ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

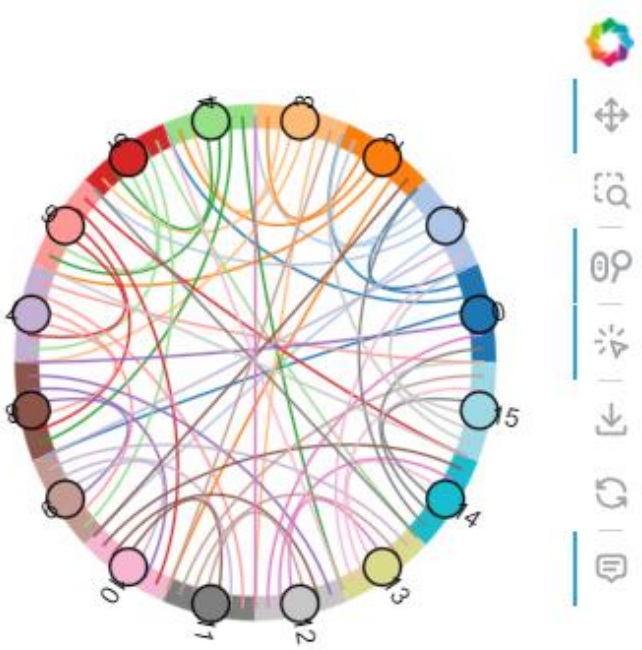
ΤΕΧΝΟΛΟΓΙΕΣ ΑΠΟΚΕΝΤΡΩΜΕΝΩΝ ΔΕΔΟΜΕΝΩΝ

Όνομα	ΧΡΙΣΤΙΝΑ-ΕΛΕΑΝΝΑ
Επώνυμο	ΣΑΜΑΡΑ
Έτος	4ο
ΑΜ	1084622

Ημερομηνία Παράδοσης: 11/02/2024

Περιεχόμενα

Εισαγωγή



Σκοπός του έργου είναι η ανάπτυξη ενός αποκεντρωμένου ευρετηρίου (Decentralized Index) βασισμένου σε κατανεμημένο κατακερματισμό και υλοποιημένου σε περιβάλλον Python. Η αξιολόγηση του έργου έγινε πειραματικά, χρησιμοποιώντας πραγματικά σύνολα δεδομένων. Οι κύριες λειτουργίες περιλαμβάνουν τη δημιουργία του ευρετηρίου, την εισαγωγή, τη διαγραφή, την ενημέρωση κλειδιών, την αναζήτηση κλειδιών, την προσθήκη κόμβου, τη διαγραφή ή αποτυχία κόμβου, ερωτήματα εύρους, ερωτήματα ομοιότητας και άλλες σχετικές λειτουργίες.

Dataset

Το dataset που χρησιμοποιήθηκε αποτελείται από τα ονοματεπώνυμα επιστημόνων πληροφορικής, τα πανεπιστήμια στα οποία σπούδασαν, καθώς και τον αριθμό των βραβείων που έχουν.

Web Crawler

Ο web crawler που υλοποιήθηκε πραγματοποιεί την ανάκτηση δεδομένων από τη Wikipedia σχετικά με επιστήμονες της πληροφορικής και αποθηκεύει αυτά τα δεδομένα σε ένα αρχείο CSV. Αρχικά, εισάγονται οι απαραίτητες βιβλιοθήκες (requests, BeautifulSoup, csv, re) για την ανάκτηση δεδομένων από το web, το parsing HTML, τη δημιουργία αρχείων CSV και την επεξεργασία κειμένου. Κατόπιν, γίνεται μια αίτηση HTTP GET στη διεύθυνση URL της Wikipedia που περιέχει τη λίστα των επιστημόνων της πληροφορικής. Αν η αίτηση είναι επιτυχής (κωδικός κατάστασης 200), προχωράμε στην ανάλυση της σελίδας, αλλιώς εκτυπώνετε κατάλληλο μήνυμα σφάλματος.

Μετά από την επιτυχή αίτηση, αποθηκεύουμε όλα τα link των σελιδών των επιστημόνων στην λίστα **scientist_links**. Διατρέχοντας αυτήν την λίστα, ακολουθείται η εξής διαδικασία:

Αρχικά, παίρνουμε το όνομα του επιστήμονα από το link της σελίδας του αντικαθιστώντας τις κάτω παύλες με κενό. Από το string που θα πάρουμε ως αποτέλεσμα, αφαιρούμε με κανονική έκφραση τυχόν παρενθέσεις που μπορεί να υπάρχουν, όπως *(computer scientist), (programmer)* κ.ά., καθώς παρατηρήθηκε ότι πολλές σελίδες της Wikipedia προσθέτουν στο όνομα την ιδιότητα του ατόμου για διαχωρισμό του από άλλους με το ίδιο όνομα. Επίσης, χρησιμοποιούμε από την βιβλιοθήκη urllib.parse την συνάρτηση unquote, έτσι ώστε μη λατινικοί χαρακτήρες να αναπαρασταθούν σωστά και όχι με την url μορφή τους. Για παράδειγμα Corrado B%C3%B6hm σε Corrado Böhm.

