Διαχείριση Σύνθετων Δεδομένων
3η Σειρά Ασκήσεων

Εαρινό 2020

Πασόη Σοφία ΑΜ:2798
Υπεύθυνος Καθηγητής: κ. Μαμουλής Νικόλαος

Μέρος 1: Αλγόριθμος A top-k join

Αρχικά, ανοίγει τα 2 αρχεία και ορίζει μια βοηθητική μεταβλητή turn, τους μετρητές countMales(μετράει τις έγκυρες γραμμές του αρχείου males_sorted) και countFemales(μετράει τις έγκυρες γραμμές του αρχείου females_sorted), τις μεταβλητές p1_cur, p1_max, p2_cur, p2_max και τα 2 λεξικά, ένα για κάθε κατηγορία.

Επαναληπτικα, διαβάζει γραμμή-γραμμή το αρχείο με τα males και κάνει split στο κόμμα. Ελέγχει αν η γραμμή είναι έγκυρη, δηλαδή το male είναι άνω των 18 και όχι παντρεμένος, Κρατάει τις τιμές των πεδίων age, instanceWeight και code σε 3 μεταβλητές αντίστοιχα. Στη συνέχεια, ακολουθώντας των αλγόριθμο HRJN βάζω την τιμή του πεδίου instanceWeight στις μεταβλητές p1_cur και p1_max και βάζω στο λεξικό το male μου με key την τιμή του age και values τις τιμές του code και του instanceWeight. Επαναληπτικά, διαβάζει το αρχείο με τα females και εκτελεί ακριβώς την ίδια διαδικασία με το males. Μετά το διάβασμα κάθε γραμμής γίνεται break γιατί θέλουμε τα αρχεία να διαβάζονται ταυτόχρονα(διαβάζει 1 γραμμή από το ένα, γινεται break, διαβάζει 1 γραμμή από το άλλο, γίνεται break και ξανά το ίδιο).

Έπειτα, ορίζεται η συνάρτηση T ως το άθροισμα των p1_max και p2_max και ο σωρός.

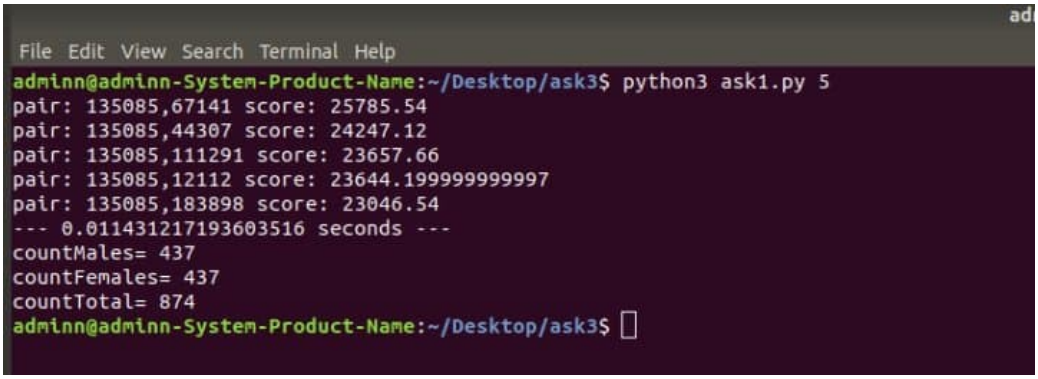
Επαναληπτικά, ελέγχει ποιο αρχείο διαβάζει. Όταν διαβάζεται το males η τιμή της μεταβλητής turn έχει άρτια τιμή(αντίστοιχα όταν έχει περιττή τιμή διαβαζει το females). Έτσι, ελέγχει αν το υπόλοιπο της ακέραιας διαίρεσης με το 2 είναι 0. Αν είναι, αυξάνει το turn κατά 1 και ξεκινά το διάβασμα του αρχείου males. Επειδή δεν έχει καμιά πληροφορία για το μέγεθος του αρχείου, πραγματοποιεί ακόμα έναν έλεγχο για το αν η γραμμή που διαβάζει είναι κενή. Αν είναι σημαίνει πως το αρχείο τελείωσε και κάνει break, ώστε να διαβάσει το άλλο αρχείο για να βρει όλα τα πιθανά ζεύγη(αν κάνει return ή yield τότε χάνονται κάποια από τα ζευγάρια). Αν δεν είναι κενή, τότε ελέγχει αν η γραμμή είναι έγκυρη(άνω των 18 και όχι παντρεμένος). Αν περασει τον έλεχο, αυξάνεται ο μετρητής countMales και οι τιμές των πεδίων age, instanceWeight και code μπαίνουν στις αντίστοιχες μεταβλητές, το p1_cur γίνεται ίσο με instanceWeight, παίρνει τη λίστα που αντιστοιχεί στο age που έχει διαβάσει και σε αυτήν προσθέτει το νέο και το επανατοποθετεί στο λεξικό(κάνει get ώστε να επιστρεψει το [] εάν δεν υπαρχει το age αυτό στο λεξικό) και στον πίνακα values βάζει τα values με key την τιμή του age από το λεξικό των females. Αν το υπόλοιπο της ακέραιας διαίρεσης του turn με το 2 δεν είναι 0, τότε το turn έχει περιττή τιμή, άρα διαβάζει το αρχεί females. Εκτελείται η ίδια διαδικασία με το males με τη μόνη διαφορά ότι αυξάνεται ο counterFemales, ενημερώνεται με τον ίδιο τρόπο(με το λεξικό των males) το λεξικό των females, και τα values που μπαίνουν στον πίνακα values αφορούν τα males. Μετά το διάβασμα κάθε γραμμής των αρχείων γίνεται break για να εξασφαλίσουμε ότι τα αρχεία διαβάζονται ταυτοχρόνως.

Ορίζεται, έπειτα, η συνάρτηση T ως το μέγιστο ανάμεσα στα αθροίσματα p1_max+p2_cur και p2_max+p1_cur. Διατρέχει τον πίνακα values και υπολογίζει το άθροισμα των instanceWeight(sumInstanceWeight). Επειδή ο σωρός που ορίστηκε, είναι σωρός ελαχίστων, για να μετατραπεί σε σωρό μεγίστων, οι τιμές

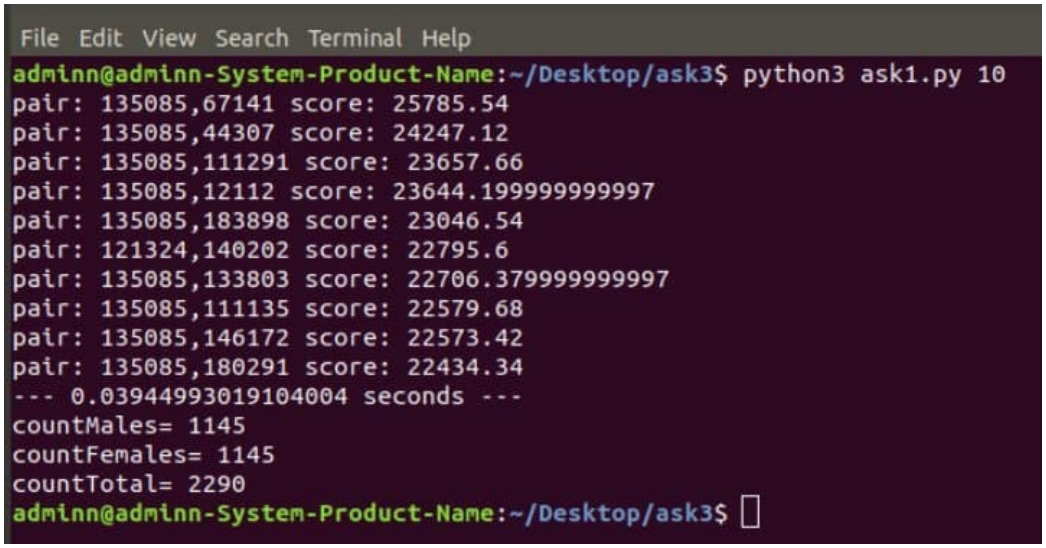
πολλαπλασιαζονται με -1. Έτσι η μεγαλύτερη τιμή θα βρίσκεται στην αρχή. Στη συνέχεια, βάζει σε μια μεταβλήτή pair τις τιμές των code, του αντίστοιχου male και female, και σε μια δεύτερη μεταβλητή join το άθροισμα sumInstanceWeight και το pair, και τέλος βάζει στο σωρό το ζευγάρι(join). Όσο ο σωρός έχει στοιχεία, βγαίνει από τον σωρό τον παρόν στοιχείο και μπαίνει σε 1 μεταβλητή value. Αν το value(το οποίο ξανά πολλαπλασιάζουμε με -1 γι αν απάρουμε τη σωστή τιμή του) έχει μεγαλύτερη ή ίση τιμή από την συνάρτηση T, γίνεται yield το value και οι μετρητές counterMales και counterFemales. Αν δεν είναι, το value μπαίνει στον σωρό γίνεται break και yield.

Τέλος, για να τρέξει το πρόγραμμα, παίρνω το K από τη γραμμή εντολών, καλέιται η part1 και οι τιμές τις μπαίνουν σε ένα πίνακα g. Έπειτα, καλείται η time για να πάρουμε το χρόνο εκκίνησης. Στη συνέχεια, επαναληπτικά, για κάθε τιμή του K, παίρνει τις αντίστοιχες τιμές που επιστρέφονται από την part1. Τέλος, υπολογίζει τον χρόνο εκτέλεσης καλώντας ξανά την time για να πάρει το χρόνο λήξης και αφαιρεί από αυτόν τον χρόνο έναρξης, τυπώνει τον χρόνο που βρήκε και τους countMales, countFemales και έναν μετρητή που είναι το άθροισμα των άλλων 2.

Δείγμα εξόδου για K=5:



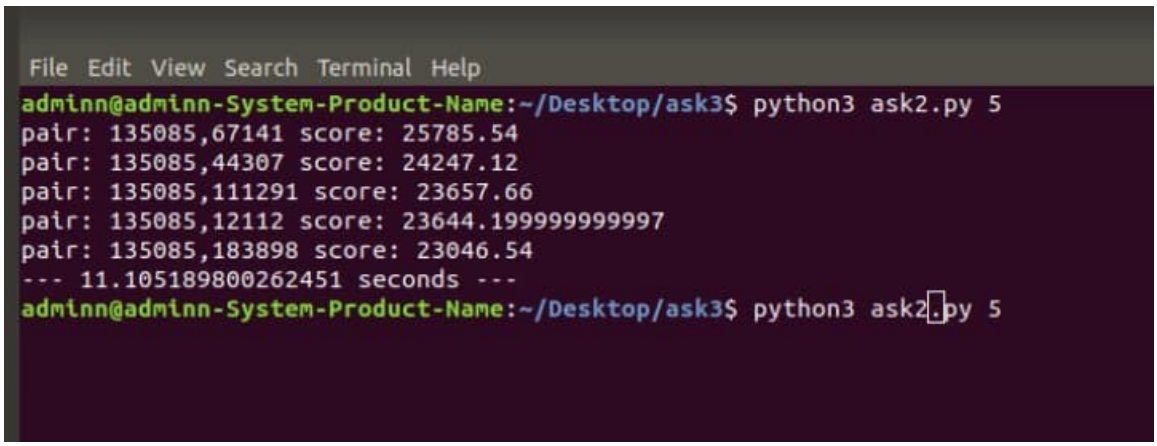
Δείγμα εξόδου για K=10:



