CENG 215 Veri Yapıları Laboratuvarı
Ödev 3: Cuckoo Hashing

Şevket Umut ÇAKIR

20 Aralık 2023

Ödevde Cuckoo özetleme ile ilgili ekleme ve arama metotlarının yazılması istenmektedir. Ekleme ve arama işlemi yaparken diziler kullanılabilir fakat hash fonksiyonlarında amaç sabit adım sayısına($O(1)$) sahip erişimin olmasıdır. Sizlerin dizilere erişim sayılarınızı kontrol altında tutabilmek için kod 2 sınıfı verilmiştir. Diziden bir değer alabilmek için `HashTable` sınıfının `get` metodunu kullanmanız gerekir. Yapılan okuma işlemleri sonda olarak değerlendirilir, ekleme ve arama işlemlerinin sonundaki sonda işlemleri sayısı olması gereken sayı ile aynı olmalıdır. Yapılan az veya fazla sondalar testlerin başarısız olmasına neden olacaktır. Örnek bir ekleme işlemi ve sonda sayıları bölüm 2'de verilmiştir.

1 Cuckoo Özetleme

Cuckoo özetlemede, özet(hash) fonksiyonları birbirinden farklı birden fazla özet tablosu bulunur. Değerler tablolardan bir tanesinde bulunur ve ilgili tablonun özet değeri ile erişilebilir. Teorik derste işlenen özetlemeden farklı olarak bu ödevde maksimum deneme sayısı(`MAX_REHASH_ATTEMPTS`) bulunmaktadır. Ödevde Cuckoo özetlemenin iki tablolu halinin gerçekleştirilmesi beklenmektedir. Ekleme işlemi sırasında boş bir yer bulunana kadar değerler iki tablo arasında değiştirilir. Değiştirme sayısı maksimum deneme sayısını aşarsa ekleme işlemi başarısız olacaktır. Arama işleminde ise aranan değer iki tablodan bir tanesinde olacağı için sonda sayısı ikiyi geçmeyecektir.

2 Örnek Ekleme İşlemi

Tablo boyutu(N) **7** olan Cuckoo özetleme gerçekleştirilecektir. Birinci ve ikinci tablonun özet fonksiyonları aşağıda verilmiştir(N: tablo boyutu, R: N'den küçük en büyük asal sayı):

$$H_1 = x \ mod \ N \tag{1}$$

$$H_2 = R - (x \ mod \ R) \tag{2}$$

Tabloya **8, 10, 2, 99, 3, 85, 47, 6, 71** değerleri sırasıyla eklenecektir. Eklenecek sayıların hash değerleri Tablo 1'de verilmiştir. Bir değeri eklerken gerçekleştirilebilecek en çok sonda sayısı bu örnek için beştir(5). Son değer olan 71'in eklenmesi başarısız olmaktadır çünkü her iki tabloda yapılan sonda sayısı 5 olmuştur ve tüm değerleri yerleştirilecek uygun bir konum bulunamamıştır. Ekleme işlemleri sonucundaki özet tabloları ve sonda sayıları Tablo 2 ile 11 arasında verilmiştir.

Tablo 1: Eklenecek sayıların hash değerleri. $H_1 = x \ mod \ N$, $H_2 = R - (x \ mod \ R)$

Key	**H1**	**H2**
8	1	2
10	3	5
2	2	3
99	1	1
3	3	2
85	1	5
47	5	3
6	6	4
71	1	4

Tablo 2: Başlangıç

i	T1	T2
0		
1		
2		
3		
4		
5		
6		
Sonda	0	0

Tablo 3: 8 ekleme

i	T1	T2
0		
1	8	
2		
3		
4		
5		
6		
Sonda	1	0

Tablo 4: 10 ekleme

i	T1	T2
0		
1	8	
2		
3	10	
4		
5		
6		
Sonda	2	0

Tablo 5: 2 ekleme

i	T1	T2
0		
1	8	
2	2	
3	10	
4		
5		
6		
Sonda	3	0

Tablo 6: 99 ekleme

i	T1	T2
0		
1	99	
2	2	8
3	10	
4		
5		
6		
Sonda	4	1

Tablo 7: 3 ekleme

i	T1	T2
0		
1	99	
2	2	8
3	3	
4		
5		10
6		
Sonda	5	2

Tablo 8: 85 ekleme

i	T1	T2
0		
1	85	99
2	2	8
3	3	
4		
5		10
6		
Sonda	6	3

Tablo 9: 47 ekleme

i	T1	T2
0		
1	85	99
2	2	8
3	3	
4		
5	47	10
6		
Sonda	7	3

Tablo 10: 6 ekleme

i	T1	T2
0		
1	85	99
2	2	8
3	3	
4		
5	47	10
6	6	
Sonda	8	3

Tablo 11: 71 ekleme(başarısız)

i	T1	T2
0		
1	8	99
2	2	3
3	10	
4		
5	47	85
6	6	
Sonda	11	5

3 Ödev Sınıfları

Ödev sınıfları ve yapılarının hepsi **AbstractCuckoo.java** dosyası içinde bulunmaktadır, bu nedenle aşağıdaki kodların satır numaraları 1'den başlamamaktadır.

