Πολυδιάστατες Δομές Δεδομένων και Υπολογιστική Γεωμετρία

Project-2: DHTs (Chord)

Μέλη ομάδας:

ΑΓΓΕΛΆΚΗ ΦΩΤΕΙΝΗ ΑΜ: 1067540

ΠΕΤΡΟΠΟΥΛΟΣ ΑΝΤΩΝΙΟΣ ΑΜ: 1043812

ΠΡΆΠΠΑΣ ΤΡΙΑΝΤΆΦΥΛΛΟΣ ΑΜ: 1067504

ΦΩΤΕΙΝΌΣ ΕΜΜΑΝΟΥΉΛ ΑΜ: 1067428

Εισαγωγή:

Η συλλογή δεδομένων έγινε με την χρήση της ιστοσελίδας https://openflights.org/data.html. Μετά την συλλογή τους, έγινε η κατάλληλη επεξεργασία για την αποθήκευση τους ως παράδειγμα στην δομή που δημιουργήσαμε.

Το Chord είναι ένα πρωτόκολλο και ένας αλγόριθμος για έναν κατανεμημένο πίνακα κατακερματισμού peer-to-peer. Ένας κατανεμημένος πίνακας κατακερματισμού αποθηκεύει ζεύγη κλειδιού-τιμής εκχωρώντας κλειδιά σε διαφορετικούς υπολογιστές. Ένας κόμβος θα αποθηκεύσει τις τιμές για όλα τα κλειδιά για τα οποία είναι υπεύθυνος. Το Chord καθορίζει πώς εκχωρούνται τα κλειδιά σε κόμβους και πώς ένας κόμβος μπορεί να ανακαλύψει την τιμή για ένα δεδομένο κλειδί, εντοπίζοντας πρώτα τον κόμβο που είναι υπεύθυνος για αυτό το κλειδί.

Γενική επεξήγηση λειτουργίας κώδικα:

Ο κώδικας ξεκινά τρέχοντας το αρχείο controller, οπου ελέγχει την ροή του προγράμματος. Το αρχείο controller ζητά το input του χρήστη για το ορισμό των παραμέτρων και μολις πληκρολογηθούν, δημιουργεί ένα thread για κάθε node, το οποίο θα είναι υπεύθυνο για το node. Στην συνέχεια το κάθε thread δημιουργεί το node του και ξεκινά να το προσομοιώνει με την join. Αφού κάνει join, το κάθε node μπορεί να αναλαμβάνει requests και να στέλνει μηνύματα μέσω priority queue. Μόλις ολοκληρωθούν όλα τα join, επιτρέπεται η ενημέρωση του finger table μεγέθους m και στην συνέχεια της λίστας διαδόχων μεγέθους r. Μόλις ολοκληρωθούν το

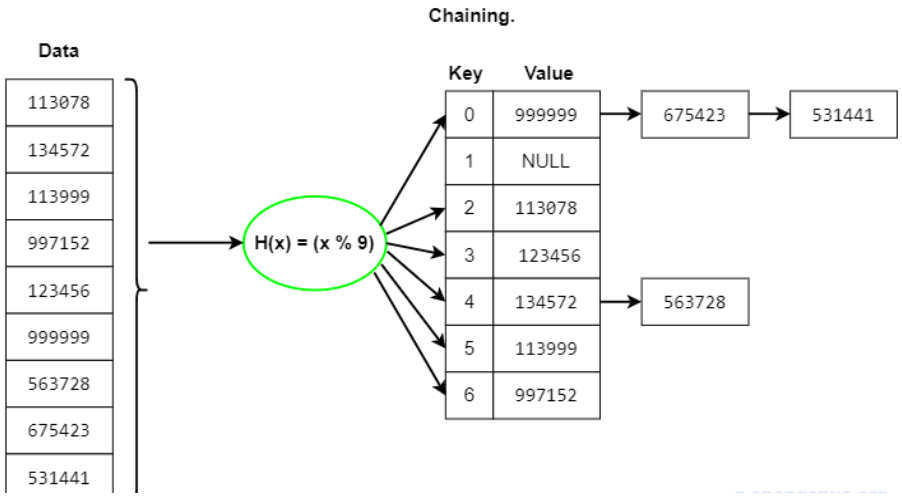
controller παγώνει την προσομοίωση του chord και εμφανίζει στον χρήστη τι επιλογές έχει, οι οποίες είναι: Insert key, delete key, lookup key, update key, add node, remove node, massive node failure, continue simulation, quit.

