***Τσόχλας Ιωάννης***

AM 4993 | cs04993@uoi.gr

**ΜΥΕ037**

**Ψηφιακή Επεξεργασία Εικόνας**

**ΕAΡΙΝΟ ΕΞΑΜΗΝΟ 2023-2024**

***ΕΡΓΑΣΙΑ***

**Project Goal :**

Η εργασία αποτελείται από 2 μέρη:

* Στο πρώτο μέρος μας ζητείται να υλοποιήσουμε τη συνάρηση **image\_patches.** Η συνάρτηση αυτή διαιρεί μια ασπρόμαυρη εικόνα σε ένα σύνολο από μη επικαλυπτόμενα τμήματα εικόνας 16 επί 16 pixel.Έπειτα παρατίθενται γιατι είναι καλό τα τμήματα να έχουν μηδενική μέση τιμή και για το αν θα ήταν καλά ή κακά για προβλήματα όπως η αντιστοίχιση (matching) ή η αναγνώριση ενός αντικειμένου.
* Στο δεύτερο μέρος, μας ζητείται να υλοποιήσουμε τη συνάρτηση **convolve()** για τη συνέλιξη μιας εικόνας με χρήση συμπλήρωσης μηδενικών και αντιστροφή του πυρήνα φίλτρου. Επίσης, να δείξουμε ότι η συνέλιξη με 2Δ Gaussian φίλτρο ισοδυναμεί με διαδοχική εφαρμογή δύο 1Δ Gaussian φίλτρων. Θα εφαρμόσουμε ένα Gaussian φίλτρο στην εικόνα grace hopper.png θα συζητήσουμε τη σημασία της κανονικοποίησης του φίλτρου, θα υπολογίσουμε τους πυρήνες για τις παραγώγους της εικόνας και θα υλοποιήσουμε τη ρουτίνα **edge\_detection()** για την ανίχνευση ακμών, συγκρίνοντας τα αποτελέσματα σε αρχική και φιλτραρισμένη εικόνα.

**Μέρος 1 Τοπικές περιοχές/Τμήματα εικόνας (Patches)**

1. Συμπληρώστε τη συνάρτηση **image\_patches**.

Αξιοσημείωτο θεωρώ να επισημάνω τις αλλαγές πλαισίων που έκανα για να τρέξει ο κώδικας μιας και μετά την υλοποίηση της **image\_patches**, uπήρχαν errors.

* Debug για τη διαχείριση των errors (τα παρακάτω αντικείμενα είναι κενά γιατί δεν τα έχουμε υλοποιήσει ακόμα για αυτό το λόγο βάζουμε μια συνθήκη τύπου try catch για να καταφέρουμε να έχουμε κάποιο αποτέλεσμα στην κονσόλα:

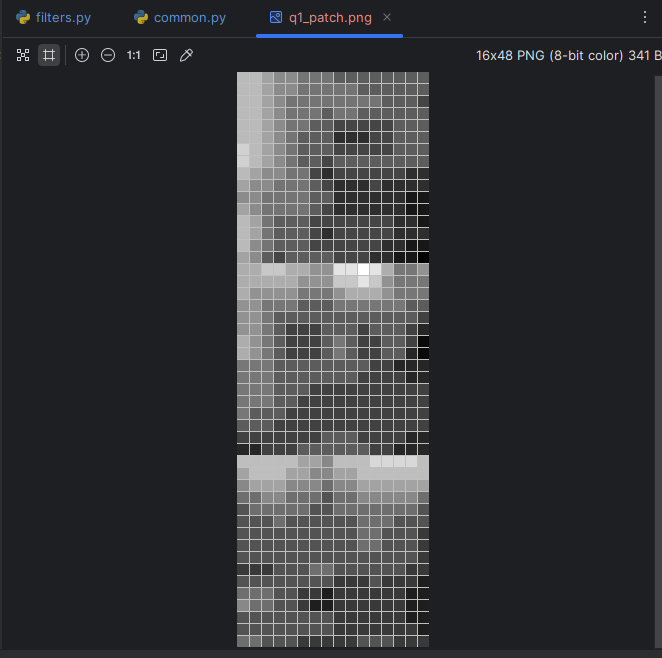
Εικόνα που περιέχει κείμενο, λογισμικό, λογισμικό πολυμέσων, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