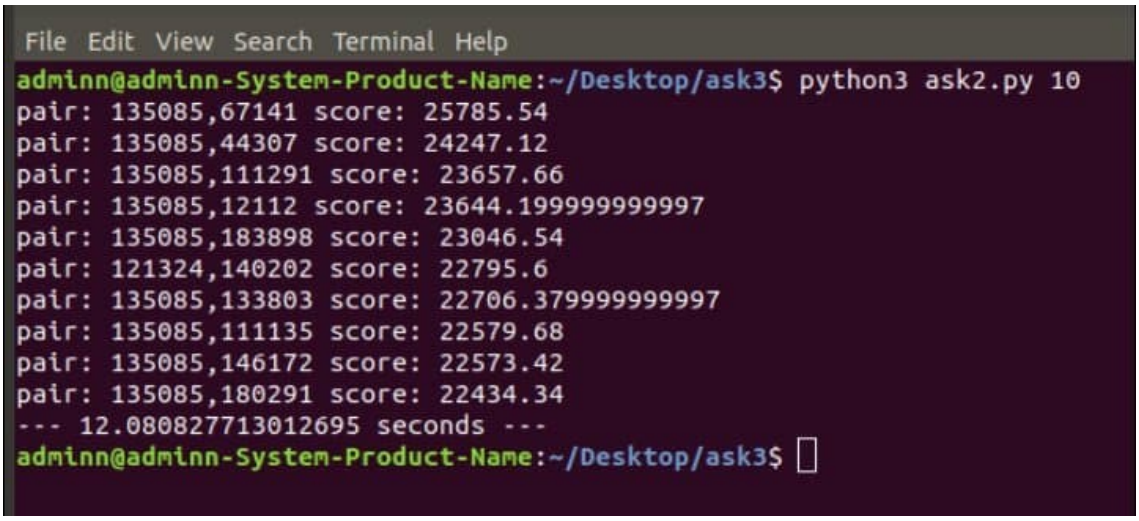
Μέρος 2: Αλγόριθμος B top-k join

Αρχικά, καλείται η time για να πάρει τον χρόνο έναρξης, και η τιμή της κρατείται σε μια μεταβλητή start_time. Ανοίγει το αρχείο males_sorted και το διαβάζει ολόκληρο. Έπειτα, ορίζει το λεξικό. Επαναληπτικά, διαβάζει γραμμή-γραμμή το αρχείο με τα males και κάνει split στο κόμμα. Ελέγχει αν η γραμμή είναι έγκυρη, δηλαδή το male είναι άνω των 18 και όχι παντρεμένος, Κρατάει τις τιμές των πεδίων age, instanceWeight και code σε 3 μεταβλητές αντίστοιχα και βάζει στο λεξικό το male με key την τιμή του age και values τις τιμές του code και του instanceWeight. Κλείνει το αρχείο, ορίζει το σωρό ελαχίστων και ξεκινά το διάβασμα του αρχείου females_sorted. Διαβάζει ολόκληρο το αρχείο και επαναληπτικά, διαβάζει γραμμή-γραμμή το αρχείο με τα females και κάνει split στο κόμμα. Ελέγχει αν η γραμμή είναι έγκυρη, δηλαδή το female είναι άνω των 18 και όχι παντρεμένη, Κρατάει τις τιμές των πεδίων age, instanceWeight και code σε 3 μεταβλητές αντίστοιχα. Βάζει σε μια λίστα makeList τα values από το λεξικό με key την τιμή του πεδίου age. Διατρέχει τη λίστα και για κάθε male υπολογίζεται το άθροισμα του instanceWeight του male και του female και το βάζει στη μεταβλητή sumInstanceWeight, βάζει τα code των male και female σε μια μεταβλητή pair και τέλος βάζει σε μια μεταβλητή join το sumInstanceWeight και pair. Αν το μήκος του σωρού είναι μικρότερο από το K το join μπαίνει στο σωρό. Αλλιώς, βγάζω το πρώτο στοιχείο του σωρού(είναι το ελάχιστο) και το βάζω σε 1 μεταβλητή heapSmallest. Αν το heapSmallest είναι μικρότερο από το sumInstanceWeight το join μπαίνει στο σωρό. Αλλιώς, το heapSmallest ξανά μπαίνει στο σωρό. Έπειτα ορίζει έναν πίνακα results και επαναληπτικά όσο ο σωρός έχει στοιχεία βγάζει από τον σωρό το πρώτο στοιχείο(heapSmallest) και το βάζει στην πρώτη θέση του πίνακα. Τέλος, τυπώνει τον πίνακα results, όπου κάθε στοίχειο του στην πρώτη θέση βρίσκεται η μεταβλητή pair και στη δεύτερη το score που είναι το sumInstanceWeight. Τέλος, καλείται η time για να πάρει το χρόνο λήξης και υπολογίζει το χρόνο που χρειάστηκε για να τρέξει ο αλγόριθμος ως χρόνος λήξης-χρόνος έναρξης και τον τυπώνει. Για να τρέξει το πρόγραμμα παίρνει το K από τη γραμμλή εντολών και με αυτό καλείται η part2().

