Assignment 4

r e a d m e

ΣΩΤΗΡΙΑ ΚΑΣΤΑΝΑ, 2995

ΜΕΡΟΣ Ι

>py part1.py italian ΄late night΄ dinner ΄dining on a budget΄

(με copy paste η παραπάνω εντολή δεν τρέχει σωστά λόγω των “΄”, πρέπει να πληκτρολογηθεί)

Σε αυτό το μέρος υλοποιούνται δύο συναρτήσεις η ‘kwSearchIF’ και η ‘kwSearchRaw’ οι οποίες επιστρέφουν τις εγγραφές του αρχείου ‘Restaurants_London_England.tsv’ (κάθε εγγραφή αντιστοιχεί σε ένα εστιατόριο) που περιέχουν τα query keywords που δέχονται σαν όρισμα. Για να το πετύχουν αυτό, η πρώτη χρησιμοποιεί ένα ανεστραμμένο αρχείο και η δεύτερη διαβάζει όλα τα δεδομένα από την λίστα που αναφέρεται στο αρχικό αρχείο. Αναλυτικότερα, η λίστα στην οποία αποθηκεύονται τα δεδομένα του αρχείου ‘Restaurants_London_England.tsv’ ονομάζεται ‘restaurants’ και το ανεστραμμένο αρχείο δημιουργείται με όνομα 'inverted_file.txt'. Για την δημιουργία του ανεστραμμένου αρχείου χρησιμοποιείται ένα λεξικό ‘inverted’ στο οποίο κλειδί είναι η κάθε ετικέτα και η τιμή του είναι η λίστα με τις εγγραφές που περιέχουν αυτή την ετικέτα. Στη συνέχεια ταξινομείται σε αύξουσα σειρά η λίστα ‘frequencies’ η οποία περιέχει την συχνότητα των ετικετών που εμφανίζονται στα δεδομένα. Για να κρατήσουμε τις ετικέτες που δίνονται από τον χρήστη στην γραμμή εντολών δημιουργείται μια λίστα ‘l_query_tags’ η οποία είναι το όρισμα των συναρτήσεων που αναφέρθηκαν παραπάνω. Για την δημιουργία αυτής της λίστας εξετάζεται και η περίπτωση στην οποία η ετικέτα είναι παραπάνω από μια λέξεις και συνεπώς βρίσκεται μέσα σε “΄΄”. Αφού εκτελεστούν τα παραπάνω βήματα καλούνται οι δύο συναρτήσεις οι οποίες αναλύονται παρακάτω.

Η συνάρτηση ‘kwSearchIF’ χρησιμοποιεί το λεξικό ‘inverted’, δηλαδή ουσιαστικά το ανεστραμμένο αρχείο. Για κάθε ετικέτα που δέχεται σαν όρισμα, εξετάζει αν υπάρχει μέσα στο λεξικό και αν ναι τότε παίρνει την αντίστοιχη ταξινομημένη λίστα με τις εγγραφές που αναφέρεται στη συγκεκριμένη ετικέτα και την συγχωνεύει με την ήδη υπάρχουσα λίστα ‘result_lines’ στην οποία κρατιούνται ουσιαστικά όλες οι εγγραφές αυτές. Επειδή η λίστα αυτή συγχωνεύεται με ‘merge’ τα στοιχεία της είναι ταξινομημένα σε αύξουσα σειρά, αυτό μας δίνει την δυνατότητα να μετρήσουμε τα στοιχεία (δηλαδή τις εγγραφές) τα οποία είναι ίδια στη λίστα, αυτό με επιτυγχάνεται με έναν έλεγχο μεταξύ διαδοχικών στοιχείων. Αν το πλήθος εμφάνισης των εγγραφών στη λίστα αυτή(με χρήση του μετρητή ‘counter_similar’) ισούται με το πλήθος των ετικετών που δόθηκαν σαν όρισμα, αυτό σημαίνει ότι όλες οι ετικέτες που ζητούνται ανήκουν στην συγκεκριμένη εγγραφή

του αρχείου και συνεπώς η εγγραφή προστίθεται στην λίστα ‘list_if’, η οποία μετά την κλήση της συνάρτησης θα επιστρέψει όλες αυτές τις εγγραφές που περιέχουν όλες τις ετικέτες, διαφορετικά ο μετρητής ‘counter_similar’ ξανά αρχικοποιείται σε 1. Να σημειωθεί πως γίνεται έλεγχος για το πλήθος των ετικετών που δίνονται σαν όρισμα, καθώς αν είναι ίσο με ένα δεν χρειάζεται η παραπάνω διαδικασία εφόσον οι εγγραφές που περιέχονται στην λίστα ‘result_lines’ πρέπει να εισαχθούν όλες στην λίστα ‘list_if’.

