Παραδοτέο 1

Μάθημα: Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα

Μέλη Ομάδας:

Ραμαντάν Κονόμι - ΑΜ: 1115201800281

Θεμιστοκλής Παπαθεοφάνους - ΑΜ: 1115202100227

Μάριος Γιαννόπουλος - ΑΜ: 1115202000032

Πίνακας Περιεχομένων

1. HopscotchTable 2
2. Δομές Δεδομένων 2
3. RobinHoodTable 2
4. CuckooTable 2
5. Δομές Δεδομένων 2
6. Δομές Δεδομένων 2
   - 6.1. CuckooEntry 2
   - 6.2. Πίνακες και Χωρητικότητα 3
7. Συναρτήσεις Κατακερματισμού 3
   - 7.1. Συνάρτηση `h_1` 3
   - 7.2. Συνάρτηση `h_2` 3
8. Μηχανισμός Εισαγωγής (The Kick Process) 3
   - 8.1. Βήματα Εισαγωγής 4
   - 8.2. Όριο Εκτοπίσεων (MAX_KICKS) 4
9. Ανακατακερματισμός (Rehash) 4
   - 9.1. Συνθήκη Ανακατακερματισμού 4
   - 9.2. Διαδικασία 4
10. Αναζήτηση (Search) 4

Ανάλυση της Κλάσης **HopscotchTable**

HopscotchTable

Δομές Δεδομένων

Ανάλυση της Κλάσης **RobinHoodTable**

RobinHoodTable

Ανάλυση της Κλάσης **CuckooTable**

CuckooTable

Δομές Δεδομένων

Η κλάση `CuckooTable<Key>` υλοποιεί τον αλγόριθμο Κατακερματισμού Cuckoo (Cuckoo Hashing), μια προηγμένη τεχνική κατακερματισμού που εγγυάται σταθερό χρόνο αναζήτησης `O(1)` στην **χειρότερη** περίπτωση. Σε αντίθεση με τις μεθόδους ανοικτής διευθυνσιοδότησης που βασίζονται σε chaining ή γραμμική διερεύνηση (linear probing), το Cuckoo Hashing χρησιμοποιεί πολλαπλούς πίνακες και συναρτήσεις κατακερματισμού για να εξασφαλίσει ότι κάθε στοιχείο βρίσκεται ακριβώς σε μία από τις λίγες πιθανές θέσεις του.

Δομές Δεδομένων

Ο πίνακας Cuckoo αποτελείται από δύο πίνακες, `table_1` και `table_2`, ίσου `capacity`.

CuckooEntry

Κάθε θέση στους πίνακες μπορεί να περιέχει ένα `CuckooEntry` ή να είναι κενή (`std::nullopt`). Η δομή `CuckooEntry` αποθηκεύει το κλειδί (`Key`) και έναν συσσωρευτή δεικτών (`std::vector indices`), καθώς η υλοποίηση επιτρέπει σε πολλαπλές εγγραφές να κατακερματίζονται στο ίδιο κλειδί.

```
struct CuckooEntry {
    Key key;
    std::vector<size_t> indices;
};
```

Πίνακες και Χωρητικότητα

Ο πίνακας διαχειρίζεται δύο εσωτερικούς πίνακες `table_1` και `table_2`, καθένας με χωρητικότητα `capacity`.

```
std::vector<std::optional<CuckooEntry<Key>>> table1;
std::vector<std::optional<CuckooEntry<Key>>> table2;
size_t capacity;
```

Συναρτήσεις Κατακερματισμού

Χρησιμοποιούνται δύο ανεξάρτητες συναρτήσεις κατακερματισμού, `h_1` και `h_2`, για την αντιστοίχιση ενός κλειδιού σε μια θέση στους `table_1` και `table_2` αντίστοιχα.

Συνάρτηση `h_1`

Η `h_1` είναι η τυπική συνάρτηση κατακερματισμού, χρησιμοποιώντας την `std::hash<Key>`.

```
    size_t h1(const Key& key) const {
    return key_hasher(key) % capacity;
}
```

Συνάρτηση `h_2`

Η `h_2` προκύπτει από μια απλή κυκλική μετατόπιση (rotation) του αρχικού hash value, εξασφαλίζοντας μια δεύτερη, ανεξάρτητη διεύθυνση.

```
    size_t h2(const Key& key) const {
    size_t h = key_hasher(key);
    // Κυκλική μετατόπιση αριστερά (e.g., κατά 1 bit)
    return ((h << 1) | (h >> (sizeof(size_t) * 8 - 1))) % capacity;
}
```

Μηχανισμός Εισαγωγής (The Kick Process)

Η εισαγωγή ενός νέου στοιχείου γίνεται μέσω της διαδικασίας «εκτόπισης» (kicking) που υλοποιείται στη μέθοδο `insert_internal`.

Βήματα Εισαγωγής

Η διαδικασία εισαγωγής ακολουθεί τους εξής κανόνες:
1. **Έλεγχος:** Το νέο στοιχείο ελέγχεται αρχικά στη θέση `h_1` του `table_1`.
2. **Εκτόπιση (Kick):** Εάν η θέση `h_1` είναι κατειλημμένη, το υπάρχον στοιχείο εκτοπίζεται (διώχνεται). Το εκτοπισμένο στοιχείο αναζητά στη συνέχεια τη δική του εναλλακτική θέση.
3. **Εναλλακτική Θέση:** Το εκτοπισμένο στοιχείο προσπαθεί να εισαχθεί στη θέση `h_2` του `table_2`.
4. **Επανάληψη:** Εάν και η θέση `h_2` είναι κατειλημμένη, το στοιχείο που βρισκόταν εκεί εκτοπίζεται και η διαδικασία επιστρέφει στον `table_1` για το εκτοπισμένο στοιχείο (χρησιμοποιώντας τη δική του συνάρτηση `h_1`).

Η διαδικασία αυτή συνεχίζεται έως ότου βρεθεί μια κενή θέση.

Όριο Εκτοπίσεων (MAX_KICKS)

Για να αποφευχθεί ο ατέρμονος βρόχος (cycle) που μπορεί να προκληθεί από την κυκλική εκτόπιση στοιχείων, η υλοποίηση θέτει ένα όριο `MAX_KICKS` (σταθερά `500`). Αν το όριο αυτό ξεπεραστεί, θεωρείται ότι έχει εντοπιστεί ένας κύκλος και απαιτείται ανακατακερματισμός.

Ανακατακερματισμός (Rehash)

Συνθήκη Ανακατακερματισμού

Ο ανακατακερματισμός ενεργοποιείται όταν η εισαγωγή ενός στοιχείου αποτύχει να βρει μια κενή θέση εντός του ορίου `MAX_KICKS`.

Διαδικασία

Η μέθοδος `rehash()` εκτελεί τα εξής:
1. **Διπλασιασμός Χωρητικότητας:** Ο `capacity` διπλασιάζεται (`capacity` $\rightarrow 2 \times$ `capacity`).
2. **Επαναφορά Πινάκων:** Δημιουργούνται νέοι, κενοί πίνακες `table_1` και `table_2` με τη νέα χωρητικότητα.
3. **Επανεισαγωγή:** Όλα τα στοιχεία από τους παλιούς πίνακες μετακινούνται και επανεισάγονται στους νέους πίνακες.

Αναζήτηση (Search)

Η αναζήτηση (`search`) είναι η απλούστερη λειτουργία του Cuckoo Hashing, καθώς το στοιχείο μπορεί να βρίσκεται μόνο σε δύο πιθανές θέσεις:

1. Στη θέση `h_1(key)` του `table_1`.
2. Στη θέση `h_2(key)` του `table_2`.

Η `search` απλά ελέγχει αυτές τις δύο θέσεις. Αν το κλειδί βρεθεί σε οποιονδήποτε από τους δύο πίνακες, επιστρέφεται ένας δείκτης (`std::optional<std::vector<size_t>*>`) στη συλλογή δεικτών του. Αυτό εγγυάται `O(1)` χρόνο αναζήτησης στην χειρότερη περίπτωση.
