Γεώργιος Δημόπουλος

Α.Μ. 2964

1η σειρά ασκήσεων στο μάθημα: Διαχείριση Σύνθετων Δεδομένων (κ.Μαμουλής)

Αρχικά θα πρέπει να κάνετε πρώτα compile το αρχείο sorted_files.py, το οποίο κάνει ταξινόμηση τα αρχεία τα οποία χρειαζόμαστε στα ερωτήματα. Αυτό γιατί σε όλα μου τα προγράμματα χρησιμοποιώ τα ταξινομημένα αρχεία που τα έχω ονομάσει εγώ στο πρόγραμμα sorted_files.py.

Ταξινόμηση: Έχω δημιουργήσει μια συνάρτηση *takeSecond(elem)* η οποία απλά επιστρέφει σε float τα ψηφία του tconst ώστε να μπορεί να ταξινομηθούν μετά εύκολα. Αυτό που κάνω στην ταξινόμηση είναι να διαβάζω κάθε φορά το αρχείο που θέλω και να το βάζω μέσα σε μια λίστα. Μετά γράφω την πρώτη γραμμή του αρχείου στο νέο ταξινομημένο αρχείο και έπειτα διαγράφω την πρώτη γραμμή από την λίστα ώστε να μπορέσω να την ταξινομήσω. Την ταξινόμηση την κάνω χρησιμοποιώντας την συνάρτηση *sort* της python και σαν κλειδί (με ποιον τρόπο θέλω να ταξινομηθεί η λίστα) βάζω την συνάρτηση που έχω δημιουργήσει, δηλαδή την *takeSecond(elem).*

Ερώτημα 1.1: Για αυτό το ερώτημα δημιουργώ την συνάρτηση *yield_function(file)*, η οποία επιστρέφει με **yield** την επόμενη γραμμή του αρχείου χωρίς το "\n". Για να δω που τελειώνει το αρχείο *title_crew* το βάζω μέσα σε μια λίστα και έπειτα παίρνω το τελευταίο της στοιχείο. Μετά διαγράφω την λίστα για να απελευθερώσω την μνήμη. Τώρα του λέω μέχρι να τελειώσει το αρχείο *title_crew* να παίρνεις την επόμενη γραμμή του αρχείου. Αν τώρα έχουμε πάνω από 2 σκηνοθέτες τότε μπες στην συνάρτηση *merge_by_directors(file,id,directors)* και σαν id πάρε το tconst και σαν directors τους directors. Όταν μπεις στην συνάρτηση προχώρα την επόμενη γραμμή του *title_basics*. Όσο το tconst του *basics* διαφέρει από το id τότε του λέω απλά να προχωρήσει στην επόμενη γραμμή. Όταν το tconst είναι ίδιο με το id τότε του λέω να γράψει στο αρχείο αυτά που μας ζητούνται(primaryTitle,directors).

Ερώτημα 1.2: Για αυτό το ερώτημα δημιουργώ την συνάρτηση *yield_function(file)*, η οποία επιστρέφει με **yield** την επόμενη γραμμή του αρχείου χωρίς το "\n". Για να δω που τελειώνει το αρχείο *title_episode* το βάζω μέσα σε μια λίστα και έπειτα παίρνω το τελευταίο της στοιχείο. Μετά διαγράφω την λίστα για να απελευθερώσω την μνήμη. Τώρα του λέω μέχρι να τελειώσει το αρχείο *title_episode* να παίρνεις την επόμενη γραμμή του αρχείου. Αν τώρα το episosdeNumber=1 μπες στην συνάρτηση *merge_by_episode(file,id,parentTconst,seasonNumber)* και σαν id πάρε το tconst, σαν parentTconst πάρε το parentTconst, σαν seasonNumber πάρε το

seasonNumber. Τώρα του λέω να προχωρήσει το αρχείο *title_basics μία γραμμή*. Αν το tconst του *basics* είναι διαφορετικό από το id τότε απλά προχώρα μια γραμμή το *basics*. Διαφορετικά γράψε στο αρχείο αυτά που μας ζητούνται(primaryTitle,parentTconst,seasonNumber).

Ερώτημα 1.3: Για αυτό το ερώτημα δημιουργώ την συνάρτηση *yield_function(file)*, η οποία επιστρέφει με **yield** την επόμενη γραμμή του αρχείου χωρίς το "\n". Για να δω που τελειώνει το αρχείο *title_basics* το βάζω μέσα σε μια λίστα και έπειτα παίρνω το τελευταίο της στοιχείο. Μετά διαγράφω την λίστα για να απελευθερώσω την μνήμη. Τώρα του λέω μέχρι να τελειώσει το αρχείο *title_basics* να παίρνεις την επόμενη γραμμή του αρχείου. Τώρα όσο τα δύο tconst ταυτίζονται απλά προχωράω και τα δύο αρχεία από μια γραμμή. Αν όμως τα δύο tconst είναι διαφορετικά αυτό σημαίνει ότι ένα αρχείο από το *basics* δεν υπάρχει στο *rating* δηλαδή αυτή η ταινία δεν έχει ratings. Τότε γράφω στο αρχείο αυτά που μας ζητούνται(primaryTitle) και προχωράω **MONO** το αρχείο *basics* μια γραμμή.

Ερώτημα 2.1_sorting: Για αυτό το ερώτημα το μόνο που κάνω είναι να ανοίγω το αρχείο *title_ratings* και απλά να περνάω σε μια λίστα το averageRating αφού πρώτα το κάνω float. Μετά ταξινομώ την λίστα και απλά μετράω με τους counter μου αυτά που μας ζητούνται.

