Διαχείριση Σύνθετων Δεδομένων
4η Σειρά Ασκήσεων

Εαρινό 2020

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

Μέρος 1: Ανεστραμμένο αρχείο και αναζήτηση με λέξεις-κλειδιά

Συνάρτηση readFile: Παίρνει όρισμα ένα αρχείο και το διαβάζει. Για κάθε γραμμή του αρχείου βρίσκει όλα τα tags. Σε ένα πίνακα value κρατάει κάθε γραμμή-εγγραφή που περιέχει το συγκεκριμένο tag. Έτσι, για κάθε tag μέσα στο λεξικό invertedFileDict, κρατάει το value που επιστρέφει σε μια μεταβλητή value. Σε αυτό το value προστίθεται και η νέα γραμμή στην οποία βρέθηκε το tag. Ενημερώνεται το value του λεξικού με key το tag με το νέο value. Ορίζει ένα σωρό και κρατάει σε ένα πίνακα words όλα τα κλειδιά του λεξικού, τα οποία είναι τα tags. Για κάθε tag παίρνω τον πίνακα με τους αριθμούς των γραμμών στις οποίες βρίσκεται. Σε ένα πίνακα value κρατάω τους αριθμούς γραμμώνπου βρίσκεται το tag και σε ένα πίνακα h βάζει τον value, το tag και τον lines(πίνακας με γραμμές-εγγραφές). Έπειτα βάζει τον h μέσα στον σωρό και ορίζει τον πίνακα invertedFile, που αντιπροσωπεύει το ανεστραμένο αρχείο στη μνήμη και ένα λέξικό keywordPositions. Τέλος φτιάχνει το ανεστραμένο αρχείο. Όσο ο σωρός έχει στοιχεία, τραβάει ένα ένα από αυτόν και τα τοποθετεί σε μια λιστα σε αυξουσα ταξινομηση ως προς το πληθος.
Επιστρέφει το ανεστραμένο αρχείο, τον lines και το keywordPositions.

Συνάρτηση kwSearchRaw: Παίρνει όρισμα δύο λίστες, μία με τις λέξεις που κάνουμε την αναζήτηση(search tags) και μία με τα lines(γραμμές-εγγραφές). Για κάθε γραμμή βρίσκει όλα τα tags και τα κρατάει σε ένα πίνακα. Για καθένα από τα tags ελέγχει αν βρίσκεται στη λίστα με τα search tags. Αν όλα τα tags που ελέγχθηκαν υπάρχουν στη λίστα, τότε σε ένα πίνακα results βάζει την εγγραφή στην οποία βρίσκεται. Τέλος επιστρέφει τον results.

Συνάρτηση myMergeJoin: Παίρνει όρισμα δύο λίστες και ακολουθεί τον αλγόριθμο merge join για να συγχωνευτούν οι λίστες. Αυτό που επιστρέφεται είναι η τομή των λιστών.

Συνάρτηση kwSearchIF: Παίρνει όρισμα μία λίστα με τις λέξεις αναζήτησης(search tags), με το ανεστραμένο αρχείο, τις γραμμές-εγγραφές και το λεξικό keywordPositions. Αρχικά κρατάει στον πίνακα position όλα τα values που επιστρέφονται από το keywordPositions με key το κάθε tag και τον ταξινομεί. Έπειτα βρίσκει το αποτέλεσμα. Το αποτέλεσμα(τις γραμμές-εγγραφές) είναι τα αρχεία που βρίσκονται στις λιστες όλων των keywords, επομένως παίρνουμε την πρώτη λίστα σαν αποτέλεσμα και μετά για κάθε επόμενο keyword βρίσκουμε την τομή τους με την mergejoin. Τέλος, μέσα σε ένα πίνακα results βάζει τις γραμμές-εγγραφές που περιέχουν το keyword.

Για το τύπωμα των ζητούμενων: Καλείται η readFile με όρισμα το αρχείο Restaurants_London_England.tsv και η επιστροφή της μπαίνει σε 3 μεταβλητές, το invertedFile(ανεστραμένο αρχείο), το lines(πίνακας με τις γραμμές-εγγραφές) και το λεξικό keywordsPositions(). Τυπώνει τον αριθμό των keywords, ο οποίος είναι το μήκος του invertedFile και έπειτα της συχνότητα, η οποία βρίσκεται στην πρώτη

θέση κάθε γραμμής του invertedFile. Παίρνει από τη γραμμή εντολών τις λέξεις αναζήτησης και τις βάζει σε ένα πίνακα searchTags. Για να μετρήσουμε το χρόνο εκτέλεσης, πριν από κάθε κλήση των συναρτήσεων καλείται η time για να πάρει το χρόνο έναρξης και αντίστοιχα μετά την κλήση για να πάρει το χρόνο λήξης. Ο χρόνος εκτέλεσης είναι η διαφορά των δύο χρόνων. Η κάθε συνάρτηση καλείται με τα αντίστοιχα ορίσμα και έπειτα τυπώνονται:

1. Ο αριθμός των αποτελεσμάτων της.
2. Το κόστος σε δευτερόλεπτα.
3. Οι εγγραφές της.

