3η Άσκηση Διαχείριση Σύνθετων Δεδομένων

Μάθημα	Διαχείριση Σύνθετων Δεδομένων (ΜΥΕ041)
Διδάσκων	Μαμουλής Νικόλαος
Ονοματεπώνυμο	Καζακίδης Θεοχάρης
ΑΜ	4679
Εξάμηνο	Εαρινό
Ακαδημαϊκό Έτος	2022-2023

Πως τρέχουμε τα προγράμματα?

→Άσκηση_3

- Όνομα Αρχείου: Askisi3_Kazakidis_Theocharis_4679.py

  - Εντολή: python3 Askisi3_Kazakidis_Theocharis_4679.py 5

    Όπου το k = 5

Τι τυπώνει?

→

```
chareskazakides@192 Diax_Dedom_3 % python3 Askisi3_Kazakidis_Theocharis_4679.py 5

Number of sequential accesses= 3018
Top k objects:
50905: 14.84
85861: 14.76
22652: 14.74
75232: 14.74
20132: 14.74

Brute - Force
Number of sequential accesses= 300000
Top k objects:
50905: 14.84
85861: 14.76
75232: 14.74
22652: 14.74
20132: 14.74
chareskazakides@192 Diax_Dedom_3 %
```

Κύριες συναρτήσεις επεξήγηση:

→ Άσκηση_3

Το πρώτο μέρος αποτελείται από τρείς συναρτήσεις.

- **def readR() :** Συνάρτηση που η δουλεία της είναι να διαβάζει το αρχείο rnd.txt και να παίρνει τα δεδομένα τις κάθε γραμμής του ξεχωρίζοντας τα πρώτα με κενό και να προσθέτει τα score που πήρα σε μια λίστα των score R.

- **def process(line, fileId) :** Συνάρτηση που η δουλεία της είναι να παίρνει ως όρισμα την γραμμή και το fileId για να ξέρει πιο αρχείο διαβάζει,στην αρχή παίρνει τα δεδομένα ξεχωρίζοντας τα μέσω tokens και έτσι παίρνει το id και το score της κάθε γραμμής. Μετά ελέγχει άμα το objectid που διάβασε είναι μέσα στο λεξικό lowerScore και άμα είναι προσθέτει στο objectScore την τιμή του objectScore του objectid που έλεγξε και όταν γίνει αυτό τα πετάει έξω από το λεξικά lowerScore και readFrom. Άμα το flag isHeapReady είναι false η άμα το κατώφλι Τ είναι μεγαλύτερο από το ελάχιστο στοιχειο της λίστας σωρού τότε προσθέτει στο objectScore την τιμή του του score του αντικειμένου με objectId. Έπειτα άμα το myHeap είναι μικρότερο από το top k(top αντικείμενα με το μεγαλύτερο score)προστίθεται μια λίστα με τα τα score και id στη λίστα σωρό. Αλλιώς άμα δηλαδή το k είναι ισο με τη λίστα σωρό τότε ελέγχω αν το flag isHeapReady και μετατρέπω την λίστα σωρό myHeap με την βοήθεια της συνάρτησης heapq.heapify(myHeap) σε σωρό. Τότε κάνει το flag isHeapReady true. Μετά στην λίστα heapIds αποθηκεύονται τα id της λίστας σωρού και checkarei άμα το object id βρίσκεται μέσα στα heapid, άμα δεν είναι τότε ελέγχει άμα το top στοιχείο της myHeap είναι μικρότερο από το objectScore και άμα είναι τότε το πετάει από την myHeap με την heapq.heappop και προσθέτει με την heapq.heappush την λίστα [objectScore, objectId]. Αν το objectid είναι ήδη στο heapids βρίσκει το που είναι στο myHeap τα πετάει έξω και το αποθηκεύει στο item όπως και το objectscore του όπου τα αποθηκεύει σε μια άλλη λίστα items. Άμα δεν είναι τότε "pusharei" τα δεδομένα που πήρε στο σωρό.

- **def checkStop() :** Συνάρτηση που η δουλεία της είναι συγκρίνει αρχικά άμα το κατώφλι είναι μεγαλύτερο από το top στοιχείο του myHeap, άμα είναι "επιστρέφει" false γιατί δεν το θέλω αλλιώς παίρνει τα κλειδιά από το λεξικό lowerScore και βλέπει άμα το λεξικό είναι 0 τότε επιστρέφει 0 γιατί όλα τα χρησιμοποιήθηκαν. Μετά μέσω μιας for ,για κάθε k αντιστοιχεί το fileid με το λεξικό readfrom για κάθε k και άμα το ένα αρχείο είναι 0 κάνει το άλλο 1.Τέλος βρίσκει το upperbound "προστίθοντας" την τιμή του lowerscore για κάθε k και την τιμή του score για το άλλο αρχείο για να ξέρει που είναι το πάνω όριο και άμα αυτό είναι μεγαλύτερο από τη top τιμή του heap γυρνάει false γιατί δεν το θέλω.