Αρχείο parseTesting:

Με το αρχείο **parseTesting** επεξεργαζόμαστε τα δεδομένα που συλλέξαμε από τον παραπάνω σύνδεσμο. Αρχικά χρησιμοποιούμε το function **GetHashValue(value)** το οποίο κάνει hash την τιμή που θέτουμε στο value με την εντολή hash() που παρέχει η python. Στην περίπτωση μας σαν value θεωρούμε το όνομα του κάθε αεροδρομίου, όπως στο παρακάτω παράδειγμα.

```
name of airport: Húsavík Airport name of airport hashed: -941870247227004844
```

Το **GetHashKey(value, divider)** του οποίου η λειτουργία εξηγείται με τη βοήθεια της παρακάτω εικόνας:



Λόγω της ύπαρξης πολλών hash values, για τον καλύτερο διαμοιρασμό τους, διαιρούνται με έναν αριθμό και ανάλογα με το υπόλοιπο που αφήνουν τους ορίζουμε ένα κλειδί.

Επομένως πολλά hash values αντιστοιχούν σε έναν hash-key, τα οποία στην συνέχεια εισάγονται στα κατάλληλα nodes.

Στη συνέχεια συναντάμε την **data_hash(data, divider)**, ο οποία απλώς κάνει hash μια γραμμή δεδομένων. Χρησιμοποιείται στην περίπτωση που κάνει input ο χρήστης μία γραμμή δεδομένων.

Τέλος η **get_data(start, stop, divider)** χρησιμοποιείται στα άλλα υπόλοιπα αρχεία με σκοπό να εισάγουμε τα δεδομένα στα nodes που αντιστοιχούν. Στην μέθοδο αυτή «διαβάζουμε» το επιλεγμένο αρχείο και το χωρίζουμε με βάση τις γραμμές του, και στην συνέχεια δημιουργούμε ένα tuple (μια πλειάδα) το οποίο περιέχει το hash-key που παίρνει η γραμμή, το hash-value του ονόματος του αεροδρομίου και την γραμμή σαν data. Η μέθοδος αυτή έχει σαν παραμέτρους τα start, end τα οποία ορίζουν τη γραμμή που θέλουμε να αρχίσει και να τελειώσει το parse του αρχείου αντίστοιχα και το divider που χρησιμοποιείται για να καθορίσουμε τα πόσα hash-keys θέλουμε να έχει.

Αρχείο parameters:

Στο αρχείο parameters έχουμε αρχικοποίηση και δήλωση των μεταβλητών:

```python
import threading
import queue
used_ids = []  # To store ids in use
thread_list = []  # To store all active threads
node_list = []  # To store all active nodes
number_of_nodes = 8  # Maximum number of nodes
id_limit = 63  # Maximum value of id that a node can have
m = 5  # Number of fingers
r = 3  # Number of successors
queues = {}  # Global queue list, stores the queues of all reachable nodes

enabled = threading.Event()  # If enabled, thread loops

join_lock = threading.Lock()  # Used so that nodes join one by one
update_fingers = threading.Event()  # Used to allow or forbid finger updates
update_successors = threading.Event()  # Used to allow or forbid successor list updates
freeze = threading.Event()  # Used to temporarily stop threads/nodes
query_time = queue.Queue()
```

Αρχείο thread_script:

Στο thread_script ορίζεται η κλάση Node με τα εξής attributes:

```python
def __init__(self):

    self.id = None  # Stores the id of the node
    self.predecessor = None  # Stores the predecessor of the node
    self.successor = None  # Stores the successor of the node
    self.finger_table = [None] * m  # Stores the finger table of the node
    self.queue = queue.PriorityQueue()  # Stores the priority queue of the node, which is where it receives messages
    self.stored_data = []   # Stores the data of the node
    self.successor_list = [None] * r   # Stores the next r successors of the node
```

Η πρώτη συνάρτηση που συναντάται είναι η **join_network**. Η οποία χρησιμοποιείται για την είσοδο ενός νέου node στο δίκτυο. Συγκεκριμένα ,γίνεται χρήση ενός lock, το οποίο μας διαβεβαιώνει ότι θα γίνει μόνο ένα join κάθε φορά. Στη συνέχεια επιλέγουμε ένα τυχαίο id και αν έχει χρησιμοποιηθεί ήδη συνεχίζουμε την αναζήτηση έως ότου βρεθεί κάποιο διαθέσιμο. Μόλις βρεθεί, προσθέτει το id αυτό σε μία λίστα και θέτει στο κάθε ένα τον successor και predecessor.