3.1 HashFunction Arayüzü

Tamsayılar üzerinde çalışan bir özet arayüzüdür. İçinde bir adet metot bulunmaktadır, parametre olarak verilen sayının özet değerini döndürür. Bu arayüzü gerçekleştiren sınıflar bulunmaktadır. Kullandığınız özet tablosunda hangi gerçekleştirmenin(implementation) önemi bulunmamaktadır, ilgili özet fonksiyonunun `getHash` metodunu çağırmak yeterli olacaktır.

Listing 1: HashFunction

```
7  interface HashFunction {
8      public int getHash(int value);
9  }
```

3.2 HashTable Sınıfı

Dizilere yapılan erişimi kontrol altında tutmak için HashTable sınıfı oluşturulmuştur. Bu sınıf içinde hash tablosunun tutulacağı dizi, hash fonksiyonu, tablo boyutu ve sonda sayısı gibi bilgiler saklanmaktadır. Tablo

boyutuna ihtiyacınız olmayacağı için(tablodaki konumları `hash` metodunu kullanarak hesaplamanız gerekmektedir) `private` olarak verilmiştir, diğer özellikler için `get` metotları bulunmaktadır. Hash değerini hesaplamak için bu sınıfın `hash` metodu kullanılabilir. Tablodan bir değer okumak için `get`, tabloya bir değer yazmak için `set` metodu kullanılabilir.

Listing 2: HashTable

```
14  class HashTable {
15      // Tablo boyutu
16      private int tableSize;
17      // Özet tablosu
18      private Integer[] table;
19      // Sonda sayısı
20      private int probeCount;
21      // Özet fonksiyonu
22      private HashFunction hashFunction;
23      // Sıfırlama işlemine karşı koruma
24      private String resetPass;
25      // Constructor
26      public HashTable(int tableSize, HashFunction hashFunction, String resetPass) {
27          this.tableSize = tableSize;
28          this.table = new Integer[tableSize];
29          this.hashFunction = hashFunction;
30          this.probeCount = 0;
31          this.resetPass = resetPass;
32      }
33      /**
34       * Tablonun hash fonksiyonunu çağırır
35       * @param key Değer
36       * @return Hash değeri
37       */
38      public int hash(int key) {
39          return hashFunction.getHash(key);
40      }
41
42      /**
43       * Tablodaki sonda sayısını verir
44       * @return Sonda sayısı
45       */
46      public int getProbeCount() {
47          return probeCount;
48      }
49
50      /**
51       * Sonda sayısını sıfırlar, öğrencilerin bu metodu çağırması beklenmemektedir.
52       * @param pass
53       */
54      public void resetProbeCount(String pass) {
55          if (resetPass.equals(pass)) {
56              probeCount = 0;
57          }
58      }
59
60      /**
61       * Tablodan bir değer okur ve sonda sayısını artırır
62       * @param i Okunacak konum
63       * @return i konumundaki değer
64       */
65      public Integer get(int i) {
66          if (i >= tableSize) {
```

```
67              throw new ArrayIndexOutOfBoundsException(
68                      "Özet tablosunun boyutu aşılmış. Tablo boyutu: " + tableSize + "
                        ↪  Erişilen indeks: " + i + "\n");
69          }
70          probeCount++;
71          return table[i];
72      }
73
74      /**
75       * Tablo içindeki bir konuma değer atar
76       * @param i Konum
77       * @param key Atanacak değer
78       */
79      public void set(int i, int key) {
80          if (i >= tableSize) {
81              throw new ArrayIndexOutOfBoundsException(
82                      "Özet tablosunun boyutu aşılmış. Tablo boyutu: " + tableSize + "
                        ↪  Erişilen indeks: " + i + "\n");
83          }
84          table[i] = key;
85      }
86
87      /**
88       * Tablo içindeki diziyi liste olarak verir, bilgi amaçlıdır
89       * @return Elemanların listesi
90       */
91      public List<Integer> getTableAsList() {
92          return Arrays.asList(table);
93      }
94
95      /**
96       * Hash sınıfının türünü döndürür, bilgi amaçlıdır
97       * @return
98       */
99      public Class getHashClass() {
100         return hashFunction.getClass();
101     }
102 }
```

3.3 AbstractCuckoo Soyut Sınıfı

AbstractCuckoo sınıfı ödev için kullanacağınız Cuckoo sınıfının üst sınıfıdır. İçinde tanımlı, ekleme sırasında yapılacak maksimum sonda sayısı(`MAX_REHASH_ATTEMPTS`) ve iki adet hash tablosu bulunmaktadır. `toString` ve `resetProbeCounts` metotları test işlemleri için kullanılmaktadır, öğrencilerin kullanımı için değildir.

