Διαχείριση Σύνθετων Δεδομένων
Εργασία 3

Όθωνας Γκαβαρδίνας
ΑΜ: 2620

Μέρος 1

Πρόβλημα: Top-k ερωτήματα με χρήση του αλγορίθμου NRA
Αρχεία εισόδου: 2017_PTS.csv, 2017_BLK.csv, 2017_AST.csv, 2017_TRB.csv, 2017_STL.csv, 2017_ALL.csv
Γλώσσα: Python 3.6.7

Για την επίλυση του προβλήματος διαχώρισα το πρόβλημα στα παρακάτω βήματα, που αποτελούν ξεχωριστές συναρτήσεις:

(1) get_input():

Η συνάρτηση αυτή λαμβάνει τα ορίσματα του χρήστη και με βάση αυτά, ορίζει ένα πίνακα με αριθμούς που χρησιμοποιούνται ως δείκτες στην εύρεση των σωστών αρχείων για άνοιγμα, και επίσης ορίζει τον αριθμό των καλύτερων αποτελεσμάτων k.

(2) open_files():

Λαμβάνει τον πίνακα της προηγούμενης συνάρτησης ως όρισμα, και μία καθολική μεταβλητή, που είναι μία προκαθορισμένη λίστα με αρχικά που αντιστοιχούν σε κάθε αρχείο εισόδου. (Η σειρά καθορίζεται με βάση, έναν πίνακα που έχει ολόκληρη την πληροφορία, στο αρχείο 2017_ALL.csv.) Η συνάρτηση ανοίγει τα αρχεία που χρειάζονται και επιστρέφει λίστα με τους file descriptors τους.

(3) find_top_k():

Στη συνάρτηση αυτή, υλοποιείται ο αλγόριθμος NRA.
-Αρχικά, υπολογίζονται κάποιες μέγιστες τιμές που χρειάζονται στη συνάρτηση f (που κάνει κανονικοποίηση) του προβλήματος.

Γίνεται το growing phase, όπου:
-Στη συνέχεια, μέχρι να βρώ k αποτελέσματα, διαβάζω μία γραμμή από κάθε αρχείο, και εισάγω τον παίκτη που βρίσκεται σ' αυτή, σε μία δομή. Ακόμη, χρησιμοποιώ τη συνάρτηση f για κανονικοποίηση κάθε τιμής που βρίσκω και τοποθετώ το αποτέλεσμα στο lower bound του συγκεκριμένου παίκτη.
-Για t = μέγιστη ως τώρα τιμή με βάση τα lower bounds.
T = άθροισμα κανονικοποιημένων αποτελεσμάτων στις γραμμές που μόλις διαβάστηκαν (δηλαδή, αναμενόμενος αριθμός για τισ ακριβώς επόμενες γραμμές που θα διαβαστούν)
-Ελέγχεται η σχέση t>=T, και αν ναί, το πρόγραμμα συνεχίζει στο shrinking phase.

Στο shrinking phase:
-Για το upper bound, κρατόνται, από τις γραμμές που μόλις διαβάστηκαν, τα αποτελέσματα των συναρτήσεων f. Αυτά, μέσω της συνάρτησης max_upper_bound(), συνθέτουν όλα τα πάνω όρια, και επιστρέφεται το μέγιστο άνω όριο.
-Ελέγχεται εάν σε αυτήν την επανάληψη έχει βρεθεί κάποιο κάτω όριο >= του μεγίστου άνω ορίου.
-Αν ναι, αποθηκεύεται το αποτέλεσμα σε λίστα που θα επιστραφεί και αφαιρείται ο top παίκτης από τις δομές, που είναι αποθηκευμένοι οι ως τώρα υποψήφιοι παίκτες.
Στην διάρκεια των επαναλήψεων κρατάται και ο αριθμός των γραμμών που έχουν διαβαστεί σε κάθε ένα από τα αρχεία. (είναι για όλα ο ίδιος)

(Σημείωση: Η μεταβλητή my_k, έχει το ρόλο του αριθμού k, και χρησιμεύει στην περίπτωση που ζητούνται περισσότερα αποτελέσματα, από αυτά που υπάρχουν)

(4) get_max_values():

Βοηθητική στην (3). Δημιουργεί ένα λεξικό, στο οποίο αποθηκεύει με κλειδί τον file descriptor και τιμή, τη μέγιστη τιμή του κάθε αρχείου. Η τιμή αυτή βρίσκεται στην πρώτη γραμμή των αρχείων, αφού τα αρχεία είναι ταξινομημένα ως προς αυτή, σε φθίνουσα σειρά.