Σύγκριση αποδόσεων των συναρτήσεων:

Tags	**kwSearchRaw**	**kwSearchIF**
sushi thai	1 results, cost = 0.012480735778808594	1 results, cost = 0.012480735778808594
european lunch dinner	57 results, cost = 0.01159048080444336	57 results, cost = 0.0004322528839111328
greek	93 results, cost = 0.011188030242919922	93 results, cost = 1.52587890625e-05
bar	344 results, cost = 0.011151790618896484	344 results, cost = 0.00012087821960449219
italian reservations	487 results, cost = 0.011900901794433594	487 results, cost = 0.0007970333099365234
indian	936:results, cost: 0.011129140853881836	936:results, cost : 0.00013136863708496094
contemporary	164 results, cost = 0.012695550918579102	164 results, cost = 2.7894973754882812e-05
french dinner	122 results, cost = 0.012069463729858398	122 results, cost = 0.0002825260162353515 6
bar grill	224 results, cost = 0.011693000793457031	224 results, cost = 0.00011587142944335938
international vegetarian	7 results, cost = 0.011650323867797852	7 results, cost = 0.00011014938354492188

Από τα παραπάνω ενδεικτικά αποτελέσματα, καθώς και μετά από άλλες δοκιμές με διάφορα tags, προκύπτει ότι η kwSearchIF είναι γρηγορότερη της kwSearchRaw. Αυτό συμβαίνει γιατί η kwSearchIF χρησιμοποιεί το ανεστραμένο αρχείο, έναντι της kwSearchRaw. Μέσο του ανεστραμένου αρχείου βρίσκει γρηγορότερα σε ποιες γραμμές βρίσκονται τα tags, χωρίς να χρειάζεται να ψάξει όλο το αρχείο με τη σειρά όπως η kwSearchRaw.

Μέρος 2: Χωρικό ευρετήριο και χωρική αναζήτηση

Συνάρτηση readFile: Παίρνει όρισμα α αρχείο και κατασκευάζει το grid. Για κάθε γραμμή-εγγραφή βρίσκει τις συντεταγμένες τις οποίες κρατάει σε δύο διαφορετικούς πίνακες X και Y αντίστοιχα. Για να βρει το εύρος τιμών, για κάθε συντεταγμένη βρίσκει το min και το max της από τους αντίστοιχους πίνακες X και Y, και υπολογίζει το πλάτος του κάθε κελιού(stepX, stepY). Έπειτα ορίζει τα όρια γαι τις συντεταγμένες και σύμφωνα με αυτά πραγματοποιεί ελέγχους για να βάλει στα σωστά κελιά τις εγγραφές. Τέλος τυπώνει για κάθε διάσταση την μικρότερη και τη μεγαλύτερη τιμή των συντεταγμένων των εστιατορίων και το εύρος τιμών σε κάθε διάσταση και για κάθε κελί που δεν είναι άδειο τον αριθμό των εστιατορίων σε αυτό.

Συνάρτηση spaSearchRaw: Παίρνει ορίσματα τα min και max των τιμών(το range δηλαδή) και τον πίνακα με τις γραμμές-εγγραφές. Για κάθε line βρίσκει τις συντεταγμένες και αν βρίσκεται εντός του range τότε η εγγραφή τοποθετείται σε ένα πίνακα results, οποίος και επιστρέφεται.

Συνάρτηση spaSearchGrid: Παίρνει όρισμα την ορθογώνια περιοχή, το grid,τον πίνακα με τις εγγραφές. Υπολογίζει το πλάτος του κάθε κελιού(stepX, stepY), καθώς το πάνω και το κάτω όριο των συντεταγμένων. Έπειτα, διατρέχει το grid πραγματοποιώντας τους παρακάτω ελέγχους. Αν κάποιο από τα όρια του κελιού ξεφεύγει από τα όρια που έλαβε ως όρισμα, τότε περνάει στο επόμενο κελί ενημερώνοντας της μεταβλητές. Αλλιώς ελέγχει αν όλα τα όρια του κελιού βρίσκονται εντός των ορίων που δόθηκαν. Αν ισχύει, τότε προσθέτει στη λίστα resultsFiles τα περιεχόμενα του κελιού. Αν δεν ισχύει, τότε επαναληπτικά για κάθε περιεχόμενο του κελιού, το κρατάει σε μία μεταβλητή, βρίσκει τις συντεταγμένες του και ελέγχει αν βρίσκονται εντός των δοθέντων ορίων και βάζει τα περιεχόμενα στη λίστα resultsFiles. Τέλος, τοποθετεί σε μία λίστα results τις εγγραφές που περιέχουν κάθε στοιχείο της resultsFiles.

Για το τύπωμα των ζητούμενων: Καλείται η readFile με όρισμα το αρχείο Restaurants_London_England.tsv και η επιστροφή της μπαίνει σε 6 μεταβλητές, το grid, τη μικρότερη και τη μεγαλύτερη τιμή των συντεταγμένων(minX, maxX, minY, maxY) και τον πίνακα με τις εγγραφές. Παίρνει από τη γραμμή εντολών την ορθογώνια περιοχή και βλάζει τις τιμές σε 4 μεταβλητές. Για να μετρήσουμε το χρόνο εκτέλεσης, πριν από κάθε κλήση των συναρτήσεων καλείται η time για να πάρει το χρόνο έναρξης και αντίστοιχα μετά την κλήση για να πάρει το χρόνο λήξης. Ο χρόνος εκτέλεσης είναι η διαφορά των δύο χρόνων. Η κάθε συνάρτηση καλείται με τα αντίστοιχα ορίσμα και έπειτα τυπώνονται:

1. Ο αριθμός των αποτελεσμάτων της.
2. Το κόστος σε δευτερόλεπτα.
3. Οι εγγραφές της.

Σύγκριση αποδόσεων των συναρτήσεων:

query range	**spaSearchRaw**	**spaSearchGrid**
51.1 51.2 -0.5 1	5 results, cost = 0.009709835052490234	5 results, cost = 0.0001952648162841797
51.0 51.4 -0.78 0	55 results, cost = 0.010200023651123047	55 results, cost = 0.0003335475921630859 4
51.27 51.41 -0.68 1	95 results, cost = 0.011955738067626953	95 results, cost = 0.0002923011779785156
51.25 51.45 -0.5 0	538 results, cost = 0.009492635726928711	538 results, cost = 0.007345438003540039
51.3 51.98 -0.07 1	1505 results, cost = 0.009612321853637695	1505 results, cost = 0.0020618438720703125
51.0 51.569 -0.14 0	5195 results, cost = 0.011215925216674805	5195 results, cost = 0.009265422821044922
51.2 51.5 -0.58 1	3010 results, cost = 0.010166406631469727	3010 results, cost = 0.00759434700012207
51.5 51.8 -0.55 1	7814 results, cost = 0.010059118270874023	7814 results, cost = 0.009467363357543945
51.0 51.9 -0.27 0	9863 results, cost = 0.010140657424926758	9863 results, cost = 0.0048999786376953125
51.27 51.73 -0.68 1	10821 results, cost = 0.011543035507202148	10821 results, cost = 0.0016450881958007812

Από τα παραπάνω ενδεικτικά αποτελέσματα, καθώς και μετά από άλλες δοκιμές με διάφορα query ranges, προκύπτει ότι η spaSearchGrid είναι γρηγορότερη της spaSearchRaw. Αυτό συμβαίνει γιατί η spaSearchGrid χρησιμοποιεί το χωρικό ευρετήριο, έναντι της spaSearchRaw. Μέσο του χωρικού ευρετηρίου βρίσκει γρηγορότερα τους αριθμούς των γραμμών (εγγραφών) των εστιατορίων που πέφτουν μέσα στο αντίστοιχο διάστημα τιμών, χωρίς να χρειάζεται να ψάξει όλο το αρχείο με τη σειρά όπως η spaSearchRaw.

Μέρος 3: Χωρο-κειμενική αναζήτηση

Συνάρτηση createGrid: Παίρνει όρισμα α αρχείο και κατασκευάζει το grid. Για κάθε γραμμή-εγγραφή βρίσκει τις συντεταγμένες τις οποίες κρατάει σε δύο διαφορετικούς πίνακες X και Y αντίστοιχα. Για να βρει το εύρος τιμών, για κάθε συντεταγμένη βρίσκει το min και το max της από τους αντίστοιχους πίνακες X και Y, και υπολογίζει το πλάτος του κάθε κελιού(stepX, stepY). Έπειτα ορίζει τα όρια για τις συντεταγμένες και σύμφωνα με αυτά πραγματοποιεί ελέγχους για να βάλει στα σωστά κελιά τις εγγραφές. Τέλος τυπώνει για κάθε διάσταση την μικρότερη και τη μεγαλύτερη τιμή των συντεταγμένων των εστιατορίων και το εύρος τιμών σε κάθε διάσταση και για κάθε κελί που δεν είναι άδειο τον αριθμό των εστιατορίων σε αυτό.

Συνάρτηση createInvertedFile: Παίρνει όρισμα ένα αρχείο και το διαβάζει. Για κάθε γραμμή του αρχείου βρίσκει όλα τα tags. Σε ένα πίνακα value κρατάει κάθε γραμμή-εγγραφή που περιέχει το συγκεκριμένο tag. Έτσι, για κάθε tag μέσα στο λεξικό invertedFileDict, κρατάει το value που επιστρέφει σε μια μεταβλητή value. Σε αυτό το value προστίθεται και η νέα γραμμή στην οποία βρέθηκε το tag. Ενημερώνεται το value του λεξικού με key το tag με το νέο value. Ορίζει ένα σωρό και κρατάει σε ένα πίνακα words όλα τα κλειδιά του λεξικού, τα οποία είναι τα tags. Για κάθε tag παίρνω τον πίνακα με τους αριθμούς των γραμμών στις οποίες βρίσκεται. Σε ένα πίνακα value κρατάω τους αριθμούς γραμμώνπου βρίσκεται το tag και σε ένα πίνακα h βάζει τον value, το tag και τον lines(πίνακας με γραμμές-εγγραφές). Έπειτα βάζει τον h μέσα στον σωρό και ορίζει τον πίνακα invertedFile, που αντιπροσωπεύει το ανεστραμένο αρχείο στη μνήμη και ένα λέξικό keywordPositions. Τέλος φτιάχνει το ανεστραμένο αρχείο. Όσο ο σωρός έχει στοιχεία, τραβάει ένα ένα από αυτόν και τα τοποθετεί σε μια λιστα σε αυξουσα ταξινομηση ως προς το πληθος. Επιστρέφει το ανεστραμένο αρχείο, τον lines και το keywordPositions.

Συνάρτηση kwSearchIF: Παίρνει όρισμα μία λίστα με τις λέξεις αναζήτησης(search tags), με το ανεστραμένο αρχείο, τις γραμμές-εγγραφές και το λεξικό keywordPositions. Αρχικά κρατάει στον πίνακα position όλα τα values που επιστρέφονται από το keywordPositions με key το κάθε tag και τον ταξινομεί. Έπειτα βρίσκει το αποτέλεσμα. Το αποτέλεσμα(τις γραμμές-εγγραφές) είναι τα αρχεία που βρίσκονται στις λιστες όλων των keywords, επομένως παίρνουμε την πρώτη λίστα σαν αποτέλεσμα και μετά για κάθε επόμενο keyword βρίσκουμε την τομή τους με την mergejoin. Τέλος, μέσα σε ένα πίνακα results βάζει τις γραμμές-εγγραφές που περιέχουν το keyword.

Συνάρτηση kwSpaSearchRaw: Παίρνει ορίσματα τα min και max των τιμών(το range δηλαδή) και τον πίνακα με τις γραμμές-εγγραφές. Για κάθε line βρίσκει τις συντεταγμένες και αν βρίσκεται εντός του range τότε βρίσκει τα tags κάθε εγγραφής και τα βάζει σε ένα πίνακα. Κάθε tag ελέγχεται αν βρίσκεται στις λέξεις αναζήτησης

που δόθηκαν. Αν όλα τα tag βρίσκονται μέσα στη λίστα searchTags, τότε σε ένα πίνακα results μπαίνει η εγγραφη.

Συνάρτηση spaSearchGrid: Παίρνει όρισμα την ορθογώνια περιοχή, το grid,τον πίνακα με τις εγγραφές. Υπολογίζει το πλάτος του κάθε κελιού(stepX, stepY), καθώς και το πάνω και το κάτω όριο των συντεταγμένων. Έπειτα, διατρέχει το grid πραγματοποιώντας τους παρακάτω ελέγχους. Αν κάποιο από τα όρια του κελιού ξεφεύγει από τα όρια που έλαβε ως όρισμα, τότε περνάει στο επόμενο κελί ενημερώνοντας της μεταβλητές. Αλλιώς ελέγχει αν όλα τα όρια του κελιού βρίσκονται εντός των ορίων που δόθηκαν. Αν ισχύει, τότε προσθέτει στη λίστα resultsFiles τα περιεχόμενα του κελιού. Αν δεν ισχύει, τότε επαναληπτικά για κάθε περιεχόμενο του κελιού, το κρατάει σε μία μεταβλητή, βρίσκει τις συντεταγμένες του και ελέγχει αν βρίσκονται εντός των δοθέντων ορίων και βάζει τα περιεχόμενα στη λίστα resultsFiles. Τέλος, τοποθετεί σε μία λίστα results τις εγγραφές που περιέχουν κάθε στοιχείο της resultsFiles.