Τέλος η συνάρτηση ‘kwSearchRaw’ διατρέχει κάθε εγγραφή του αρχείου και για κάθε ετικέτα ελέγχει αν υπάρχει στην αντίστοιχη εγγραφή. Αν υπάρχει, αυξάνεται ο μετρητής ‘counter_lqts’ ο οποίος κρατάει το πλήθος των ετικετών που βρέθηκαν σε κάθε εγγραφή του αρχείου. Αν αυτός ο μετρητής γίνει ίσος με το μήκος των ετικετών που δόθηκαν σαν όρισμα, σημαίνει πως στην εγγραφή αυτή εμφανίζονται όλες οι ετικέτες που ζητήθηκαν και συνεπως προστίθεται η αντίστοιχη εγγραφή στη λίστα αποτελεσμάτων ‘list_raw’, η οποία επιστρέφεται μετά την κλήση της συνάρτησης. Η παραπάνω διαδικασία επαναλαμβάνεται για όλα τα δεδομένα του αρχείου ‘Restaurants_London_England.tsv’, όπου για κάθε έγγραφη μηδενίζεται και ο μετρητής ‘counter_lqts’.

ΜΕΡΟΣ ΙΙ

>part2.py 51 51.20 -0.5 0

Σε αυτό το μέρος υλοποιούνται δύο συναρτήσεις, η ‘spaSearchGri’ και ‘spaSearchRaw’, οι οποίες παίρνουν σαν όρισμα μια δισδιάστατη ορθογώνια περιοχή, range query, και επιστρέφουν τις εγγραφές του αρχείου ‘Restaurants_London_England.tsv’ στις οποίες η τοποθεσία των εστιατορίων στα οποία αναφέρονται βρίσκεται μέσα σε αυτή την περιοχή. Η πρώτη συνάρτηση χρησιμοποιεί ένα απλό χωρικό ευρετήριο βασισμένο σε σχάρα, ‘grid’, και η δεύτερη επιστρέφει τα αποτελέσματα διατρέχοντας απευθείας την λίστα στην οποία έχουν αποθηκευτεί τα δεδομένα του αρχείου, την ‘restaurants’. Για την δημιουργία του δισδιάστατου πίνακα ‘grid’ απαιτείται αρχικά η εύρεση της μικρότερης και της μεγαλύτερης τιμής για κάθε συντεταγμένη(x και y), συνεπώς δημιουργούνται δύο λίστες ‘sort_x’ και ‘sort_y’ οι οποίες περιέχουν όλα τα x και y των εγγραφών και αφού ταξινομηθούν υπολογίζονται εύκολα οι μικρότερες τιμές ως τις πρώτες θέσεις αυτών και οι μεγαλύτερες ως τις τελευταίες, όπου και αποθηκεύονται στις αντίστοιχες μεταβλητές (‘min_x’, ’min_y’, ’max_x’, ’max_x’). Έπειτα για να χωριστούν τα εύρη τιμών των συντεταγμένων σε 50 ίσα διαστήματα τιμών, αφαιρείται από την μεγαλύτερη τιμή η μικρότερη και το αποτέλεσμα διαιρείται με το 50 όπου και αποθηκεύεται στις μεταβλητές ‘rx’ και ’ry’ αντίστοιχα για κάθε συντεταγμένη. Στην συνέχεια δημιουργούνται δύο λίστες ‘list_x’ και ‘list_y’ η οποίες έχουν η κάθε μια μήκος 50 και σε κάθε θέση τους κρατούνται οι συντεταγμένες που θα αντιστοιχούν στα 50 διαστήματα τιμών κάθε κελιού για κάθε διάσταση. Αυτές οι συντεταγμένες των πρώτων διαστημάτων που αντιστοιχούν στο πρώτο κελί, υπολογίζονται για την μικρότερη τιμή(αριστερή -> ‘lrx/y’) αυτών, με ανάθεση των μικρότερων τιμών(‘min_x/y’) και για την μεγαλύτερη τιμή(δεξιά -> ‘rrx/y’) με πρόσθεση των μικρότερων τιμών και των διαστημάτων ‘rx/y’ αντίστοιχα για κάθε διάσταση. Στην συνέχεια για κάθε κελί ανανεώνονται αυτές οι τιμές με την αριστερή να παίρνει την τιμή της δεξιάς του προηγούμενου κελιού και την δεξιά να παίρνει την τιμή της τωρινής αριστερής αυξημένης κατά το διάστημα ‘rx/y’. Αφού δημιουργηθούν αυτές οι βοηθητικές λίστες ξεκινάει η δημιουργία του δισδιάστατου πίνακα ‘grid’ στον οποίο αρχικά αποθηκεύεται σε όλες τις θέσεις του μια κενή λίστα. Για κάθε εγγραφή των εστιατορίων, αφού βρεθούν οι συντεταγμένες της τοποθεσίας, ελέγχεται αν η συντεταγμένη x ανήκει μέσα σε κάποιο από τα διαστήματα που υπάρχουν σε κάθε θέση της λίστας ‘list_x’ και με το που βρεθεί σταματάει την αναζήτηση, γνωρίζοντας την θέση της λίστας στην οποία βρέθηκε η συντεταγμένη. Αντίστοιχα ακολουθεί την ίδια διαδικασία και για την συντεταγμένη y. Εφόσον έχει τελειώσει με τις αναζητήσεις στις βοηθητικές λίστες, γνωρίζει την