Listing 3: AbstractCuckoo

```
107 public abstract class AbstractCuckoo {
108     // Ekleme yaparken kullanılacak maksimum sonda sayısı
109     protected final int MAX_REHASH_ATTEMPTS;
110     // Tablo 1
111     protected HashTable table1;
112     // Tablo 2
113     protected HashTable table2;
114     /**
115      * Constructor
116      * @param max_rehash_attempts Maksimum sonda sayısı
117      * @param table1 Tablo 1
```

```
118         * @param table2 Tablo 2
119         */
120        public AbstractCuckoo(int max_rehash_attempts, HashTable table1, HashTable table2) {
121            MAX_REHASH_ATTEMPTS = max_rehash_attempts;
122            this.table1 = table1;
123            this.table2 = table2;
124        }
125
126        /**
127         * Sonda sayılarını sıfırlamak için kullanılır, öğrencilerin kullanması beklenmez
128         * @param pass
129         */
130        public void resetProbeCounts(String pass) {
131            table1.resetProbeCount(pass);
132            table2.resetProbeCount(pass);
133        }
134
135        /**
136         * Cuckoo tablosunu metne dönüştüren metot
137         */
138        @Override
139        public String toString() {
140            StringBuilder sb = new StringBuilder();
141            List<Integer> t1 = table1.getTableAsList();
142            List<Integer> t2 = table2.getTableAsList();
143            sb.append("|------------|\n");
144            sb.append(String.format("|%2s|%4s|%4s|\n", "i", "t1", "t2"));
145            sb.append("|--|----|----|\n");
146            for (int i = 0; i < t1.size(); i++) {
147                sb.append(String.format("|%2d|%4d|%4d|\n",
148                        i,
149                        t1.get(i),
150                        t2.get(i)));
151            }
152            sb.append("|--|----|----|\n");
153            sb.append(String.format("|PC|%4s|%4s|\n",
154                    table1.getProbeCount(),
155                    table2.getProbeCount()));
156            sb.append("|------------|\n");
157            sb.append("H1: " + table1.getHashClass().getName() + "\n");
158            sb.append("H2: " + table2.getHashClass().getName() + "\n");
159            sb.append("MAX_REHASH_ATTEMPTS: " + MAX_REHASH_ATTEMPTS + "\n");
160            return sb.toString().replace(' ', '.');
161        }
162
163        // Ekleme metodu
164        public abstract boolean insert(int key);
165        // Arama metodu
166        public abstract boolean search(int key);
167    }
```

3.4 Cuckoo Sınıfı

Ödevde içeriğini değiştirebileceğiniz tek sınıftır. Sınıfın yapıcı metodunu(constructor) değiştirmemeniz gerekmektedir. İçerisinde bulunan ekleme ve arama metotlarını yazınız.

Listing 4: AbstractCuckoo

```java
public class Cuckoo extends AbstractCuckoo{
    /**
     * Yapıcı metodu ve imzasını değiştirmeyin, aksi halde testleriniz çalışmaz
     */
    public Cuckoo(int max_rehash_attempts, HashTable table1, HashTable table2) {
        super(max_rehash_attempts, table1, table2);
    }

    @Override
    public boolean insert(int key) {
        // Metodu yazınız
        return false;
    }

    @Override
    public boolean search(int key) {
        // Metodu yazınız
        return false;
    }

}
```

4 Ödev Açıklamaları

- Ödevler `https://bilmoodle.pau.edu.tr` adresindeki arayüzlerden cevaplanacaktır. Ödevinizdeki kod dosyasını kişisel bilgisayarınıza indirip, ödevi çözüp tekrar sisteme geri yükleyebilirsiniz fakat sınıf, dosya ve paket yapısının korunduğundan emin olmalısınız.
- Ödevler bireysel olarak cevaplanmalıdır, grup çalışması yapmak yasaktır
- Ödevleri yapay zeka sohbet robotlarına cevaplatmak yasaktır
- Ödevlerde kopya kontrolü yapılacaktır ve benzerlik oranı belirli bir yüzdenin üzerinde olan ödevler kopya olarak değerlendirilecektir. Değişken isimlerini değiştirmek, kodların sırasını değiştirmek anlaşılabilen yöntemlerdir, lütfen bu yola başvurmayınız.
- Ödevin son teslim tarihi **27.12.2023 saat 23:59**'dur. Belirtilen saatte sistem kapanacaktır, lütfen son ana kadar beklemeden ödevinizi gönderin.