Ερώτημα 2.1_hashing: Για αυτό το ερώτημα το μόνο που κάνω είναι να ανοίγω το αρχείο *title_ratings* και απλά να περνάω σε μια λίστα το averageRating αφού πρώτα το κάνω float. Επίσης δημιουργώ μια λίστα με 10 στοιχεία που είναι οι μετρήσεις που μας ενδιαφέρουν. Μετά διατρέχω την λίστα με το averageRating και αυξάνω το κατάλληλο στοιχείο της λίστας που έχω δημιουργήσει. *Παρατηρώ ότι το hashing είναι πιο γρήγορο από το sorting.*

Ερώτημα 2.2: Για αυτό το ερώτημα δημιουργώ την συνάρτηση *yield_function(file)*, η οποία επιστρέφει με **yield** την επόμενη γραμμή του αρχείου χωρίς το "\n". Για να δω που τελειώνει το αρχείο *title_basics* το βάζω μέσα σε μια λίστα και έπειτα παίρνω το τελευταίο της στοιχείο. Μετά διαγράφω την λίστα για να απελευθερώσω την μνήμη. Τώρα του λέω μέχρι να τελειώσει το αρχείο *title_basics* να παίρνεις την επόμενη γραμμή του αρχείου. Αν τώρα τα δύο tconst ταυτίζονται μπες μέσα στην συνάρτηση *merge_by_starYear(year,rating)* και πάρε σαν year το startYear και σαν rating το averageRating. Αν τώρα δεν έχει startYear απλά το παραλείπουμε. Διαφορετικά βάζουμε το year σε ένα λεξικό. Που σαν κλειδί έχει την χρονολογία και σαν τιμή είναι μια λίστα η οποία έχει το rating και πόσες φορές μπήκαμε σε αυτή την ημερομηνία. Αν τώρα έχει ξαναεμφανιστεί αυτή η ημερομηνία το μόνο που κάνω είναι τα μπω στο λεξικό να προσθέσω στο rating το καινούριο μου rating και

να αυξήσω το δεύτερο στοιχείο της λίστας κατά 1. Τέλος τυπώνω το λεξικό βρίσκοντας και τον μέσο όρο των χρονολογιών διαιρώντας το πρώτο στοιχείο της λίστας με το δεύτερο.

ΣΧΕΣΙΑΚΗ ΑΛΓΕΒΡΑ

Ερώτημα 1.1:

$$R_1 \leftarrow title.basics \bowtie title.crew$$

$$R_2 \leftarrow \sigma_{length(directors)>10}(R_1)$$

$$R_3 \leftarrow \Pi_{(primaryTitle,directors)}(R_2)$$

Ερώτημα 1.2:

$$R_1 \leftarrow title.basics \bowtie title.episode$$

$$R_2 \leftarrow \sigma_{episodeNumber=1}(R_1)$$

$$R_3 \leftarrow \Pi_{(primaryTitle,parentTconst,seasonNumber)}(R_2)$$

Ερώτημα 1.3:

$$R_1 \leftarrow \Pi_{(tconst,primaryTitle)}(title.basics)$$

$$R_2 \leftarrow R_1 \bowtie title.ratings$$

$$R_3 \leftarrow \Pi_{(tconst,primaryTitle)}(R_2)$$

$$R_4 \leftarrow R_1 - R_3$$

$$R_5 \leftarrow \Pi_{primaryTitle}(R_4)$$

Ερώτημα 2.1:

$$R_1(average) \leftarrow \Pi_{averageRating}(title.ratings)$$

$$R_2 \leftarrow \Pi_{averageRating}(title.ratings)$$

$$R_3 \leftarrow R_1 \bowtie_{average=averageRating} R_2$$

$$R_4 \leftarrow average\ \mathbf{G}\ count_{(averageRating)}(R_3)$$

$$R_5 \leftarrow \sigma_{average<1.1}(R_4)$$

$R_{a1} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_5)$

$R_6 \leftarrow \sigma_{average>1 \text{ and } average<2.1}(R_4)$

$R_{a2} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_6)$

$R_7 \leftarrow \sigma_{average>2 \text{ and } average<3.1}(R_4)$

$R_{a3} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_7)$

$R_8 \leftarrow \sigma_{average>3 \text{ and } average<4.1}(R_4)$

$R_{a4} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_8)$

$R_9 \leftarrow \sigma_{average>4 \text{ and } average<5.1}(R_4)$

$R_{a5} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_9)$

$R_{10} \leftarrow \sigma_{average>5 \text{ and } average<6.1}(R_4)$

$R_{a6} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_{10})$

$R_{11} \leftarrow \sigma_{average>6 \text{ and } average<7.1}(R_4)$

$R_{a7} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_{11})$

$R_{12} \leftarrow \sigma_{average>7 \text{ and } average<8.1}(R_4)$

$R_{a8} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_{12})$

$R_{13} \leftarrow \sigma_{average>8 \text{ and } average<9.1}(R_4)$

$R_{a9} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_{13})$

$R_{14} \leftarrow \sigma_{average>9}(R_4)$

$R_{a10} \leftarrow \mathbf{G}\ \mathrm{sum}_{(count\_averageRating)}(R_{14})$

$R_{teliko} \leftarrow R_{a1} \cup R_{a2} \cup R_{a3} \cup R_{a4} \cup R_{a5} \cup R_{a6} \cup R_{a7} \cup R_{a8} \cup R_{a9} \cup R_{a10}$

Ερώτημα 2.2:

$R_1 \leftarrow title.basics \bowtie title.ratings$

$R_a \leftarrow startYear\ \mathbf{G}\ \mathrm{avg}_{(averageRating)}(R_1)$