Στον html κώδικα της σελίδας ψάχνουμε να βρούμε το στοιχείο **infobox**, το οποίο περιέχεται σε όλες τις σελίδες της Wikipedia και περιλαμβάνει βασικές πληροφορίες για το αντικείμενο της σελίδας. Στην περίπτωσή μας, αν η Wikipedia έχει πληροφορίες για την εκπαίδευση (alma mater) και τα βραβεία (awards) του εν λόγω επιστήμονα, αυτές θα περιέχονται στο infobox.

Διατρέχοντας τις γραμμές του πίνακα infobox, ψάχνουμε για αυτές τις δύο επικεφαλίδες.

Όσον αφορά το Alma mater, τα πανεπιστήμια που φοίτησε (αν είναι πάνω από ένα) θα διαχωρίζονται με κόμμα. Επειδή εμείς θα αποθηκεύσουμε το dataset μας σε αρχείο csv, διαχωρίζουμε τα πανεπιστήμια χρησιμοποιώντας ως delimiter τον χαρακτήρα «@». Αν δεν υπάρχει το Alma mater στον πίνακα infobox, δεν κάνουμε εγγραφή στο dataset για τον συγκεκριμένο επιστήμονα. Από την λίστα που αποθηκεύουμε τα πανεπιστήμια ελέγχουμε αν υπάρχουν στοιχεία όπως "MS", "BS", "BSc", "MSc", "PhD", "B.S.", "M.S.". Η Wikipedia συχνά συμπεριλαμβάνει το επίπεδο σπουδών του επιστήμονα, κάτι που εμάς δεν μας ενδιαφέρει για το συγκεκριμένο dataset.

Όσον αφορά τα βραβεία, αυτά επίσης διαχωρίζονται με κόμμα (αν είναι παραπάνω από 1), οπότε μετράμε το μήκος της λίστας που τα βρίσκει. Αν δεν έχουμε καμία πληροφορία για τα βραβεία, τα βάζουμε μηδέν.

Στο τέλος τα δεδομένα μας αποθηκεύονται στο αρχείο “data.csv” και αποτελούνται από 397 γραμμές.

Chord: υλοποίηση

Βοηθητικές συναρτήσεις

Κάθε κόμβος του Chord δημιουργείται (**__init__()**) παίρνοντας έναν μοναδικό αριθμό (id) σαν παράμετρο. Παράλληλα, αρχικοποιούνται: το λεξικό (**data**) του κόμβου στο οποίο θα αποθηκευτούν τα δεδομένα, ο πίνακας **fingertable** του κόμβου, καθώς και μια μεταβλητή **prev**, στην οποία αποθηκεύεται ο προηγούμενος κόμβος.

Ο πίνακας data του κάθε κόμβου έχει την μορφή εμφωλευμένου λεξικού. Ως κλειδιά περιέχονται τα διακριτά πανεπιστήμια που έχουν ανακτηθεί από το dataset. Οι τιμές τους είναι πάλι λεξικά, με κλειδιά τα ονόματα των επιστημόνων και τιμές τον αριθμό βραβείων τους.

Ο πίνακας fingerTable κάθε κόμβου περιλαμβάνει K τιμές, όπου $2^K = SIZE\ του\ Chord$. Η είσοδος I του κόμβου με id n θα περιέχει τον διάδοχο $\left((n + 2^i)mod(SIZE)\right.$. Η πρώτη είσοδος του πίνακα είναι ο αμέσως επόμενος κόμβος.

getHashId(key): Δίνοντας το κλειδί (κάποιον αριθμό) επιστρέφεται ο χασαρισμένος αριθμός, ανάλογα με το μέγεθος του Chord.

distance(id1, id2): Επιστρέφεται η απόσταση των δύο id στον κύκλο.

updateFingerTable(): Διαγράφονται τα στοιχεία του fingertable (εκτός του πρώτου), και ενημερώνονται εκ νέου βρίσκοντας τους κατάλληλους κόμβους με την βοήθεια της συνάρτησης lookupNode().

Αρχείο **hashing**:

Το αρχείο hashing περιλαμβάνει δύο συναρτήσεις: την **sort_characters()** και την **hashed()**. Η πρώτη παίρνει ως όρισμα ένα κείμενο και, αφού αφαιρέσει τις λέξεις (“University”, “of”, “Institute”), ταξινομεί τους χαρακτήρες του. Η συνάρτηση hashed παίρνει το κείμενο με τους ταξινομημένους χαρακτήρες και επιστρέφει την χασαρισμένη τιμή του.

Κύριες συναρτήσεις

Δημιουργία Chord

Η δημιουργία του πρωτοκόλλου Chord και η προσθήκη κόμβων σε αυτό γίνεται με την συνάρτηση **join()**. Αρχικά, ελέγχει αν ο κόμβος που εισάγεται είναι ο πρώτος κόμβος. Αν είναι, ρυθμίζει τα την μεταβλητή prev και τις τιμές του fingerTable κατάλληλα. Αλλιώς, καλείται η lookupNode(), η οποία επιστρέφει τον κόμβο (afterNode) πριν από τον οποίο θα εισαχθεί ο νέος. Τότε ενημερώνονται κατάλληλα οι μεταβλητές prev, καθώς και τα fingerTables των δύο κόμβων. Έπειτα εκτελείται η

μεραφορά δεδομένων από τον afterNode πριν τον νέο. Συγκεκριμένα, συγκρίνονται τα κλειδιά (keys) των δεδομένων και βλέποντας με την συνάρτηση distance() αν κατατάσσονται στον afterNode ή τον newNode. Τα δεδομένα προς μετακίνηση αντιγράφονται απευθείας στο λεξικό data του newNode, και σημειώνονται στην λίστα to_delete. Στο τέλος της αντιγραφής των δεδομένων, διατρέχουμε αυτήν την λίστα και διαγράφουμε τα δεδομένα που μετακινήθηκαν από τον afterNode, έτσι ώστε να μην υπάρχουν διπλότυπα. Στο τέλος της διαδικασίας καλούνται οι **updateFIngerTable()** για τον νέο κόμβο, καθώς και η **stabilization()**, η οποία ενημερώνει τα FingerTables όλων των κόμβων.

Διαγραφή κόμβου

Εξίσου σημαντική λειτουργία αποτελεί και η διαγραφή των κόμβων, η οποία υλοποιείται με την συνάρτηση **delete()**, και καλείται από τον κόμβο προς διαγραφή. Στην περίπτωση που διαπιστωθεί ότι ο κόμβος προς διαγραφή είναι ο μοναδικός κόμβος του Chord (ο προηγούμενος και ο επόμενος κόμβος είναι ο εαυτός του), τότε απλά διαγράφουμε τα δεδομένα του. Σε άλλη περίπτωση ακολουθούμε την εξής διαδικασία: Ενημερώνουμε τον προηγούμενο και τον επόμενο κόμβο του κατάλληλα, έτσι ώστε να είναι διαδοχικοί. Έπειτα, σειρά έχει η μετακίνηση των δεδομένων. Εφόσον διαγράφουμε τον κόμβο, όλα του τα δεδομένα θα πρέπει να αποθηκευτούν στον επόμενο κόμβο, οπότε εκτελούμε την μεταφορά και έπειτα διαγράφουμε το λεξικό του κόμβου που θέλουμε να διαγράψουμε. Τελευταίο βήμα για την διαγραφή κόμβου αποτελεί η διαγραφή του κόμβου από τα fingerTables των υπολοίπων. Αυτό το επιτυγχάνουμε με μια σειρά από κλήσεις της stabilization() και της updateFingerTable(), έτσι ώστε να ενημερωθούν όλοι οι πίνακες και να αλλάξουν τα δεδομένα τους αν χρειάζεται.

Αναζήτηση κόμβου

Η συνάρτηση **lookupNode(id)** χρησιμοποιείται έτσι ώστε να βρεθεί ο κόμβος του Chord στον οποίο κατατάσσεται το id. Σε περίπτωση που η απόσταση του id από τον τωρινό κόμβο είναι μικρότερη από την απόσταση από τον επόμενο κόμβο, καταλαβαίνουμε ότι βρίσκεται ενδιάμεσα του τωρινού κόμβου και του επόμενου, και έτσι επιστρέφεται ο επόμενος. Αλλιώς, καταλαβαίνουμε ότι βρίσκεται στο άλλο τόξο και πλέον πρέπει να ελεγχθούν και οι υπόλοιποι κόμβοι. Στην συνέχεια αρχικοποιούμε την μεταβλητή nextNode με το τελευταίο στοιχείο του fingerTable του τρέχοντος κόμβου. Εάν δεν βρεθεί κόμβος κοντινότερος στο id, αυτό είναι που επιστρέφεται και καλείται για νέο έλεγχο με την lookupNode. Αν κάποια στιγμή το id βρεθεί ίσο με το id του τρέχοντος κόμβου, επιστρέφεται ο τρέχων κόμβος.

Σταθεροποίηση Chord

Τέλος, έχουμε την συνάρτηση **stabilization()**, η οποία κρατάει ενημερωμένους τους πίνακες fingerTables όλων των κόμβων. Αυτό υλοποιείται καλώντας την updateFingerTable() για τον τρέχων κόμβο, και στην συνέχεια για τους επόμενους με σειρά. Η stabilization() καλείται μετά από κάθε προσθήκη και αφαίρεση κόμβου. Επίσης, θα πρέπει να καλείται και σε τακτά χρονικά διαστήματα για να εξασφαλίζεται η σωστή δομή του Chord.

Chord: data functions

Εισαγωγή Δεδομένων

Με την συνάρτηση **insert_data(row)** παίρνουμε μια γραμμή από το σετ δεδομένων μας και τα εισάγουμε στο chord μέσω ενός κόμβου. Τα δεδομένα αρχικά χωρίζονται σε όνομα, πανεπιστήμια και βραβεία. Διατρέχοντας την λίστα με τα πανεπιστήμια, χρησιμοποιούμε την συνάρτηση hashed του αρχείου hashing για να βρούμε το hashId, με το οποίο θα προσδιορίσουμε τον κόμβο στον οποίο θα αποθηκευτεί η πληροφορία. Σημειώνουμε ότι τα λεξικά των κόμβων έχουν ως κλειδιά τα μοναδικά πανεπιστήμια. Βρίσκουμε τον κόμβο στον οποίο θα αποθηκεύεται το συγκεκριμένο πανεπιστήμιο και ανάλογα με το αν υπάρχει ήδη ή όχι ενημερώνουμε / κάνουμε νέα εγγραφή στο λεξικό με το καινούριο όνομα και τα αντίστοιχα βραβεία.

Αναζήτηση Δεδομένων

Έχουν υλοποιηθεί δύο συναρτήσεις αναζήτησης, η μια για αναζήτηση στο πεδίο του alma mater, και η άλλη για αναζήτηση με βάση το όνομα.

Η συνάρτηση **search_education()** παίρνει ορίσματα το όνομα του πανεπιστημίου που ψάχνουμε, καθώς και έναν ελάχιστο αριθμό βραβείων. Το όνομα του πανεπιστημίου περνάει από την συνάρτηση hashed και παίρνουμε την μεταβλητή hashed_education, την οποία θα χρησιμοποιήσουμε στην συνάρτηση lookupNode() για να βρούμε σε ποιόν κόμβο θα ανήκει αυτό το πανεπιστήμιο (αν υπάρχει). Διατρέχουμε το λεξικό αυτού του κόμβου, ψάχνοντας αν το όνομα του πανεπιστημίου υπάρχει στις εγγραφές του. Εάν ναι, προσθέτουμε τα ονόματα των επιστημόνων που έχουν πάνω από τον ελάχιστο αριθμό βραβείων.

Η συνάρτηση **search_scientist()** παίρνει ως όρισμα το επώνυμο (προαιρετικά και το όνομα) του επιστήμονα που ψάχνουμε. Εφόσον έχουμε δομήσει τα δεδομένα μας με βάση το alma mater, στην αναζήτηση με βάση το όνομα θα πρέπει να διατρέξουμε το σύνολο των κόμβων έτσι ώστε να βρούμε όλα τα πανεπιστήμια στα οποία φοίτησε ο συγκεκριμένος επιστήμονας. Για να το κάνουμε αυτό, αρχικά ψάχνουμε στο λεξικό του τρέχοντος κόμβου, ελέγχοντας εάν υπάρχει το επώνυμο (ή και το όνομα) σε κάποιο κλειδί του λεξικού. Εάν ναι, προσθέτουμε στο λεξικό αποτελεσμάτων το κλειδί με τιμές alma mater και awards κατάλληλα. Στην συνέχεια, διατρέχουμε τους κόμβους διαδοχικά, ακολουθώντας την ίδια διαδικασία με πριν. Μόλις ολοκληρωθεί αυτή, στο λεξικό αποτελεσμάτων θα έχουμε ως εγγραφές τους επιστήμονες με το συγκεκριμένο επώνυμο (- όνομα), καθώς και πληροφορίες για τις σπουδές και τα βραβεία τους.

Chord: visualization

Η συνάρτηση οπτικοποίησης του Chord προσαρμόστηκε με βάση το documentation του Holoviews και βρίσκεται εδώ: [Chord — HoloViews v1.18.1]

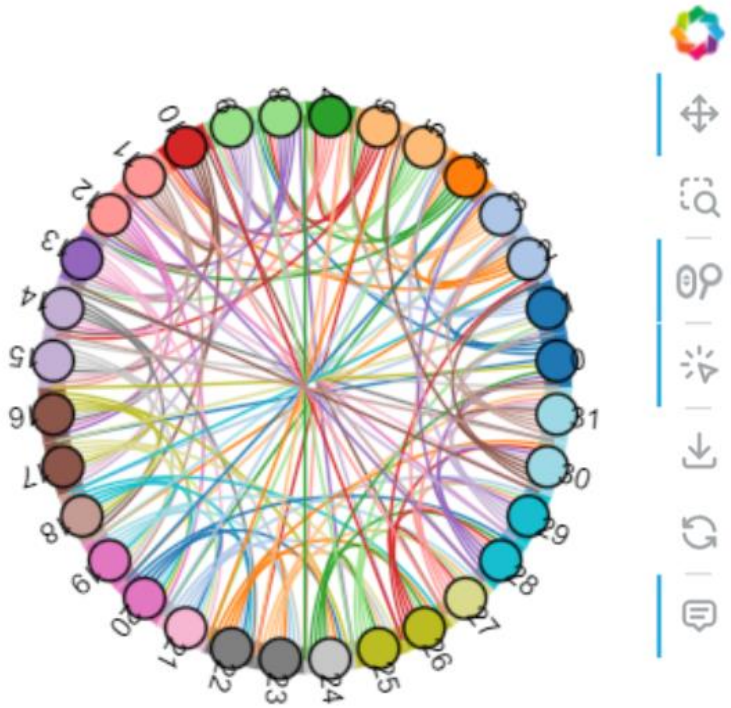


Figure 1- Chord Ring with 32 Nodes

Παραδείγματα

(Αρχείο test_chord.py)

Παρακάτω θα δημιουργούμε Chord με 16 κόμβους.

Αφού ορίσουμε κατάλληλο K έτσι ώστε $2^K = SIZE$, δημιουργείται το Chord με την εισαγωγή του πρώτου κόμβου και την μεταφορά των δεδομένων σε αυτόν. Έπειτα με ένα for loop προχωράμε στην εισαγωγή και των υπόλοιπων κόμβων.

```
# First node joining
nodes[0].join(nodes[0])

# Data into Chord
with open("data.csv", 'r') as file:
    for row in file:
        nodes[0].insert_data(row.strip())

# Other nodes joining - data is being distributed
for i in range(1, SIZE):
    nodes[0].join(nodes[i])

nodes[0].visualize_chord_ring()
```

Καλώντας την συνάρτηση visualize_chord_ring() έχουμε:

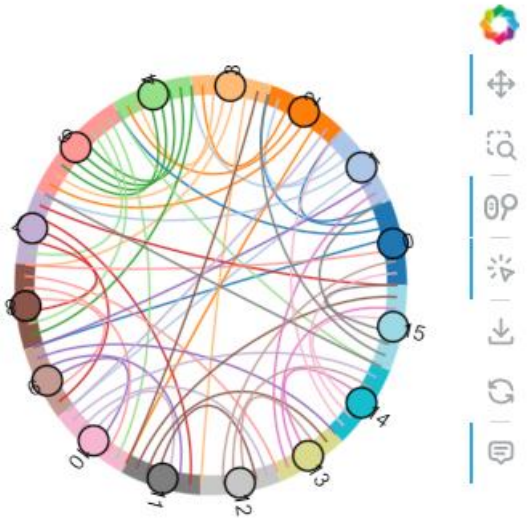
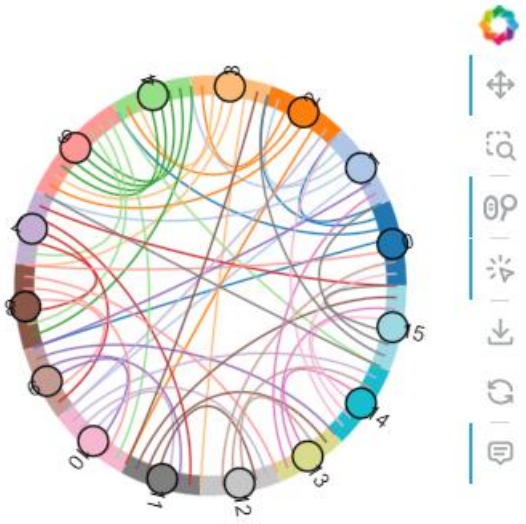
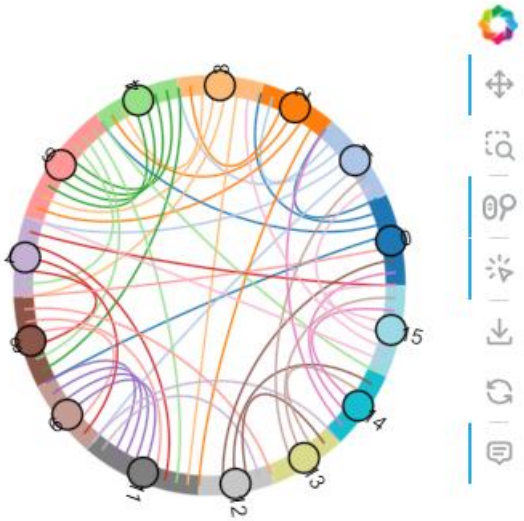


Figure 2- Chord Ring with 16 Nodes

Έπειτα διαγράφουμε τον κόμβο με id = 5 (`nodes[5].delete()`) και καλούμε πάλι την visualize_chord_ring():

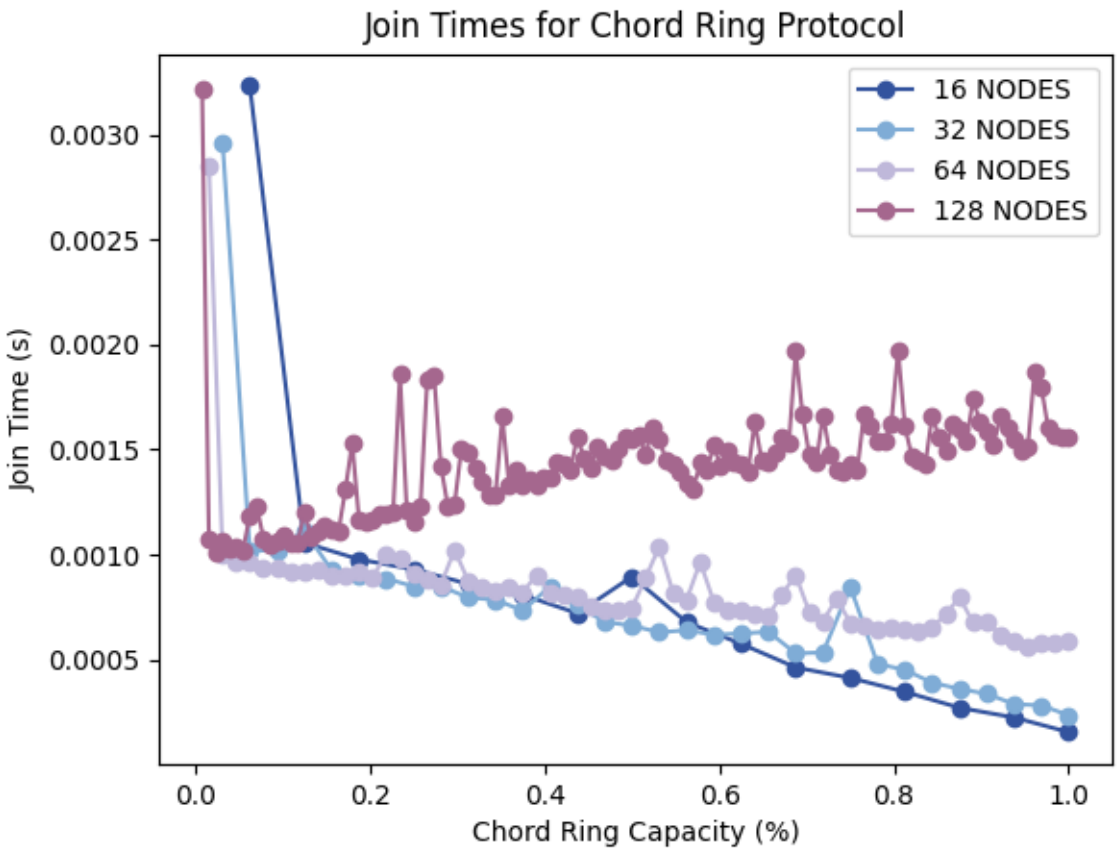


Όπως βλέπουμε ο κόμβος 5 διαγράφτηκε και οι ακμές του γραφήματος άλλαξαν αναλόγως. Διαγράφοντας ακόμη έναν κόμβο (τον 10), μένουμε με το παρακάτω σχήμα:



+ times in diff chords

ΧΡΟΝΟΙ ΑΠΟΔΟΣΗΣ ΓΙΑ ΕΙΣΑΓΩΓΗ ΚΟΜΒΟΥ



Παραπάνω βλέπουμε τους χρόνους εισαγωγής κόμβων για Chord 4 διαφόρων μεγεθών: 16, 32, 64 και 128. Παρατηρούμε ότι και στις 4 περιπτώσεις η εισαγωγή του πρώτου κόμβου παίρνει τον μεγαλύτερο χρόνο σε σχέση με τους επόμενους. Αυτό γίνεται γιατί κατά την δημιουργία της δομής περνάνε όλα τα δεδομένα μέσα στον πρώτο κόμβο, ενώ κατά την εισαγωγή κάποιου άλλου κόμβου μόνο ένα μέρος τον δεδομένων μετακινείται προς αυτόν. Σε γενικές γραμμές παρατηρούμε ότι οι χρόνοι εισαγωγής κόμβου είναι ανάλογοι με το μέγεθος της δομής στην οποία εισάγονται. Αυτό οφείλεται στο γεγονός ότι μέσα στην συνάρτηση join() χρησιμοποιούμε την συνάρτηση lookupNode(), η οποία όταν ψάχνει κόμβους σε μεγαλύτερη δομή κάνει και περισσότερο χρόνο. Το βλέπουμε στον παρακάτω πίνακα:

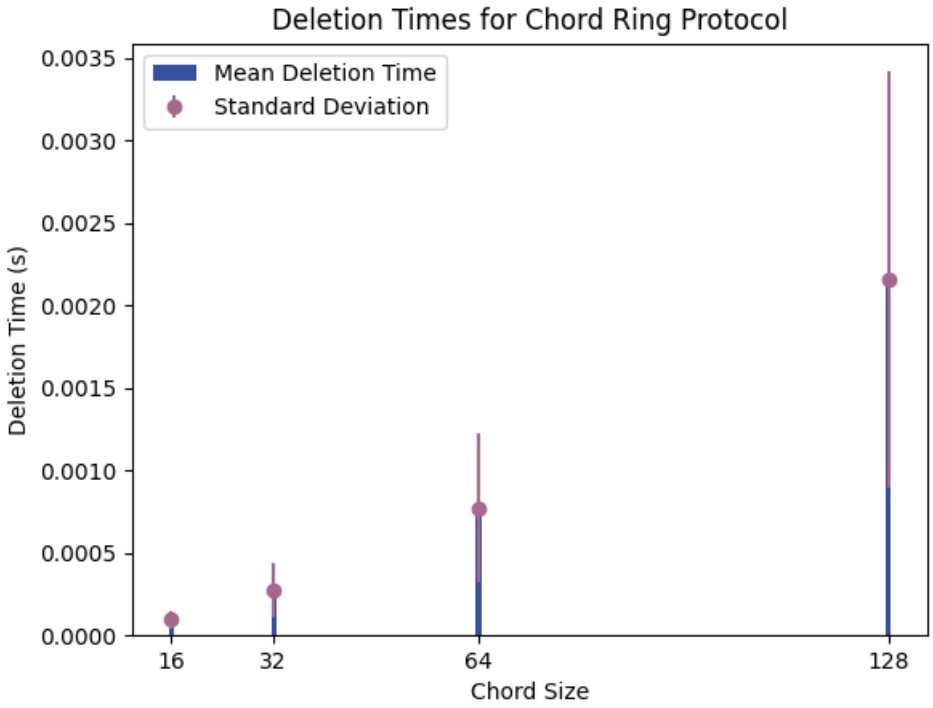
LOOK UP TIMES (s)	**16**	**32**	**64**	**128**
Mean Average	$1.51132\ x\ 10^{-6}$	$1.98554\ x\ 10^{-6}$	$2.73789\ x\ 10^{-6}$	$3.62709\ x\ 10^{-6}$
Standard Deviation	$6.76719\ x\ 10^{-7}$	$9.61307\ x\ 10^{-7}$	$3.84687\ x\ 10^{-6}$	$3.85466\ x\ 10^{-6}$

Έχουμε τον Μέσο Όρο και την Τυπική Απόκλιση των χρόνων lookup στις διαφορετικού μεγέθους δομές Chord. Ο Μέσος Όρος αυξάνεται όσο μεγαλώνει το μέγεθος του Chord, όπως άλλωστε και η Τυπική Απόκλιση. Η Τυπική Απόκλιση αντιπροσωπεύει την μεταβλητότητα στους χρόνους

αναζήτησης, και έτσι το Chord-16 έχει σχετικά συνεκτικούς χρόνους αναζήτησης, ενώ το Chord-128 έχει μεγάλη μεταβλητότητα σε αυτούς.