Η συνάρτηση **check_predecessor** ελέγχει αν ο κόμβος έχει σωστό predecessor: αν δεν έχει τότε θέτει τον εαυτό του, αλλιώς αν το id είναι ανάμεσα στο τρέχων node και στον predecessor του, τότε στέλνει στο παλιό predecessor να κάνει stabilize και αντικαθιστά το παλιό predecessor με το καινούργιο.

Με την **stabilize**, το τρέχων node ζητάει τον predecessor του successor του: αν ο predecessor του successor είναι το σωστό successor για το node μας τότε γίνεται αντικατάσταση του παλιού successor με τον καινούργιο.

Με την συνάρτηση **find_data_range**, εντοπίζεται ανάμεσα σε ποιά δύο nodes βρίσκεται το key των data.

Με την συνάρτηση **find_data_owner**, αν το key είναι κοντινότερα στο predecessor ή στο successor του και αποστέλλεται στον κοντινότερο μήνυμα ώστε ο ίδιος να αναλάβει την επεξεργασία των δεδομένων του.

Η **manipulate_data** περιλαμβάνει τις λειτουργίες: insert, update, lookup, delete, για εισαγωγή, ενημέρωση, αναζήτηση και διαγραφή δεδομένων αντίστοιχα.

Η συνάρτηση **update_fingers** αναζητά τα fingers του κάθε node και συμπληρώνει το finger table.

Η συνάρτηση **find_successor**, δοθέντος ενός id, ελέγχει αν το συγκεκριμένο id βρίσκεται ανάμεσα στο id του τρέχοντος node

και του επόμενου. Αν όχι, τότε περνά μήνυμα στο επόμενο κατα σειρά node να εκτελέσει την ίδια διαδικασία.

Η συνάρτηση **update_successors** αναζητά τους r επόμενους successors του κάθε node και συμπληρώνει τον πίνακα successor_list.

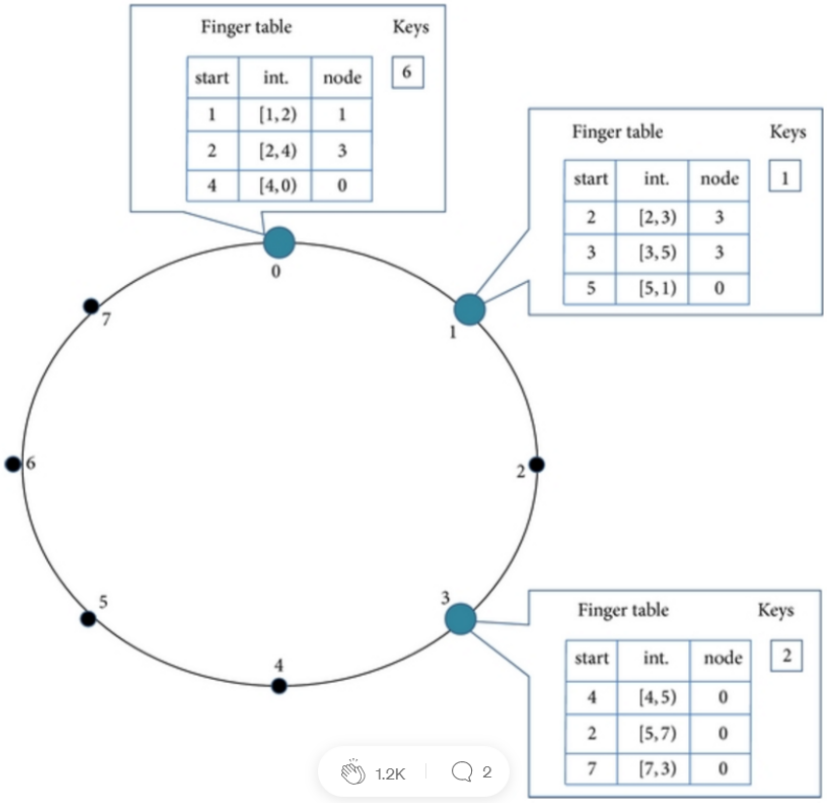
Η συνάρτηση **check_if_alive** κάνει ping το ζητούμενο node ώστε να δει αν είναι ενεργό ή όχι, χρησιμοποιώντας την ping και αναμένοντας απάντηση για ένα περιορισμένο χρονικό διάστημα. Αν το node δεν απαντήσει μέσα σε αυτό το διάστημα, θεωρείται ανενεργό

Η συνάρτηση **ping** στέλνει μήνυμα στο ζητούμενο node και ζητά ανταπόκριση

Η **node_remove** στέλνει request για αφαίρεση του node με βάση το id και εμφανίζονται τα αντίστοιχα μηνύματα.