Συνάρτηση myMergeJoin: Παίρνει όρισμα δύο λίστες και ακολουθεί τον αλγόριθμο merge join για να συγχωνευτούν οι λίστες. Αυτό που επιστρέφεται είναι η τομή των λιστών.

Για το τύπωμα των ζητούμενων: Καλούνται η createGrid και η createInvertedFile με όρισμα το αρχείο Restaurants_London_England.tsv και οι επιστροφές τους μπαίνουν σε μεταβλητές, που θα χρησιμοποιηθούν για την κλήση των υπολοίπων συναρτήσεων. Παίρνει από τη γραμμή εντολών την ορθογώνια περιοχή και βλάζει τις τιμές σε 4 μεταβλητές και τις λέξεις αναζήτησης και τις βάζει σε ένα πίνακα searchTags. Για να μετρήσουμε το χρόνο εκτέλεσης, πριν από κάθε κλήση των συναρτήσεων καλείται η time για να πάρει το χρόνο έναρξης και αντίστοιχα μετά την κλήση για να πάρει το χρόνο λήξης. Ο χρόνος εκτέλεσης είναι η διαφορά των δύο χρόνων. Η κάθε συνάρτηση καλέιται με τα αντίστοιχα ορίσμα και έπειτα τυπώνονται:

1. Ο αριθμός των αποτελεσμάτων της.
2. Το κόστος σε δευτερόλεπτα.
3. Οι εγγραφές της.

Σύγκριση αποδόσεων των συναρτήσεων:

query range	**kwspaSearchRaw**	**kwSpaSearchIF**	kwSpaSearchGrid
51.1 51.2 -0.5 1 international	1 results, cost = 0.009701251983642578	1 results, cost = 0.000280618667602539 06	1 results, cost = 0.000160455703735351 56
51.27 51.73 -0.68 1	5 results, cost =	5 results, cost =	5 results, cost =

european bar	0.011914253234863281	0.0002422332763671875	0.01931929588317871
51.0 51.569 -0.14 0 sushi	24 results, cost = 0.015088081359863281	24 results, cost = 0.0001254081726074188 8	24 results, cost = 0.014231204986572266
51.3 51.98 -0.07 1 italian reservations	38 results, cost = 0.010760068893432617	38 results, cost = 0.0011913776397705078	38 results, cost = 0.003448963165283203
51.0 51.9 -0.27 0 english	643 results, cost = 0.01840996742248535	643 results, cost = 0.0007584095001220703	643 results, cost = 0.015802860260009766
51.5 51.8 -0.55 1 dinner	753 results, cost = 0.017328977584838867	753 results, cost = 0.0010519027709960938	753 results, cost = 0.016563892364501953

Από τα παραπάνω ενδεικτικά αποτελέσματα, καθώς και μετά από άλλες δοκιμές με διάφορα query ranges και tags, προκύπτει ότι η **kwSpaSearchIF** είναι γρηγορότερη τις άλλες δύο. Η kwSpaSearchGrid είναι γρηγορότερη της kwspaSearchRaw. Η kwSpaSearchRaw είναι η πιο αργή από τις τρεις επειδή εξετάζει όλο το αρχείο με τις εγγραφές. **Η kwSpaSearchIF είναι γρηγορότερη από την** kwSpaSearchGrid γιατί βρίσκει κατευθείαν τις γραμμές που βρίσκονται τα tags μέσο του ανεστραμένου αρχείου, σε αντίθεση με τη δεύτερη η οποία πρέπει να κάνει κάποιες συγκρίσεις με την ορθογώνια περιοχή και τα όρια των κελιών.

Παρατηρήσεις:

1. Για τον σωρό χρησιμοποιήθηκε έτοιμη βιβλιοθηκη της python(μετράει τις έγκυρες γραμμές του αρχείου males_sorted) και heapq), για να
πάρουμε τον χρόνο εκτέλεσης η βιβλιοθήκη time και για να πάρουμε είσοδο το
K η sys.
2. Τα αρχεία εκτελούνται με τις εντολές:

• Άσκηση 1: python3 part1.py <list of tags>
• Άσκηση 2: python3 part2.py <query range>
• Άσκηση 3: python3 part1.py <query range list of tags>