* Η κονσόλα μας επισήμανε να αλλάξουμε τη μεταβλητή n.float σε np.float64

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα



Τα 3 τμήματα τις εικόνας από το αρχείο grace\_hopper.png

1. Η κανονικοποίηση των τμημάτων της εικόνας ώστε να έχουν μηδενική μέση τιμή έχει αρκετά πλεονεκτήματα . Bοηθά στη βελτίωση της ποιότητας της εικόνας και της απόδοσης των αλγορίθμων επεξεργασίας εικόνας, μειώνοντας παράλληλα την επίδραση του θορύβου.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα

**Υπόδειξη:**

Το δεύτερο σενάριο, όπου η τιμή της σκοτεινής αντιστοιχεί σε -1 και η τιμή της φωτεινής σε 1, είναι πιο κατάλληλο για να μετρήσουμε την ομοιότητα χρησιμοποιώντας εσωτερικά γινόμενα. Αυτό οφείλεται στο γεγονός ότι η συμμετρική αυτή κλίμακα παρέχει μια πιο ισορροπημένη και λεπτομερή μέτρηση των διαφορών φωτεινότητας

1. 1. Όταν το αντικείμενο μετακινείται μέσα στην εικόνα, η μηδενική μέση τιμή διασφαλίζει ότι η συνολική φωτεινότητα της εικόνας δεν επηρεάζεται από τη θέση του αντικειμένου, γιατι οι θετικές και αρνητικές τιμές θα αλληλοακυρώνονται.

2.Η αλλαγή κλίμακας μπορεί να επηρεάσει την ένταση των τιμών των εικονοστοιχείων.

3.Η σταθερή μέση τιμή βοηθά στη διατήρηση της σύγκρισης ανεξάρτητα από την ένταση του φωτός.

# Συμπέρασμα

Η χρήση τμημάτων με μηδενική μέση τιμή είναι ιδιαίτερα χρήσιμη για προβλήματα όπως η αντιστοίχιση και η αναγνώριση αντικειμένων.

**Μέρος 2****Φιλτράρισμα στο πεδίο του χώρου**

Για τον δεύτερο αλγόριθμο έχω 1 συνάρτηση dummy(given\_file) και μια main() στην οποία υλοποιώ τον αλγόριθμο B Top-K join.