DELETION TIMES

Ένας ακόμη παράγοντας σύγκρισης είναι οι χρόνοι διαγραφής των κόμβων στα διαφορετικά Chord Rings. Έχουμε:



Παρατηρούμε ότι όσο μεγαλώνει το μέγεθος του Chord, ο μέσος όρος σφάλματος καθώς και η τυπική απόκλιση αυξάνονται. Αυτό συμβαίνει διότι κάθε φορά που διαγράφεται ένας κόμβος, πρέπει να ενημερωθούν οι fingerTables $\log(SIZE)$ κόμβων.

SEARCH TIMES

Όπως αναφέραμε πριν, έχουμε δύο συναρτήσεις αναζήτησης. Στην πρώτη κάνουμε αναζήτηση με βάση το εκπαιδευτικό ίδρυμα και στην δεύτερη με βάση το ονοματεπώνυμο του επιστήμονα.

ALMA MATER

Εκτελώντας τον κώδικα: `result = nodes[1].search_education("Massachusetts Institute of Technology", 2)` βρίσκουμε τους επιστήμονες που σπούδασαν στο συγκεκριμένο πανεπιστήμιο, οι οποίοι είναι οι εξής:

```
['Lenore Blum', 'Fernando J. CorbatГ³', 'Peter J. Denning', 'Whitfield Diffie', 'Charles
Stark Draper', 'Lance Fortnow', 'Zoubin Ghahramani', 'Johan HГ¥stad', 'Danny Hillis', 'Ma
nolis Kellis', 'Leslie Lamport', 'Tom M. Mitchell', 'Paul Mockapetris', 'Brad A. Myers',
'Andrew Ng', 'Claude Shannon', 'Thomas Sterling', 'Ivan Sutherland', 'David Waltz', 'Lotf
i Zadeh']
```

Η παράμετρος “Massachusetts Institute of Technology” χασάρεται και έπειτα βρίσκουμε τον κόμβο στον οποίο αντιστοιχεί. Αν το πανεπιστήμιο υπάρχε σε αυτόν τον κόμβο, τότε επιστρέφουμε όλους τους επιστήμονες που σπούδασαν σε αυτό και έχουν πάνω από 2 βραβεία. Εάν δεν ορίσουμε αριθμό βραβείων, τότε θεωρείται ότι είναι 0.

Για την ίδια αναζήτηση έχουμε τους εξής χρόνους:

	16	32	64	128
SEARCH TIME	$3.70999\ x\ 10^{-5}$	$3.89000\ x\ 10^{-5}$	$3.91000\ x\ 10^{-5}$	$4.12999\ x\ 10^{-5}$

Οι διαφορές στους χρόνους μεταξύ των Chord διαφορετικών μεγεθών είναι σχετικά μικρές, αλλά βλέπουμε πως και σε αυτήν την περίπτωση πάνε ανάλογα με το μέγεθος.

Η αναζήτηση έβγαλε τα ίδια αποτελέσματα και στις 4 περιπτώσεις.

SURNAME AND NAME

Εκτελώντας τον κώδικα: `results= nodes[1].search_scientist("Blum")` βρίσκουμε τους επιστήμονες με επώνυμο Blum, οι οποίοι είναι οι εξής:

```
Name:  Lenore Blum
Alma Mater:  ['Simmons College', 'Massachusetts Institute of Technology']
Awards:  4

Name:  Manuel Blum
Alma Mater:  ['Massachusetts Institute of Technology']
Awards:  1

Name:  Dorothy Blum
Alma Mater:  ['Brooklyn College']
Awards:  0
```

Επιστρέφονται όλοι οι επιστήμονες με το συγκεκριμένο επίθετο, καθώς και τα πανεπιστήμια που σπούδασε ο καθένας μαζί με τα βραβεία.

Εκτελώντας τώρα την εντολή `results= nodes[1].search_scientist("Blum", "Lenore")`

```
Name:  Lenore Blum
Alma Mater:  ['Simmons College', 'Massachusetts Institute of Technology']
Awards:  4
```

Βλέπουμε ότι επιστρέφεται μόνο ένας επιστήμονας που ταυτίζεται το όνομά του με τις παραμέτρους που έχουμε βάλει στην συνάρτηση.

Παρακάτω βλέπουμε τους χρόνους για αυτήν την αναζήτηση:

	16	32	64	128
SEARCH TIME	$8.30000\ x\ 10^{-5}$	$8.560002\ x\ 10^{-5}$	$8.33000\ x\ 10^{-5}$	$8.52999\ x\ 10^{-5}$

Οι χρόνοι μεταξύ τους έχουν μικρές διαφορές. Σε σχέση με τους χρόνους της προηγούμενης αναζήτησης είναι αρκετά αυξημένοι, αφού η συγκεκριμένη αναζήτηση χρειάζεται να διατρέξει όλους τους κόμβους του Chord Ring.