Η **data_id_check** ελέγχει αν ένα key είναι πιο κοντά στον predecessor του ή στον successor του.

Η **id_check** ελέγχει αν μια συγκεκριμένη id είναι ανάμεσα σε δύο άλλες.



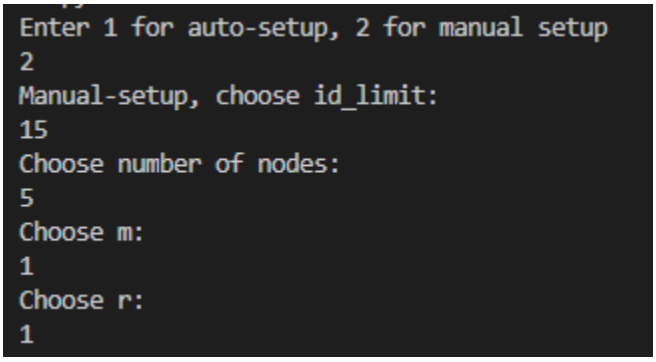
Εκτέλεση controller:

Το αρχείο controller περιέχει το interface του χρήστη.

Η επιλογή του auto-setup δημιουργεί μια δομή με 8 κόμβους, όριο 63 ids, 5 fingers ανά node και λίστα successors μεγέθους 3.

```
Enter 1 for auto-setup, 2 for manual setup
1
Auto-setup with 8 nodes, id limit: 63 and m: 5
Setting up...
```

Διαφορετικά ο χρήστης έχει την επιλογή για manual-setup, όπου μπορεί να διαλέξει ο ίδιος τις παραπάνω τιμές και επιπροσθέτως να ορίσει και την τιμή r, η οποία ορίζει το πλήθος των επόμενων node ids που γνωρίζει ο κάθε κόμβος.



Αφού γίνει επιλογή του κατάλληλου setup, δημιουργείται ένα dictionary για τα queues που θα χρησιμοποιηθούν και αποτρέπεται η ενημέρωση των fingers και των successors, μέχρι να προστεθούν όλα τα nodes. Στη συνέχεια, δημιουργούμε ένα thread για κάθε ένα node του δακτυλίου και αφού κάνουν join όλα τα nodes, επιτρέπουμε εκ νέου την ενημέρωση των fingers και αναμένουμε την ενημέρωσή τους. Μόλις ολοκληρωθεί η ενημέρωση, επιτρέπουμε την ενημέρωση της λίστας successors και αφού ολοκληρωθεί και αυτή, εισάγει μερικά δεδομένα στα nodes και εμφανίζει στον χρήστη τις επιλογές του:

Εκτέλεση ενός join:

```
18 gets lock

I am 42 checking successor for 18
Successor found: 42
18 releases lock, join time: 4.004052400588989
```

Μετά την εκτέλεση όλων των join:

```
Total join time: 45.23557782173157
```

Εκτέλεση ενός update finger table:

```
Finding finger: 9
I am 32 checking successor for 9
I am 56 checking successor for 9
Passing find successor to node: 61
I am 61 checking successor for 9
Successor found: 32
Finger Table of node: 56 with time: 6.527002811431885
[61, 61, 61, 32, 32]
```

Μετά την εκτέλεση όλων των update finger table:

```
Total finger table update time: 69.76281523704529
```

Εκτέλεση ενός update successor list:

```
Finding successor for successor list of node: 37 for index 0I am 39 checking successor for 39

Successor found: 41
Finding successor for successor list of node: 37 for index 1
I am 41 checking successor for 41
Successor found: 48
Finding successor for successor list of node: 37 for index 2I am 48 checking successor for 48

Successor found: 56
Successor list of node: 37 with time: 1.3239984512329102
```

Μετά την εκτέλεση όλων των successor update list:

```
Total successor list update time: 18.514747142791748
```

Εισαγωγή δεδομένων:

```
Data: (3, 8159142398154472387, '9,"Kangerlussuaq Airport","Sondrestrom","Greenland","SFJ","BGSF",67.0122218992
```

->συνέχεια γραμμής:

```
-50.7116031647,165,-3,"E","America/Godthab","airport","OurAirports"') inserted in node 61
```