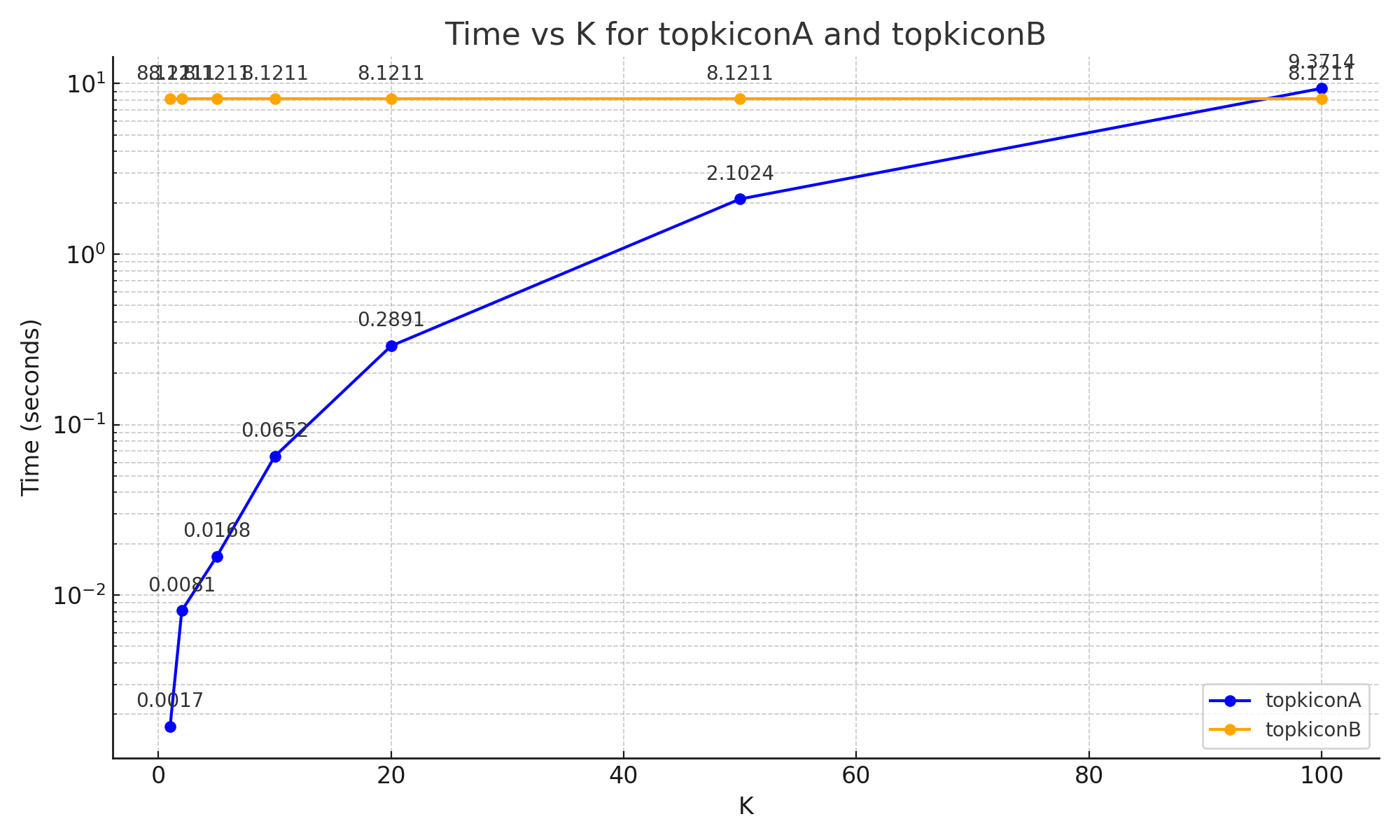
Η συνάρτηση dummy(given\_file) χρησιμοποιείται σαν generator και απλά μου επιστρέφει την επόμενη πλειάδα από το αρχείο females\_sorted. Η πρώτη δουλειά μέσα στην main() είναι να κάνω μια for loop σε όλο το αρχείο males\_sorted και να κρατήσω όλες τις «έγκυρες πλειάδες» σε ένα λεξικό L1. Δημιουργώ μια τοπική λίστα που έχει 3 εγγραφές: το id, το βάρος και την ηλικία του άντρα. Το λεξικό έχει για κλειδί το age πεδίο του και η τιμή του είναι μια λίστα από λίστες. Αν μιλάμε για την πρώτη εγγραφή του συγκεκριμένου κλειδιού τότε απλά βάζω την τοπική λίστα που δημιούργησα πιο πάνω. Αν όμως στο συγκεκριμένο κλειδί έχω ήδη κάποια τιμή τότε, επεκτείνω την τιμή με την τοπική λίστα που δημιούργησα καθώς διάβασα την πλειάδα από το αρχείο.

