Kauno technologijos universitetas
Informatikos fakultetas

DUOMENŲ STRUKTŪROS (P175B014)

3 -OJO LABORATORINIO DARBO ATASKAITA

Rokas Sičiovas IFF-9/5

Studentas

doc. Eimutis Karčiauskas

Dėstytojas

Kaunas 2020

Remove(K key)

```
@Override
public V remove(K key) {
    if(key == null)
    {
        throw new IllegalArgumentException("key is null in remove");
    }

    index = hash(key, ht);
    if(table[index] == null)
    {
        return null;
    }
    V value = null;
    int count = -1;
    for(Node<K, V> n = table[index]; n != null; n = n.next)
    {
        count++;
        if(n.key.equals(key) && count == 0){
            value = n.value;
            table[index] = table[index].next;
            size--;
            return value;
        }
        else if(n.key.equals(key) && count == size - 1){
            value = n.value;
            table[index].next = null;
            size--;
            return value;
        }
        else if(n.key.equals(key))
        {
            value = n.value;
            table[index].next = table[index].next.next;
            size--;
            return value;
        }
        table[index] = n;
    }
    if(table[index] == null)
        chainsCounter--;
    return value;
}
```

Metodo idėja:

Pirmiausia randamas maišos lentelės “bucket” indeksas. Jei list tuščias, gražinama null, jei ne, atliekamas pašalinimas.

Asimptotinis sudėtingumas – **O(n).**

containsValue(Object value)

```
public boolean containsValue(Object value) {
    if(value == null)
    {
        throw new IllegalArgumentException("Value is null in containsValue");
    }

    for(int i = 0; i < table.length; i++)
    {
        for(Node<K, V> n = table[i]; n != null; n = n.next) {
            if (n.value.equals(value)) {
                return true;
            }
        }
    }
    return false;
}
```

Metodo idėja:

Iteruojama per kiekvieną maišos lentelės mazgą ir lyginama su duotąja reikšme. Jei randama sutampanti reikšmė, gražinama true. Jei perėjus per visa maišos lentelę nerandamas sutampantis elementas, gražinama false.

Asimptotinis sudėtingumas – **O(n^2).**

putIfAbsent(K key, V value)

```
public V putIfAbsent(K key, V value) {
    if(key == null)
    {
        throw new IllegalArgumentException("Key is null in putIfAbsent");
    }
    if(!contains(key))
    {
        put(key, value);
        return null;
    }
    return get(key);
}
```

Metodo idėja:

Contains metodu patikrinama, ar egzistuoja reikšmė su duotuoju raktu, jei ne, reikšmė įterpiama duotajam raktui į maišos lentelę ir gražinama null. Jei taip, gražinama reikšmė duotajam raktui.

Asimptotinis sudėtingumas – **O(n).**

numberOfEmpties()

```java
public int numberOfEmpties() {
    int count = 0;

    for(int i = 0; i < table.length; i++)
    {
        if(table[i] == null)
        {
            count++;
        }
    }
    return count;
}
```

Metodo idėja:

Iteruojama per visus maišos lentelės “buckets” ir sumuojamos null reikšmės

Asimptotinis sudėtingumas – **O(n).**

putAll(Map<K, V> map)

```java
public void putAll(Map<K, V> map) {
    if(map == null)
    {
        throw new IllegalArgumentException("Map is null in putAll");
    }
    HashMap<K , V> newMap = (HashMap<K, V>) map;
    for(int i = 0; i < newMap.table.length; i++)
    {
        for(Node<K, V> n = newMap.table[i]; n != null; n = n.next){
            put(n.key, n.value);
        }
    }
}
```

Metodo idėja:

Iteruojama per duotos maišos lentelės reikšmes, kiekvienai reikšmei kviečiamas put metodas, kuris prideda reikšmę.

Asimptotinis sudėtingumas – **O(n^2).**

Replace(K key, V oldValue, V newValue)

```
public boolean replace(K key, V oldValue, V newValue) {
    if(key == null)
    {
        throw new IllegalArgumentException("Key is null in replace");
    }

    for(int i = 0; i < table.length; i++)
    {
        if(contains(key) && get(key).equals(oldValue))
        {
            put(key, newValue);
            return true;
        }
    }
    return false;
}
```

Metodo idėja:

Gaunama reikšmė pagal turimą raktą. Jei ji sutampa su sena reikšme, pakeičiama į naują reikšmę ir gražinama true. Jei ne, gražinama false.

Asimptotinis sudėtingumas – **O(n).**

GREITAVEIKOS TESTAVIMAS

Reikėjo ištestuoti remove() ir putAll() metodus.

Greitaveikos matavimų metodai:

```
// Removes car by id
@org.openjdk.jmh.annotations.Benchmark
public void remove(FullMap fullMap)
{
    fullMap.ids.forEach(id -> fullMap.carsMap.remove(id));
}

// puts all elements to collection
@org.openjdk.jmh.annotations.Benchmark
public void putAll(FullMap fullMap) {
    HashMap<String, Car> temp = new HashMap<>();
    temp.putAll(fullMap.carsMap);
}
```

Greitaveikos rezultatai:

Benchmark	(elementCount)	Mode	Cnt	Score	Error	Units
Benchmark.putAll	10000	avgt	5	679,108 ±	214,470	us/op
Benchmark.putAll	20000	avgt	5	2175,553 ±	1807,834	us/op
Benchmark.putAll	40000	avgt	5	6115,728 ±	6158,761	us/op
Benchmark.putAll	80000	avgt	5	14023,941 ±	10280,432	us/op
Benchmark.remove	10000	avgt	5	341,267 ±	21,016	us/op
Benchmark.remove	20000	avgt	5	1063,101 ±	173,148	us/op
Benchmark.remove	40000	avgt	5	2687,641 ±	1149,522	us/op
Benchmark.remove	80000	avgt	5	7549,076 ±	11786,587	us/op

Charakteristikos kompiuterio, kuris buvo naudojamas atliekant greitaveikos matavimą: