Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών
6ο εξάμηνο – Βάσεις Δεδομένων
Υπεύθυνος Μαθήματος: Δ. Τσουμάκος
Υπεύθυνος Εργαστηρίου: Μ. Κόνιαρης

«Υλοποίηση και Οπτικοποίηση ενός B+ Δέντρου»

Σπυρόπουλος Νικόλας, ΑΜ: 03121202

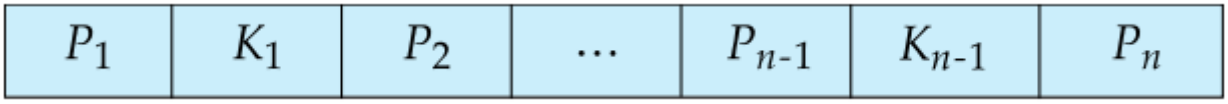
GitHub repository: https://github.com/ArionasMC/BPlusTree
Web app page: https://bplustree.onrender.com/

Περιεχόμενα

Εισαγωγή

Στο μάθημα των Βάσεων Δεδομένων διδαχθήκαμε την δομή B+ Δέντρων η οποία είναι ένα “n-ary” δέντρο [1] και χρησιμοποιείται για την υλοποίηση ευρετηρίων σε σχεσιακές βάσεις δεδομένων [2]. Συγκεκριμένα, η δομή αποτελείται από τρία ήδη κόμβων: την ρίζα, τους ενδιάμεσους κόμβους και τα φύλλα. Κύριο χαρακτηριστικό του δέντρου είναι η τάξη (order) του. Η τάξη είναι ένας ακέραιος αριθμός μεγαλύτερος ή ίσος του 2 που μας δείχνει πόσα το πολύ παιδιά μπορεί να έχει ένας κόμβος. Επιπλέον, κάθε κόμβος έχει το πολύ order-1 κλειδιά. Η χρήση της δομής ως ευρετήριο πραγματοποιείται συγκρίνοντας αυτά τα κλειδιά ώστε να βρούμε το μονοπάτι που πρέπει να ακολουθήσουμε στο δέντρο.

Η δομή κάθε κόμβου του δέντρου φαίνεται στην εικόνα 1. Με το P συμβολίζονται τα παιδιά-κόμβοι αν ο τρέχων κόμβος δεν είναι φύλλο ή pointers σε εγγραφές της βάσης αν είναι και με το K τα κλειδιά. Για όλους τους κόμβους ισχύει $K_1 < K_2 < \ldots < K_{n-1}$.



Εικόνα 1 – Δομή κόμβου (από τις διαφάνειες του μαθήματος)

Για τους κόμβους που δεν είναι φύλλα ισχύουν τα εξής:

- Όλα τα κλειδιά στο υποδέντρο που δείχνει το P_1 είναι μικρότερα του K_1
- Για 2 <= i <= n-1, όλα τα κλειδιά του υποδέντρου που δείχνει το P_i είναι μεγαλύτερα ή ίσα του K_{i-1} και μικρότερα του K_i
- Όλα τα κλειδιά του υποδέντρου που δείχνει το P_n είναι μεγαλύτερα του K_{n-1}

Για τους κόμβους που είναι φύλλα ισχύουν τα εξής:

- Για i = 1,2,…,n-1, ο P_i δείχνει στην εγγραφή με κλειδί K_i
- Αν L_i, L_j είναι κόμβοι φύλλα και i<j, το κλειδιά του L_i είναι μικρότερα ή ίσα των κλειδιών του L_j
- Ο P_n δείχνει στον επόμενο κόμβο φύλλο

Βάσει αυτής της δομής και των αλγορίθμων εισαγωγής και διαγραφής στοιχείων που περιγράφονται στο βιβλίο [2], δημιούργησα μια εφαρμογή (web app) σε Python χρησιμοποιώντας της βιβλιοθήκη Flask [3] που μου επέτρεψε να οπτικοποιήσω τα αποτελέσματα της εφαρμογής των αλγορίθμων με HTML, CSS και JavaScript. Όλος ο κώδικας της εφαρμογής βρίσκεται σε GitHub repository μαζί με οδηγίες εγκατάστασης της εφαρμογής τοπικά στα αγγλικά (https://github.com/ArionasMC/BPlusTree). Οδηγίες εγκατάστασης της εφαρμογής στα ελληνικά υπάρχουν στο επόμενο κεφάλαιο της αναφοράς.

Εάν δεν επιθυμείτε να εγκαταστήσετε τοπικά την εφαρμογή, μπορείτε να επισκεφθείτε τον παρακάτω σύνδεσμο για να την δοκιμάσετε (https://bplustree.onrender.com/). Ανέβασα την εφαρμογή στην υπηρεσία Render [4] η οποία με το δωρεάν πλάνο της παρέχει αυτή την δυνατότητα με αρκετά περιορισμένους πόρους (υπάρχει περίπτωση να χρειαστεί και 1 λεπτό για την εκπλήρωση του request στην σελίδα).

Εγκατάσταση Εφαρμογής

Αρχικά, για να τρέξει η εφαρμογή απαιτείται η εγκατάσταση της Python 3, συγκεκριμένα η έκδοση 3.11.4 χρησιμοποιήθηκε για την δημιουργία της (https://www.python.org/downloads/). Στην συνέχεια, χρειάζεται να κάνετε clone το repository της εφαρμογής χρησιμοποιώντας git:

```
$ git clone https://github.com/ArionasMC/BPlusTree.git
```

Αλλιώς μπορείτε να κατεβάσετε τον κώδικα μέσω του GitHub πατώντας το πράσινο κουμπί “Code” πάνω δεξιά και στην συνέχεια επιλέγοντας “Download ZIP”. Σε αυτή την περίπτωση, αφού το κατεβάσετε πρέπει να αποσυμπιέσετε το αρχείο.

Για να κατεβάσετε τις βιβλιοθήκες που απαιτούνται από την εφαρμογή, βρισκόμενοι στον κεντρικό φάκελο που περιέχει το αρχείο “requirements.txt”, εκτελείτε τις εντολές:

```
$ pip install –u pip
```

```
$ pip install –r requirements.txt
```

Για να τρέξετε την εφαρμογή εκτελείτε την εντολή:

```
$ python3 webapp.py
```

Τέλος, ανοίξτε τον browser της επιλογής σας (π.χ. Chrome) και γράψτε στην γραμμή διευθύνσεων:

```
localhost:8000
```

Επίδειξη εφαρμογής

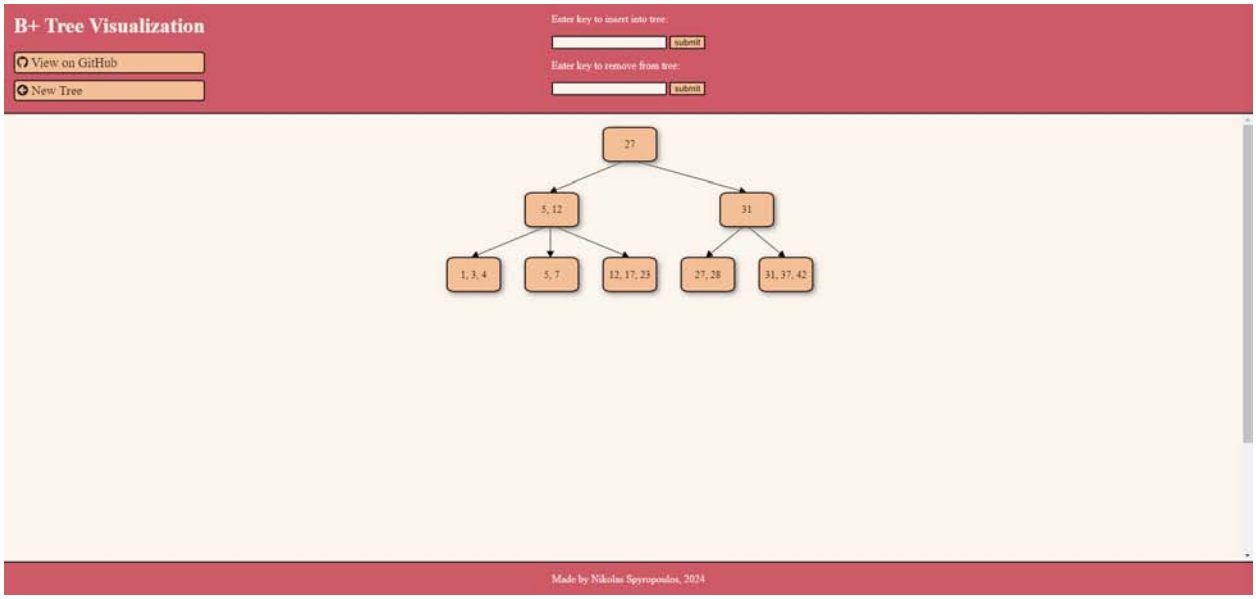
Αφού έχουμε ανοίξει την εφαρμογή ακολουθώντας τα παραπάνω βήματα, βρισκόμαστε στην αρχική σελίδα (εικόνα 2) στην οποία καλούμαστε να επιλέξουμε την δομή του δέντρου και τον «τύπο» του. Ο τύπος του δέντρου δηλώνει ως τι θα αντιμετωπίζονται τα κλειδιά των κόμβων. Υπάρχουν δύο επιλογές είτε ως ακέραιοι αριθμοί (int), είτε ως συμβολοσειρές (string).

Εικόνα 2 – Αρχική σελίδα εφαρμογής

Αφού επιλέξουμε ό,τι επιθυμούμε (π.χ. order: 4 και type: int) πατάμε “Submit” και προχωράμε στην κύρια σελίδα της εφαρμογής στην οποία γίνεται η οπτικοποίηση της δομής και των αλγορίθμων του B+ Δέντρου (Εικόνα 3).

Εικόνα 3 – Κύρια σελίδα της εφαρμογής

Στα δύο πεδία που υπάρχουν πάνω και κεντρικά της σελίδας μπορούμε να ορίσουμε κάθε φορά ένα κλειδί που θέλουμε να εισαγάγουμε στο δέντρο ή ένα κλειδί που θέλουμε να διαγράψουμε. Έτσι, μπορούμε να δημιουργήσουμε B+ δέντρα και να δούμε πώς τρέχουν οι αλγόριθμοι πάνω τους. Ένα παράδειγμα δέντρου φαίνεται στην εικόνα 4.



Εικόνα 4 – Παράδειγμα οπτικοποίησης ενός B+ Δέντρου

Υλοποίηση δομής του δέντρου

Ολόκληρη η υλοποίηση της δομής του B+ Δέντρου και των αλγορίθμων επεξεργασίας του βασίζονται στον ψευδοκώδικα του βιβλίου [2] ο οποίος έχει ακολουθηθεί όσο πιο πιστά γινόταν.

Πιστεύω ότι μόνο η δομή του κόμβου αξίζει να αναλυθεί καθώς άπαξ και κατανοηθεί, η κατανόηση του κώδικα των κυρίων συναρτήσεων “insert” και “delete” και όλων των υπό-συναρτήσεων που χρησιμοποιούν θα είναι άμεση διαβάζοντας τον ψευδοκώδικα του βιβλίου παράλληλα με τον κώδικα σε Python που έγραψα.

Η κλάση “Node” όπως φαίνεται στην εικόνα 5, αναπαριστά τον κόμβο του δέντρου. Το Boolean “is_leaf” είναι αληθές αν ο κόμβος είναι φύλλο και ψευδές σε κάθε άλλη περίπτωση. Η λίστα keys αρχικά είναι κενή, και αναπαριστά τα κλειδιά του κόμβου. Το μέγεθος της δεν μπορεί να περνά την τιμή order-1. Η λίστα pointers που επίσης είναι κενή στην αρχή, αναπαριστά τα παιδιά του κάθε κόμβου και το μέγιστο μέγεθός της είναι ίσο με order. Αν ο κόμβος είναι φύλλο και η λίστα pointers είναι μεγέθους n <= order, τότε τα πρώτα n-1 στοιχεία δείχνουν σε εγγραφές (συνήθως strings) και το τελευταίο στοιχείο δείχνει στο επόμενο φύλλο. Αν το φύλλο είναι το τέρμα δεξιά φύλλο του δέντρο του τελευταίο στοιχείο δεν υπάρχει μέσα στην λίστα και για αυτή την οριακή περίπτωση υπάρχει ειδική συνάρτηση (“__is_right_edge”) μέσα στον κώδικα για την σωστή υλοποίηση των αλγορίθμων. Αν ο κόμβος δεν είναι φύλλο τότε και τα n στοιχεία της λίστας pointers είναι δείκτες στους κόμβους παιδιά του τρέχοντος κόμβου.

```python
class Node:
    """B+ Tree Node can be either root, inner node or leaf"""
    def __init__(self, is_leaf=False):
        """
        - is_leaf   : boolean
            Check if this node is a leaf.
        - keys      : list
            Node keys with which all the sorting magic works.
        - pointers  : list
            If node is leaf then the pointers are the value parameter
            of the insert and delete methods but the last one which
            is a pointer to the next Node.
            If node is non-leaf then the pointers are pointers to this
            node's children Nodes.
        """
        self.is_leaf = is_leaf
        self.keys = []
        self.pointers = []

    def getDict(self):
        if self.is_leaf:
            return {'keys': self.keys, 'children': [], 'is_leaf': True}
        return {'keys': self.keys, 'children': [n.getDict() for n in self.pointers], 'is_leaf': False}

    def __repr__(self):
        return f"Node({self.keys})"
```

Εικόνα 5 – κλάση Node

Βιβλιογραφία

[1] https://en.wikipedia.org/wiki/B%2B_tree#:~:text=A%20B%2B%20tree%20is%20an,with%20two%20or%20more%20children.

[2] Συστήματα Βάσεων Δεδομένων, 7η Έκδοση, Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Εκδόσεις: Μ. Γκιούρδας

[3] https://flask.palletsprojects.com/en/3.0.x/

[4] https://render.com/