(5) add_player():

Βοηθητική στην (3). Δημιούργησα συνολικά δύο δομές που αφορούν τους παίκτες. Η μία είναι η κλάση Player, η οποία στον Constructor της λαμβάνει ως όρισμα τη λίστα με τα ανοικτά αρχεία και το id ενός παίκτη. Η δεύτερη δομή είναι μία ουρά πρωτεραιώτητας, ως σωρός μεγίστου που διαθέτει μία λίστα με τις τιμές του σωρού, και δύο λεξικά που περιλαμβάνουν τους παίκτες.
Στη συνάρτηση αυτή, προσθέτω νέο παίκτη στην ουρά πρωτεραιώτητας με τα δοθέντα ορίσματα.

(6) func():

Βοηθητική στην (3). Κάνει την ομαλοποίηση μιας βαθμολογίας επίδωσης και επιστρέφει το αποτέλεσμα.

(7) set_value():

Βοηθητική στην (3). Προσθέτει το ομαλοποιημένο αποτέλεσμα στο lower lower bound του παίκτη που βρίσκεται σε μια γραμμή ενός αρχείου. Επίσης, όπως αναφέρθηκε πιο πάνω, ο κάθε παίκτης έχει τη λίστα με τα ανοικτά αρχεία. Εδώ, όταν προστεθεί η τιμή από ένα από τα αρχεία, αυτό διαγράφεται από τη λίστα του σιγκεκριμένου παίκτη.
Η διαδικασία αυτή γίνεται στην ουρά πρωτεραιώτητας με την κατάλληλη μέθοδο.

(8) calculate_t():

Επιστρέφει το head της ουράς πρωτεραιώτητας, που είναι το μέγιστο στοιχείο.

(9) calculate_T():

Επιστρέφει άθροισμα των ομαλοποιημένων στοιχείων των γραμμών που διαβάστηκαν.

(10) max_upper_bound():

Βοηθητική στην (3). Στη συνάρτηση αυτή, δίνεται η λίστα με όλες τις τιμές που έχουν βρεθεί και κανονικοποιηθεί σε μία συγκεκριμένη γραμμή, των αρχείων εισόδου. (στο ίδιο σημείο για κάθε αρχείο)
Για κάθε παίκτη που βρίσκεται στην ουρά πρωτεραιώτητας των παικτών, υπολογίζει το upper bound του, με βάση τις τιμές της λίστας εισόδου, και τα lower bounds.
Κάθε φορά που βρίσκει ένα upper bound, το κρατά σε προσωρινή τιμή, για να μπορέσει να βρεί το μέγιστο upper bound. Το βρίσκει, και το επιστρέφει.

(11) check_for_top():

Βοηθητική στην (3). Ελέγχει αν το μέγιστο lower_bound είναι >= του upper_bound.

(12) close_files():

Έχοντας τη λίστα με τα αρχεία που ανοίχθηκαν στην αρχή, τα κλείνει.

(13) print_results():

Τυπώνει στην οθόνη, τους καλύτερους k παίκτες κατα φθίνουσα σειρά, και τον αριθμό των γραμμών που διαβάστηκαν από τα αρχεία.

Ουρά Πρωτεραιώτητας

Γίνεται χρήση της βιβλιοθήκης heapq, με την οποία είναι δυνατός ο χειρισμός σωρού ελαχίστου. Στην άσκηση για την προσαρμογή αυτής της βιβλιοθήκης σε σωρό μεγίστου, τα δεδομένα αποθηκεύονται με αρνητική τιμή.

Η ουρά αποτελείται από μία λίστα, που έχει τα κλειδιά της ουράς. Ακόμη, διαθέτει δύο λεξικά, από τα οποία το πρώτο (dict) έχει ως κλειδί τιμή του σωρού σε θετική εκδοχή και ως τιμή αντικείμενο τύπου Player. Το δεύτερο, έχει ως κλειδιά τα ids των παικτών και ως τιμές τη θετική εκδοχή της τιμής του σωρού, και είναι βοηθητικό στη διαχείρηση των κάτω ορίων, όπου χρειάζονται συχνές αλλαγές των κλειδιών.