- **def algorithm() :** Συνάρτηση που η δουλεία της είναι να διαβάζει τα αρχεία και μέσω ενός counter κρατάω πόσες φορές μπήκε μέσα στην επανάληψη δηλαδή πόσες γραμμές διάβασα όπου τον βοηθάνε 2 flag(finished) και flag όταν ενεργοποιηθούν τότε αυξάνεται για να τερματίσει αλλιώς καλεί την process() με το line και το αριθμό του αρχείου για να ξέρω ποιο αρχείο την κάλεσε και όταν "ενεργοποιηθεί" η συνάρτηση checkStop τότε σταματάει και τυπώνει τα αποτελέσματα

- **def readB(filename) :** Συνάρτηση που η δουλεία της είναι να ανοίγει το αρχείο που του δίνω (filename) και έπειτα μέσω της readline το διαβάζει όπου το "σπλιταρει" σε tokens μέσω του κενού και κρατεί το obgectid και το objectscore. Ακόμη στην λίστα R ενημερώνει το αντίστοιχο score στο αντίστοιχο id και όταν είναι κενό το line τερματίζει.

- **def bruteforce():** Συνάρτηση που η δουλεία της είναι να καλεί την readR για να διαβάσει όπως προανέφερα το rnd.txt και και την readB 2 φορές μία για το seq1 και μια για το seq2. Όσο είναι το πλήθος των k κρατάει το object id και objectscore όπου τα προσθέτει στην αρχικά κενή λίστα σωρού myHeap. Έπειτα καλείται η heapq.heapify(myHeap) για να κάνει την λίστα σωρό και μετά συγκρίνει το top στοιχείο του σωρού με το objectscore και άμα είναι μεγαλύτερο το objectscore τότε για να το συμπεριλάβω στα top k στοιχεία πετάω έξω το μικρό στοιχείο από το myheap με την heapq.heappop και βάζω το καινούργιο στοιχείο από το έλεγχο με την heapq.heappush και τυπώνω τα αποτελέσματα.

Πειράματα:

→ Πειράματα για k=1,2,3,4,5,10

```
chareskazakides@192 Diax_Dedom_3 % python3 Askisi3_Kazakidis_Theocharis_4679.py 4

Number of sequential accesses= 3018
Top k objects:
50905: 14.84
85861: 14.76
22652: 14.74
75232: 14.74

Brute - Force
Number of sequential accesses= 300000
Top k objects:
50905: 14.84
85861: 14.76
75232: 14.74
22652: 14.74
chareskazakides@192 Diax_Dedom_3 % python3 Askisi3_Kazakidis_Theocharis_4679.py 3

Number of sequential accesses= 3018
Top k objects:
50905: 14.84
85861: 14.76
22652: 14.74

Brute - Force
Number of sequential accesses= 300000
Top k objects:
50905: 14.84
85861: 14.76
75232: 14.74
```

```
chareskazakides@192 Diax_Dedom_3 % python3 Askisi3_Kazakidis_Theocharis_4679.py 2

Number of sequential accesses= 2666
Top k objects:
50905: 14.84
85861: 14.76

Brute - Force
Number of sequential accesses= 300000
Top k objects:
50905: 14.84
85861: 14.76
chareskazakides@192 Diax_Dedom_3 % python3 Askisi3_Kazakidis_Theocharis_4679.py 1

Number of sequential accesses= 1380
Top k objects:
50905: 14.84

Brute - Force
Number of sequential accesses= 300000
Top k objects:
50905: 14.84
chareskazakides@192 Diax_Dedom_3 % python3 Askisi3_Kazakidis_Theocharis_4679.py 10

Number of sequential accesses= 5848
Top k objects:
50905: 14.84
85861: 14.76
22652: 14.74
75232: 14.74
20132: 14.74
21824: 14.70
9041: 14.66
97866: 14.65
83759: 14.64
96407: 14.58

Brute - Force
Number of sequential accesses= 300000
Top k objects:
50905: 14.84
85861: 14.76
75232: 14.74
22652: 14.74
20132: 14.74
21824: 14.70
9041: 14.66
97866: 14.65
83759: 14.64
96407: 14.58
```

Παρατηρήσεις:

→Παρατηρώ ότι μέσω της χρήσης του αλγορίθμου εμφανίζονται αισθητά καλύτερα αποτελέσματα σε σχέση με την χρήση του bruteforce καθώς με τον αλγόριθμο έχουμε λιγότερο πλήθος διαδοχικών accesses έναντι με το bruteforce όποια και να είναι η τιμή του k που δίνω ως όρισμα στο command line. Πρόσθεσα σχόλια για το τι κάνει κάθε συνάρτηση τόσο στο κώδικα όσο και το report. Άμα υπάρχει κάποιο πρόβλημα στο compile ή οτιδήποτε άλλο επικοινωνήστε μαζί μου παρακαλώ