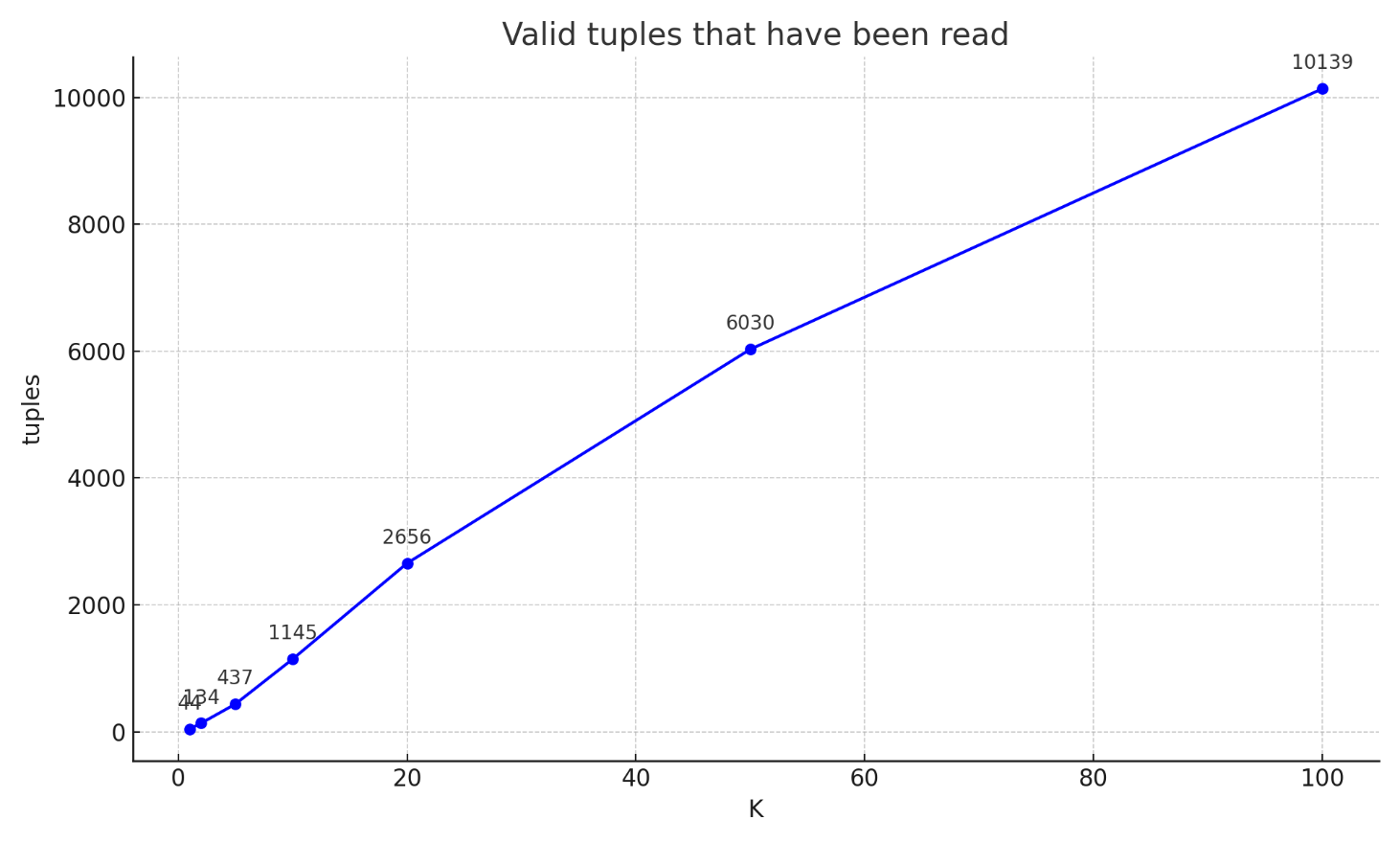
Έπειτα κάνω μια for loop στον generator(για το αρχείο females\_sorted) και αποθηκεύω σε τοπικές μεταβλητές τα απαραίτητα πεδία. Αγνοώ τις μη έγκυρες πλειάδες και προχωρώ στον έλεγχο για τα join. Για να κάνω τον έλεγχο, απλά κάνω μια for loop στο λεξικό των αντρών και συγκεκριμένα στην θέση L1[female\_age]. Έτσι εξασφαλίζω ότι έχω σίγουρα join με τις συγκεκριμένες πλειάδες. Μέσα στην for μετράω πόσες πλειάδες προσπέλασα. Αν είναι ο αριθμός μικρότερος ή ίσος από το K\_value απλώς κάνω push το ζευγάρι στην ουρά μου(Q), αλλιώς για κάθε επόμενη πλειάδα ελέγχω την κεφαλή της ουράς. Αν η τιμή του «πιθανού» join είναι μεγαλύτερη από την κεφαλή τότε κάνω pop την κεφαλή και push το νέο ζευγάρι, αλλιώς συνεχίζω τον έλεγχο για το επόμενο ζευγάρι.