θέση του κελιού όπου θα πρέπει να πάει να γραφεί στον πίνακα ‘grid’, έτσι πηγαίνει και προσθέτει τον αριθμό της αντίστοιχης εγγραφής σε αυτό, καθώς η τοποθεσία που αναφέρεται σε αυτήν την εγγραφή πέφτει μέσα στο αντίστοιχο διάστημα τιμών. Τα κελία του ‘grid’ στα οποία δεν πέφτει καμία εγγραφή μένουν με τις κενές λίστες. Στην συνέχεια τυπώνονται για κάθε διάσταση η μικρότερη και η μεγαλύτερη τιμή των συντεταγμένων των εστιατορίων, το εύρος τιμών σε κάθε διάσταση και ο αριθμός των εστιατορίων σε κάθε κελί που δεν είναι άδειο. Τέλος καλούνται οι δύο συναρτήσεις όπου αναλύονται παρακάτω.

Η συνάρτηση ‘spaSearchGrid’ αρχικά υπολογίζει τέσσερεις αριθμούς (‘first_x’, ‘last_x’, ‘first_y’, ‘last_y’) όπου αντιστοιχούν στις θέσεις για κάθε διάσταση των κελιών του ‘grid’ όπου μέσα σε αυτά τα κελιά πέφτουν τα όρια της δισδιάστατης περιοχής που δόθηκε σαν όρισμα. Για να υπολογιστούν αυτοί οι αριθμοί μπορεί να γίνει είτε με χρήση των βοηθητικών λιστών ‘list_x/y’ (κώδικας σε σχόλια) είτε με τον υπολογισμό των ‘grid_min/max_x/y’ που θα έχουν τιμές ίδιες με αυτές που υπάρχουν σε κάθε θέση των λιστών. Εφόσον είναι γνωστά τα κελιά στα οποία πέφτει μέσα το range query διατρέχονται μόνο αυτά. Για τα κελιά τα οποία έχει βρεθεί πως πέφτουν τα όρια (δηλαδή η θέση των δύο διαστάσεων είναι είτε ‘first_x/y’ είτε ‘last_x/y’) και δεν είναι κενά, για κάθε τιμή που υπάρχει σε αυτά και δείχνει στην αντίστοιχη εγγραφή,πηγαίνει απευθείας στην αντίστοιχη εγγραφή και ελέγχει εάν η τοποθεσία ανήκει στη δοθείσα περιοχή και αν ναι αποθηκεύεται στην λίστα ‘list_rests’. Αυτός ο έλεγχος γίνεται για την περίπτωση που η συντεταγμένη του query range ανήκει μέσα σε αυτό το διάστημα που ορίζεται από το κελί, όμως δεν είναι απαραίτητο πως όλες οι τιμές που δείχνουν σε εγγραφές που περιέχονται σε αυτό έχουν συντεταγμένες τοποθεσίας που περιέχονται στο query range. Για όλα τα υπόλοιπα μη κενά κελιά, καθώς είναι σίγουρο πως όλες οι εγγραφές στις οποίες δείχνουν ανήκει η τοποθεσία τους στο query range γίνεται απευθείας εισαγωγή αυτών στην λίστα ‘list_rests’.