Εμφάνιση επιλογών χρήστη:

```
0) Get node info

1) Insert data

2) Delete data

3) Update data

4) Exact match data (lookup)

5) Add nodes

6) Remove node

7) Massive node failure

8) Continue simulation for x seconds

9) End simulation
```

Επιλογή get node info:

```
0
Active nodes: [39, 56, 61, 34, 37, 41, 48, 32]
Pre: 37 Me: 39 Suc: 41 F_Table: [41, 48, 48, 48, 56] Successor list: [48, 56, 61]
Node 39 data:


Pre: 48 Me: 56 Suc: 61 F_Table: [61, 61, 61, 32, 32] Successor list: [32, 34, 37]
Node 56 data:
[(49, 5146530585736995057, '1,"Goroka Airport","Goroka","Papua New Guinea","GKA","AYGA",-6.081689834590001,145.
```

(πιο κάτω από ότι εμφανίζεται στην φωτογραφία περιλαμβάνονται οι πληροφορίες για τα υπόλοιπα node)

Επιλογή Insert data:

```
1
Give data string to insert
11,"Akureyri Airport","Akureyri","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atlantic/Reykjavik","airport","OurAirports"
('find_data_range', (20, 5941638377847593084, '11,"Akureyri Airport","Akureyri","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atl
I am 56 checking successor for 20
I am 61 checking successor for 20
Successor found: 32
Data: (20, 5941638377847593084, '11,"Akureyri Airport","Akureyri","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atlantic/Reykjavi
Time taken until completion: 9.555763483047485
```

Επιλογή Delete data:

```
2
Give key string to delete
"Akureyri Airport"
('find_data_range', (20, 5941638377847593084), 'delete')
I am 39 checking successor for 20
I am 56 checking successor for 20
I am 61 checking successor for 20
Successor found: 32
Data: (20, 5941638377847593084, '11,"Akureyri Airport","Τεστ για update","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atlantic/Reykj
Time taken until completion: 2.0736780166625977
```

Επιλογή Update data:

```
3
Give data string to update
11,"Akureyri Airport","Τεστ για update","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atlantic/Reykjavik","airport","OurAirports"
('find_data_range', (20, 5941638377847593084, '11,"Akureyri Airport","Τεστ για update","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","A
I am 34 checking successor for 20
I am 56 checking successor for 20
I am 61 checking successor for 20
Successor found: 32
Data: (20, 5941638377847593084, '11,"Akureyri Airport","Τεστ για update","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atlantic/Reykja
Time taken until completion: 3.5967419147491455
```

Επιλογή Exact match data:

```
4
Give key string to match
"Akureyri Airport"
('find_data_range', (20, 5941638377847593084), 'lookup')
I am 48 checking successor for 20
I am 61 checking successor for 20
Successor found: 32
Data: (20, 5941638377847593084, '11,"Akureyri Airport","Τεστ για update","Iceland","AEY","BIAR",65.66000366210938,-18.07270050048828,6,0,"N","Atlantic/Reykj
Time taken until completion: 11.032633781433105
```

Επιλογή remove node:

```
6
Current nodes:[39, 56, 61, 34, 37, 41, 48, 32]
Choose which node to remove:
56
Node 56 was removed!
Pre: 48 Me: 56 Suc: 61 F_Table: [61, 61, 61, 32, 32]Time taken until completion: 6.260105133056641
```

Επιλογή massive node failure:

```
7
Current nodes:None
Choose how many nodes to remove:
4
Node 48 was removed!
Pre: 41 Me: 48 Suc: 56 F_Table: [56, 56, 56, 61, 32]
```

(εδώ κάνει fail και δεν συνεχίζει)

Επιλογή continue simulation:

```
8
Continue simulation for how many seconds:
10
The simulation has been frozen.
```

(αφού περάσουν 10 δευτερόλεπτα κάνει πάλι freeze)

Επιλογή quit:

```
9) End simulation

9
Pre: 56 Me: 61 Suc: 32 F_Table: [32, 32, 32, 32, 32]Pre: 61 Me: 32 Suc: 34 F_Table: [34, 37, 37, 41, 56]
Pre: 34 Me: 37 Suc: 39 F_Table: [39, 41, 48, 48, 56]

Pre: 37 Me: 39 Suc: 41 F_Table: [41, 48, 48, 48, 56]
Pre: 39 Me: 41 Suc: 48 F_Table: [48, 48, 48, 56, 61]
Pre: 32 Me: 34 Suc: 37 F_Table: [37, 37, 39, 48, 56]
[32, 37, 39, 41]

Process finished with exit code 0
```