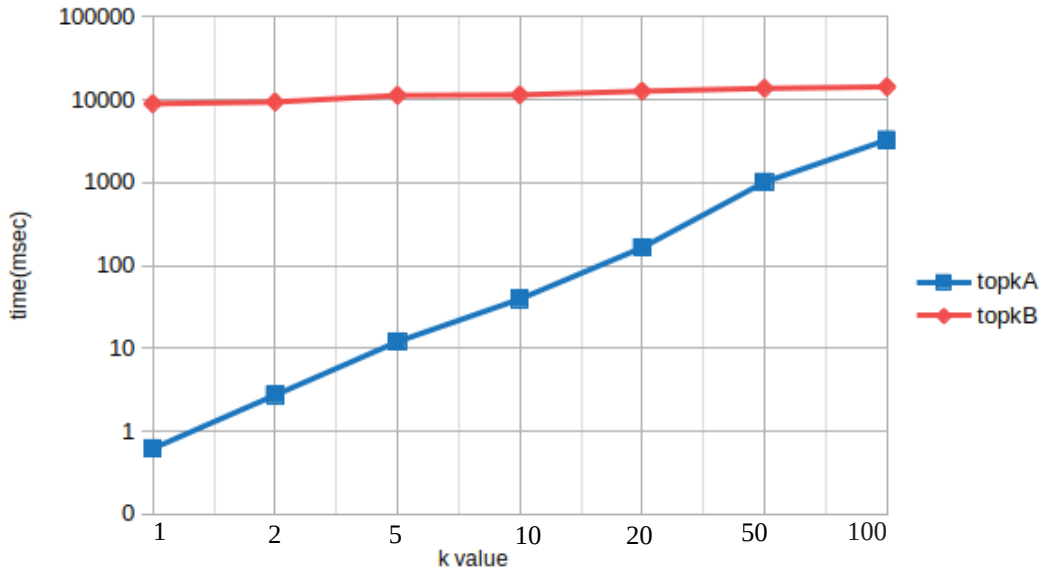
Δείγμα εξόδου για K=5:



Δείγμα εξόδου για K=10:



Μέρος 3: Γραπτό Μέρος



Ο αλγόριθμος Α διαβάζει ταυτοχρόνως τα δύο αρχεία. Δηλαδή, διαβάζει πρώτα τη γραμμή από το males_sorted, έπειτα την γραμμή από το females_sorted και ξανά από την αρχή. Αυτό συνεχίζεται είτε μέχρι να τελειώσουν τα αρχεία είτε βρεθεί ο ζητούμενος αριθμός ζευγαριών. Ο αλγόριθμος Β διαβάζει εξ ολοκλήρου και τα 2 αρχεία, χώρις να τον ενδιαφέρει(όσον αφορά το διάβασμα) η τιμή του Κ. Συμπεραίνουμε λοιπόν, ότι ο Α διαβάζει λιγότερες γραμμές από τον Β.