Τέλος η συνάρτηση ‘spaSearchRaw’ ελέγχει για κάθε εγγραφής της λίστας ‘restaurants’ αν οι συντεταγμένες της τοποθεσίας του αντίστοιχου εστιατορίου ανήκουν μέσα στον διάστημα τιμών που δέχθηκε σαν όρισμα, αν ναι τότε τις εισάγει στην λίστα ‘list_rests’ όπου και την επιστρέφει αφού διαβάσει όλα τα δεδομένα.

ΜΕΡΟΣ ΙΙΙ

>py part3.py 51 51.5 -0.5 0 ‘late night’ breakfast/brunch dinner takeout ‘large groups’

Σε αυτό το μέρος υλοποιούνται τρεις συναρτήσεις, η ‘kwSpaSearchIF’, η ‘kwSpaSearchGrid’ και η ‘kwSpaSearchRaw’, οι οποίες παίρνουν ως όρισμα ένα query range και μια λίστα με τουλάχιστον μια query keyword και επιστρέφουν ως αποτέλεσμα τις εγγραφές οι οποίες η τοποθεσία τους περιέχεται μέσα στο query range και περιέχουν όλα τα query keywords. Η πρώτη χρησιμοποιεί το ανεστραμμένο αρχείο που περιγράφτηκε στο μέρος 1, η δεύτερη χρησιμοποιεί το grid που περιγράφτηκε στο μέρος 2 και η τρίτη διαβάζει απευθείας τα δεδομένα από την λίστα ‘restaurants’. Η δημιουργία του ανεστραμμένου αρχείου και του grid ακολουθεί την ίδια διαδικασία που αναφέρθηκε παραπάνω στα αντίστοιχα κομμάτια, συνεπώς παραλείπεται.

Πιο αναλυτικά, η ‘kwSpaSearchIF’ ακολουθεί την ίδια λογική με την ‘kwSearchIF’ του μέρους 1 με την μόνη διαφορά ότι εφόσον βρεθεί, με την βοήθεια του ‘inverted’, ότι μια εγγραφή περιέχει όλα τα keywords που ζητούνται στην συνέχεια ελέγχεται εάν η τοποθεσία της συγκεκριμένης εγγραφής πέφτει μέσα στο query range που δόθηκε. Εάν ναι, τότε μόνο εισάγεται η αντίστοιχη εγγραφή στην λίστα ‘list_if’ με τα αποτελέσματα. Αντίστοιχα η ‘kwSpaSearchGrid’

βασίζεται στην λογική της ‘spaSearchGrid’ του μέρους 2 με την διαφορά ότι αφού βρεθούν οι εγγραφές που υπάρχουν στα κελιά του grid, οι συντεταγμένες των οποίων πέφτουν μέσα στο query range, ελέγχεται αν κάθε τέτοια εγγραφή περιέχει όλα τα query keywords που δόθηκαν. Εάν ναι, τότε και μόνο εισάγεται η αντίστοιχη εγγραφή στη λίστα ‘list_rests’ με τα αποτελέσματα. Σημειώνεται πως εδώ, καθώς η διαδικασία για τον έλεγχο των query keywords σε κάθε εγγραφή είναι ίδια και για τα κελιά στα οποία πέφτουν τα όρια του range query και για τα υπόλοιπα, χρησιμοποιούνται δύο μεταβλητές ‘bool1’ και ‘bool2’ οι οποίες βοηθούν να διαχωριστεί η περίπτωση στην οποία μια εγγραφή περιέχεται σε ένα κελί (σε αυτό που ορίζει τα όρια) όμως δεν περιέχεται στο range query. Τέλος, η ‘kwSpaSearchRaw’ αποτελεί την μίξη της ‘kwSearchRaw’ του μέρους 1 και της ‘spaSearchRaw’ του μέρους 2, διαβάζει δηλαδή απευθείας τα δεδομένα της λίστας ‘restaurants’ και αν μια εγγραφή αναφέρεται σε τοποθεσία εστιατόριου όπου ανήκει στο range query και περιέχει στις ετικέτες του όλα τα query keywords, τότε προστίθεται στην λίστα ‘list_raw’ με τα αποτελέσματα.

