Drugi kolokvijum iz Upravljanja Informacijama
Grupa 1

Zadatak 1:

Pokrenuti Lucene alat (Lucene folder) pomoću Eclipse-a. Indeksirati dokumente koji se nalaze u “data” direktorijumu, i zatim izvrsiti pretrage koje vracaju:

a) Sve dokumente koji sadrže reč “king” ali ne i reč “nelson”
b) Sve dokumente koji imaju reči koje u sebi sadrže reč “king” (ako dokument sadrži reč kings, na primer, treba da bude prikazan)
c) Sve dokumente koji sadrže frazu “king of united states”, ali ako se nađu još 3 (ili manje) reči unutar fraze (npr. “king of the new united states), dokument takođe treba prikazati.
d) Promeniti IndexFiles.java klasu tako da se kreiraju 2 indeksa, jedan sa naslovom članka (prva linija svakog .txt fajla), a drugi sa sadržajem članka (ostatak teksta u fajlu). Zatim napraviti upit koji vraća sve dokumente koji sadrže reč “deceased” i u naslovu i u sadržaju članka.

Same upite možete napisati u tekstualnom dokumentu resenja.txt, i dokument sacuvati u Lucene folderu

Zadatak 2:

Za pokretanje Hadoop okruženja otvorite folder “hadoop_setup” i pokrenite docker compose up. Ovde vam se nalazi i tekstualni fajl sa potrebnim komandama za rad sa hadoop-om. Podatke za zadatke (kao i mesto gde treba da sačuvate vaše .py skripte) imate u folderu MapReduce

a) Lak zadatak:

U “data” fajlu se nalaze informacije o različitim hranama u vidu csv fajla (nayiv hrane u prvoj koloni i grupa kojoj ta hrana pripada u trećoj koloni). Sumirajte koliko stavki pripada svakoj od navedenih grupa hrana.

b) Srednji zadatak:

U “data” fajlu se nalaze informacije o različitim “ukusima” i grupi kojoj oni pripadaju. Zadatak je da na kraju za svaku grupu ispišete broj ukusa koji joj pripada, kao i ukus sa najdužim i najkraćim nazivom (ako ih ima više sa istim nazivom napišite bilo koji). Ako se u datoteci “data” nalaze neispravni podaci, drugim rečima ako jedna od dve kolone ne postoji ili ima NULL vrednost, takvi podaci treba da se preskoče.

c) Težak zadatak:

U “data” fajlu se nalaze informacije o cenama deonica za ordeđen datum. Pored datuma imate naziv deonice, i njenu cebu tog dana. Na izlazu map-reduce algoritma treba da za svaku deonicu napišete najveći “day-to-day” skok i najveći “day-to-day” pad.

Za LMT deonicu treba da imate uspis LMT: 62, -25

Obratite pažnju da je moguće da se desi da deonica samo raste, ili samo pada. U tom slučaju treba naznačiti sa bilo kojim specijalnim karakterom da ne postoji najveći skok Ili najveći pad.

Zadatak 3:

U folderu KafkaStreaming se nalazi docker compose za pokretanje kafka okruženja i tekstualni fajl sa komandama potrebnim za rad sa kafka brokerom i faust aplikacijom.

Tu vam se nalaze i folderi sa spremljenim consumerima i producerima za lak, srednji i težak zadatak. U tim istim folderima treba da ostavite i vašu .py skriptu za faust aplikaciju koja će da obrađuje podatke unutar kafka *topic*-a. SVI *topic*-i koji se koriste u naredna tri primera nisu kreirani unutar kafka brokera, te to treba da uradite sami pre nego što pokrenete faust aplikaciju za određen primer. Instrukcija kako to da uradite se nalazi u fajlu sa komandama.

a) **Lak zadatak:**
   Napraviti Faust filter koji će u realnom vremenu brojati:
   - Ukupnu količinu majica za svaku od veličina (XL, L, M, S)
   - Ukupnu količinu majica za svaki od brendova (Adidas, Nike, Polo, Uniqlo)

   Te agregirane podatke ne treba nigde da šalje samo ih ispišite na standardni output

b) **Srednji zadatak:**
   Napraviti Faust filter koji će da u realnom vremenu tangled sve porudžbine (poruke) i:
   - Ako je veličina majice XXL (koju prodavnica ne prodaje više), porudžbina treba da se preusmeri na consumer_non_fulfilled_orders servis
   - Ako je bilo koja druga veličina porudžbina ide do consumer_shipping_shirts servisa

c) **Težak zadatak:**
   Servis 'Zahtev za termin' periodično na *topic* sa nazivom "zahtevi_za_termine" šalje poruku za zauzimanje jedne od bolničkih sala. Zahtev se šalje u vidu JSON-a sa sledećim poljima:

   {
   id_zahteva: string,
   broj_sale: integer
   }

   Može da zauzme salu tek kada svi servisi koji postoje u bolnici odobre zahtev. Servis koji šalje zahteve za zauzimanje sale ne zna koji sve servisi treba da odobre salu, i očekuje da se na topic-u sa nazivom "odgovori_na_zahteve" pojavi poruka u formatu:

   {
   id_zahteva: string,
   zahtev_odobren: boolean
   }

   Svaki od 3 servisa unutar bolnice (uprava bolnice, hirurski departman, odrzavanje) primaju zahteve za salu i nakon određene pauze (između 1 i 5 sekundi) vraćaju svoj odgovor u formatu:

   {
   id_zahteva: string,
   zahtev_odobren: boolean
   }

   na topic sa nazivom "odgovori_na_zahteve".

   Zadatak faust aplikacije je da:

   1) akumulira odgovore koje salju svaki od 3 bolnička servisa, i nakon toga pošalje konačni odgovor servisu (taj odgovor ne morate da primate uopše nigde, samo je bitno da se pošalje). Zahtev je odobren jedino ako ga sva tri bolnička servisa odobre.

   2) Sve dok traje odobravanje zahteva za salu sa određenim brojem, faust aplikacija treba automatski da odbije svaki zahtev koji traži salu sa istim brojem.

NAPOMENA: Servisi koje ste dobili (cetiri .py skripte) koriste 2 kafka topic-a ("zahtevi_za_termine" i "odgovori_na_zahteve"). Ako je potrebno za resenje zadatka, možete kreirati nove topic-e, ili brisati postojece.

Skripta "primljeni_odgovori.py" sluša na topic-u "odgovori_na_zahteve" i samo ispisuje odgovore na std.out. Ona je tu samo da vam olakša validiraciju vašeg rešenja. Ako vidite potrebu, mozete promeniti i u ovoj skriptni koji topic sluša za odgovore.