Από το διάγραμμα παρατηρούμε ότι ο αλγόριθμος Β έχει σχεδόν σταθερό χρόνο εκτέλεσης(αυξάνοντας το Κ βλέπουμε ελάχιστη αύξηση, διότι ο χρόνος εκτέλεσης εξαρταται σε ένα μικρό βαθμό από το Κ, αφού χρησιμοποίειται για πράξεις μέσα στον κώδικα), διότι οποιοδήποτε τιμή και αν λάβει το Κ θα αναγκαστεί να διαβάσει ολόκληρα και τα 2 αρχεία. Αντιθέτως ο Α εκτελεί ανοδική πορεία καθώς το Κ αυξάνεται(μαζί με το Κ αυξάνεται και ο αριθμός των γραμμών που διαβάζει), με χαμηλότερες τιμές από αυτές του Β για τις συγκεκριμένες τιμές του Κ. Αυτό συμβαίνει γιατί κάθε φορά που βρίσκει τα Κ ζεύγη σταματάει και επιστρέφει τα αποτελέσματα.

Βέβαια αν αυξήσουμε αρκετά το Κ, για παράδειγμα πάρει την τιμή 275 τότε παρατηρούμε πως ο χρόνος εκτέλεσης του Α ξεπερνάει τον χρόνο εκτέλεσης του Β.

Δείγμα εξόδου για Κ=275:

Αλγόριθμος Α:

```
pair: 135085,151171 score: 19769.23
--- 16.975013971328735 seconds ---
countMales= 24116
countFemales= 24294
countTotal= 48410
```

Αλγόριθμος Β:

```
pair: 135085,60406 score: 19801.32
pair: 135085,91738 score: 19782.18
pair: 135085,151171 score: 19769.23
--- 16.88128113746643 seconds ---
```

Καταλαβαίνουμε λοιπόν, πως μέχρι ένα αριθμό Κ ο αλγόριθμος Α είναι καλύτερος του Αλγορίθμου Β, γιατί και λιγότερες γραμμές διαβάζει και τρέχει σε μικρότερο χρόνο. Όμως όταν το Κ ξεπεράσει περίπου το 275 ο χρόνος του Α θα αρχίζει και γίνεται μεγαλύτερος του Β. Επιπροσθέτως, καθώς το Κ θα αυξάνεται ο Α θα αναγκάζεται να διαβάζει όλο και περισσότερες γραμμές. Σε ακραία περίπτωση δηλαδή θα αναγκαστεί να διαβάσει ολόκληρα τα 2 αρχεία. Ουσιαστικά σε αυτήν την ακραία περίπτωση οι 2 αλγόριθμοι θα έχουν διαβάσει ολόκληρα τα αρχεία, με τη διαφορά ότι ο Α θα έχει χειροτερο χρόνο εκτέλεσης από τον Β. Συνεπώς, αναλόγως του αριθμού Κ που θέλουμε θα επιλέξουμε και τον αντίστοιχο αλγόριθμο.

Πίνακας αριθμών έγκυρων γραμμών του Αλγορίμου Α:

males_sorted	females_sorted	Συνολικό
44	44	88
134	134	268
437	437	874
1145	1145	2.290
2656	2656	5.312
6030	6030	12.060
10.139	10.139	20.278

Παρατηρήσεις:

1. Για να βρω σε ποια θέση βρίσκονται τα πεδία που χρειάστηκαν, χρησιμοποίησα μόνο τα αρχεία που μας δόθηκαν(τα άνοιξα και μέτρησα θέσεις).
2. Για τον σωρό χρησιμοποιήθηκε έτοιμη βιβλιοθηκη της python(heapq), για να πάρουμε τον χρόνο εκτέλεσης η βιβλιοθήκη time και για να πάρουμε είσοδο το Κ η sys.
3. Το αρχείο ask1.py εκτός από τα ζεύγη και τον χρόνο εκτέλεσης τυπώνει και τον αριθμό των έγκυρων γραμμών των males_sorted και females_sorted, καθώς και το συνολικό τους άθροισμα.
4. Τα αρχεία εκτελούνται με τις εντολές:
   - Άσκηση 1: python3 part1.py <K> όπου Κ κάποιος ακέραιος
   - Άσκηση 2: python3 part2.py <K> όπου Κ κάποιος ακέραιος