ΜΕΡΟΣ IV

α) την απόδοση της kwSearchIF σε σχέση με την kwSearchRaw στο Μέρος 1

β) την απόδοση της spaSearchGrid σε σχέση με την spaSearchRaw στο Μέρος 2

γ) τη σχετική απόδοση των τριών συναρτήσεων στο Μέρος 3

α) Η ‘*kwSearchIF*’ σε όλες τις δοκιμές είναι πολύ πιο γρήγορη από την ‘*kwSearchRaw*’, συνεπώς και πιο αποδοτική, κάτι το οποίο είναι λογικό καθώς η δεύτερη ψάχνει σε όλες τις εγγραφές του αρχείου. Επίσης παρατηρείται πως όσο αυξάνεται το πλήθος των query keywords η ‘*kwSearchRaw*’ γίνεται ακόμα πιο αργή. Αυτό συμβαίνει καθώς για να βρει κάθε φορά τα αποτελέσματα, διατρέχει για κάθε εγγραφή του αρχείου την λίστα με τα keywords, συνεπώς όσο αυξάνεται το πλήθος, αυξάνονται και οι επαναλήψεις και επομένως μειονεκτεί ακόμα περισσότερο στην αποδοτικότητα.

Αντίθετα η ‘*kwSearchIF*’ αντί να προσπελάσει όλες τις εγγραφές, ψάχνει κάθε φορά στο ανεστραμμένο αρχείο (το οποίο έχει αποθηκευτεί σε λεξικό) το οποίο έχει πολύ μικρότερο μήκος καθώς περιέχει θέσεις τόσες όσες οι διαφορετικές ετικέτες του αρχείου. Το ανεστραμμένο αρχείο δείχνει για κάθε ετικέτα τις εγγραφές του αρχείου οι οποίες περιέχουν την αντίστοιχη ετικέτα. Έτσι, με το που βρει ότι σε μια εγγραφή υπάρχουν όλες οι ετικέτες (κλειδιά στο λεξικό) που αντιστοιχούν στις query keywords πηγαίνει και παίρνει κατευθείαν αυτή την εγγραφή.

β) Παρατηρείται πως η ‘*spaSearchGrid*’ είναι πιο γρήγορη και πιο αποδοτική σε σχέση με την ‘*spaSearchRaw*’ , αλλά και σε σχέση με τον εαυτό της όσο μικρότερα είναι τα διαστήματα που της δίνονται ως είσοδο. Η ‘*spaSearchRaw*’ έχει την ίδια απόδοση για όποιο διάστημα της δοθεί, καθώς κάθε φορά διατρέχει και ελέγχει όλες τις εγγραφές του αρχείου.

Αντίθετα η ‘*spaSearchGrid*’ χρησιμοποιεί ένα χωρικό ευρετήριο (δομή δισδιάστατου πίνακα) το οποίο κάθε κελί του δείχνει σε όλες οι εγγραφές του αρχείου, οι οποίες αναφέρονται σε εστιατόρια των οποίων οι τοποθεσίες περιέχονται στα διαστήματα που αυτό ορίζει. Έτσι κάθε φορά πηγαίνει και εισάγει απευθείας αυτές τις εγγραφές στην λίστα που θα επιστραφεί ως αποτέλεσμα χωρίς να χρειάζεται να ελέγξει αν ανήκουν στο range query (εκτός βέβαια των περιπτώσεων που στα διαστήματα των κελιών πέφτουν τα όρια του range query και πρέπει να γίνει ο έλεγχος για κάθε εγγραφή). Όταν το range query είναι μικρό, σημαίνει πως θα προσπελαστούν και λιγότερα κελιά του grid και συνεπώς ο χρόνος θα είναι λιγότερος.

γ) Όταν το range query είναι μεγάλο η πιο αποδοτική είναι η ‘*kwSpaSearchIF*’ είτε τα query keywords είναι πολλά είτε ένα. Ωστόσο, σε αυτή την περίπτωση, όσο πιο λίγα είναι τα keywords τόσο πιο αποδοτική είναι η ‘*kwSpaSearchGrid*’ σε σχέση με την ‘*kwSpaSearchRaw*’ ενώ όσο αυξάνονται, κάποια στιγμή η ‘*kwSpaSearchRaw*’ εκτελείται γρηγορότερα. Από την άλλη όταν το range query είναι μικρό η πιο αποδοτική είναι η ‘*kwSpaSearchGrid*’ είτε τα keywords είναι πολλά είτε λίγα. Η ‘*kwSpaSearchIF*’ για μικρό πλήθος keywords είναι το ίδιο αποδοτική με την ‘*kwSpaSearchGrid*’ καθώς αυξάνονται όμως υστερεί. Παρά την μείωση αποδοτικότητας της όμως συνεχίζει να είναι πολύ πιο αποδοτική από την ‘*kwSpaSearchRaw*’, καθώς και αυτήν όσο αυξάνονται τα keyowrds γίνεται και πιο αργή.