(13) PriorityQueue.insert():

Στο λεξικό dict, αποθηκεύω τους Players ως λίστες, επειδή είναι δυνατή η εύρεση δύο ίδιων τιμών. Στη μέθοδο αυτή αποθηκεύεται ένας παίκτης στις τρείς δομές.

(13) PriorityQueue.popMax():

Στη μέθοδο αυτή εξάγεται ένας παίκτης, μαζί με την τιμή του, και διαγράφονται από τις δομές.

(13) PriorityQueue.change.lb():

Στη μέθοδο αυτή γίνεται η αύξηση κάποιου lower bound, που δίνεται ως όρισμα. Αρχικά εξάγεται η παλιά τιμή και εισάγεται η νέα στην ουρά. Στη συνέχεια, η αλλαγή μεταφέρεται και στις υπόλοιπες δύο δομές, όπου σβήνεται και το αρχείο που περιείχε το στατιστικό από τη λίστα που κρατά τα αρχεία που δεν έχουν ελεγχθεί ως τώρα.

(.) dummy_main():

Συνάρτηση που χρησιμοποιήθηκε για έλεγχο, μπορεί να αγνοηθεί. Συνάρτηση που αναζητεί τους καλύτερους k παίκτες από το αρχείο 2017_ALL.csv, χωρίς τη χρήση του αλγορίθμου. Για τη χρήση της, χρειάζεται να γίνει import η βιβλιοθήκη operator.

Μέρος 2
Πρόβλημα: Ερωτήσεις κορυφογραμμής με χρήση του αλγορίθμου BNL
Αρχείο εισόδου: 2017_ALL.csv
Γλώσσα: Python 3.6.7

Η υλοποίηση του αλγορίθμου γίνεται στη συνάρτηση main(), όπου:

Λαμβάνεται ένας πίνακας με τα στατιστικά που μας ενδιαφέρουν στη βοηθητική συνάρτηση **get_input()**.

Γίνεται χρήση μιας δομής λεξικού για τους dominant παίκτες.
Ανοίγει το αρχείο εισόδου 2017_ALL.csv
Διαβάζεται η πρώτη γραμμή του αρχείου εισόδου, που περιέχει τα labels.

Στη δομή αρχικά τοποθετείται ο πρώτος παίκτης στη συνάρτηση **get_first_player()**.
Για κάθε γραμμή του αρχείου εισόδου:
Λαμβάνω έναν νέο παίκτη, ο οποίος βρίσκεται στη γραμμή.
Για κάθε παίκτη στο λεξικό των dominant:
Χρησιμοποιώ ένα μετρητή, τον **domination**.
Χρησιμοποιώ επίσης ένα μετριτή για ισότητα, τον **same**.
Για κάθε χαρακτηριστικό του παίκτη:
-Αν το χαρακτηριστικό είναι καλύτερο από αυτό ενός από τους dominant (μεγαλύτερη τιμή):
Αυξάνω κατά 1 το μετρητή.
-Αν χειρότερο (μικρότερη τιμή):
Μειώνω κατα 1 το μετρητή.
-Αλλιώς, είναι ίσες και αυξάνω το δεύτερο μετρητή.

Στη συνέχεια με βάση τον άθροισμά (και αφαίρεση) **domination + same**:
-Αν = πλήθος στατιστικών, που σημαίνει ότι ο νέος παίκτης κερδίζει κάποιον dominant:
Ο νέος παίκτης θα γίνει dominant, και
Ο παικτης που έχασε, και κάθε άλλος παίκτης που χάνει αποθηκεύεται σε λίστα, έτσι ώστε στη συνέχεια, να διαγραφούν από το λεξικό, γιατί πλέον είναι dominated.
-Αν < πλήθος στατιστικών, που σημαίνει πως δεν έχασε σε κάθε στατιστικό:
Σίγουρα δεν είναι dominant, καθώς έχει χάσει.
(Δε χρειάζεται να εξεταστούν περεταίρω περιπτώσεις γι' αυτόν)

-Αλλιώς:
Τότε γνωρίζουμε ότι μέχρι στιγμής, μπορεί να μπεί στους dominant.
(Θα γνωρίζουμε σίγουρα γι' αυτόν όταν εξεταστούν όλοι οι dominant)

Εισάγεται ο παίκτης στο λεξικό, εαν έχει αποφασιστεί.
Διαγράφονται ο παίκτες, τους οποίους έχει νικήσει ο νέος παίκτης.

Κλείνει το αρχείο εισόδου
Τυπόνονται τα αποτελέσματα, στη συνάρτηση **print_results()**.