo u cvorove grafa, a grane (ivice) predstavljaju linkovi koji ih povezuju.

Sto se tice pretrage teksta, moze se uneti jedna rec, tekst sastavljen od vise reci i moze se upotrebiti logicki operatori AND, OR i NOT.

Program prepoznaje koji je tip od ovih 5 primera i nakon toga se poziva funkcija za svaki slucaj.

Funkcija za prvi slucaj (u programu: racunanje\_vr\_jedna\_rec) je glavna, a ostali problemi se rade pozivajuci se na nju, uz malo korekcija.

U toj funkciji prvo uz pomoc Trie vraca listu trazene reci (ako vrati praznu listu nema te reci i automatski izlazi iz pretrage jer nema trazene reci) i zahvaljujuci tome se moze izrazunati broj trazene reci sto je bitno za heuristiku.

Medjutim, ne uticu samo broj trazene reci na heuristiku nego i broj linkova i reci u njima, pa se uz pomoc grafa i par funkcija vezane za graf moze izracunati heuristika za svaki slucaj.

Heuristika je recnik, ciji je kljuc putanja fajla, a vrednost je broj koji predstavlja vrednost heuristike.

Kada se zavrsi funkcija (u programu: racunanje\_vr\_jedna\_rec), sledi ispis fajlova sa najvecom heuristikom (funkcija ispis\_file(heuristika\_recnik,tekst koji se trazi)).

Tu se postavlja pitanje kako preko recnika mozemo da sortiramo vrednosti?

Odgovor je da sve vrednosti heuristike stavimo u niz (lista u pajtonu) i to da sortiramo.

Najbolji i najbrzi algoritam za ovo resenje je **Merge sort**.

Njegova vremenska slozenost u je **O(n log n)** i zasniva se na deljenju niza (liste) na dva dela i sortira se koristeci rekurziju.

Taj algoritam je najbolje resenje jer ima najbolju vremensku slozenost (pored Quick sorta). Isto tako memorijska slozenost mu je O(n) i stabilan je algoritam.

Razlog sto sam se opredelio za Merge sort pre Quick sorta je jer je u najgorem slucaju vremenska slozenost Quick sorta O(n2), ali prednost Quick sorta je sto je manja memorijska slozenost. (Njegova je O(log(n)).

Nakon sto smo sortirali niz (listu), ispisemo sve putanje od trazenih fajlova.

Ako su u tekstu logicki operator and,or,not , string splitujemo, dobijemo reci koje su trazene i izracunamo heuristike za te dve reci. (poziva se funkcija za racunanje\_vr\_jedna\_rec)

Kada je logicki operator AND, poredimo u te dve heuristike ako su im isti kljucevi stavljamo u novi recnik, cija je vrednost zbir vrednosti heuristike za prvu i drugu rec. Slicnom logikom se resava i slucaj sa OR i NOT.

Sto se tice teksta sa vise reci (npr python programming language), taj tekst splitujemo po space, pozivamo funkciju racunanje\_vr\_jedna\_rec i slicnim postupkom kao i sa logickim operatorima dolazimo do resenja.