Μόλις τελειώσω τον έλεγχο για όλα τα females, κάνω pop όλα τα στοιχεία της ουράς (Q) σε μια νέα λίστα. Ξέρω ότι η ουρά περιέχει ακριβώς K\_value στοιχεία και ότι τα στοιχεία στην λίστα θα μπούνε σε αύξουσα σειρά, μιας και η ουρά μου είναι min-heap. Επομένως κάνω reverse την λίστα μου και τυπώνω: τα ζευγάρια και τους χρόνους εκτέλεσης

**Μέρος 3**

Παρακάτω παρουσιάζω τους χρόνους εκτέλεσης (A Top-K join, Β Top-K join )καθώς και τις έγκυρες πλειάδες που προσπελάστηκαν κατά την διάρκεια εκτέλεσης του A Top-K join (HRJN). Να σημειωθεί ότι οι χρόνοι που εκτυπώνονται στο τερματικό είναι σε seconds, ενώ στα γραφήματα έχουν μετατραπεί σε mseconds(\*10^-3)





Στο δεύτερο γράφημα βλέπουμε τις πλειάδες που προσπελάστηκαν από το males\_sorted

και από το females\_sorted. Ουσιαστικά είναι ίδια τιμή και για τα δύο. Επίσης τα τυπώνω

και στο τερματικό.

Από το πρώτο γράφημα καταλαβαίνουμε ότι ο Αλγόριθμος Α είναι μέχρι ένα σημείο, και πιο

συγκεκριμένα μέχρι μια τιμή του Κ, πιο βελτιωμένος σε σύγκριση με τον Αλγόριθμο Β. Αυτό

γίνεται διότι για σχετικά μικρές τιμές του Κ(1,2,5,10,20,50,100) ο Α δημιουργεί δυναμικά τα

topK ζευγάρια σε αντίθεση με τον Β που ο αριθμός του Κ δεν επηρεάζει τον χρόνο

εκτέλεσης του. Ο χρόνος εκτέλεσης του Β επηρεάζεται αποκλειστικά από το μέγεθος των

αρχείων δεδομένων.

Ο Α αλγόριθμος είναι πιο γρήγορος, μέχρι την τιμή Κ = 100, διότι διαβάζει εναλλάξ τις

πλειάδες από τα αρχεία δεδομένων και έτσι «προλαβαίνει» να δημιουργήσει τα topK

ζευγάρια δυναμικά. Όταν όμως προχωράμε σε μεγαλύτερες τιμές, η ουρά (max-heap) του Α

γίνεται υπερβολικά μεγάλη σε μέγεθος και απαιτούνται πολλοί έλεγχοι και υπολογισμοί.

Ο Β αλγόριθμος έχει σταθερό χρόνο εκτέλεσης, για οποιαδήποτε τιμή του Κ, και αυτό

γίνεται γιατί διαβάζει εξ ολοκλήρου το αρχείο των αντρών και οι συγκρίσεις γίνονται με τις

πλειάδες των γυναικών. Επίσης ο Β αλγόριθμος χρησιμοποιεί μια ουρά (min-heap) όπου το

μέγεθος της δεν ξεπερνάει την τιμή του K-value καθώς ελέγχουμε πάντα το στοιχείο της

κεφαλής και ανάλογα την περίπτωση το αλλάζουμε.

***Συμπέρασμα:***

Το πλεονέκτημα του Α έναντι του Β είναι ότι, για μικρές τιμές του Κ ο Α αλγόριθμος είναι

γρηγορότερος.

Το μειονέκτημα του Α έναντι του Β είναι ότι η εκτέλεση του Α, για μεγάλες τιμές, αργεί

σημαντικά σε σύγκριση με τον Β που τρέχει σε σταθερό χρόνο.