Háskóli Íslands

Tölvunarfræði 2

Vikublað 11

Höfundur:
Teitur Guðmundarson

Kennari:
Páll Melsteð

29. mars 2019

1. Insert the keys E A S Y Q U T I O N in that order into an initially empty table of M = 5 lists, using separate chaining. Use the hash function 11 $k\%M$ to transform the kth letter of the alphabet into a table index.

```
key hash gildi
  E   0    0

0 E0
1 null
2 null
3 null
4 null

key hash gildi
  A   1    1

0 E0
1 A1
2 null
3 null
4 null

key hash gildi
  S   4    2

0 E0
1 A1
2 null
3 null
4 S2

key hash gildi
  Y   0    3

0 E0 -> Y3
1 A1
2 null
3 null
4 S2

key hash gildi
  Q   2    4

0 E0 -> Y3
1 A1
2 Q4
3 null
4 S2

key hash gildi
  U   1    5

0 E0 -> Y3
1 A1 -> U5
2 Q4
3 null
4 S2

key hash gildi
  T   0    6

```

```
0 E0 -> Y3 -> T6
1 A1 -> U5
2 Q4
3 null
4 S2

key hash gildi
  I   4    7

0 E0 -> Y3 -> T6
1 A1 -> U5
2 Q4
3 null
4 S2 -> I7

key hash gildi
  O   0    8

0 E0 -> Y3 -> T6 -> O8
1 A1 -> U5
2 Q4
3 null
4 S2 -> I7

key hash gildi
  N   4    9

0 E0 -> Y3 -> T6 -> O8
1 A1 -> U5
2 Q4
3 null
4 S2 -> I7 -> N9
```

2. Write a program to find values of a and M , with M as small as possible, such that the hash function $(a*k)\%M$ for transforming the kth letter of the alphabet into a table index produces distinct values (no collisions) for the keys S E A R C H X M P L . The result is known as a perfect hash function.

Daemi2.java

```java
public static int[] reykjaHash() {
    int[] gildi = new int[2];

    int[] bokstf = {19, 5, 1, 18, 3, 8, 24, 13, 16, 12};

    for(int m = 2; m <= 100; m++) {
        for(int a = 1; a <= 1000; a++) {
            Set<Integer> hashes = new HashSet<>();

            for(int i = 0; i < bokstf.length; i++) {
                int hash = (a * k) \% m;
                hashes.add(hash);
            }

            if (hashes.size() == 10) {
                gildi[0] = a;
                gildi[1] = m;
                return gildi;
            }
        }
    }
    return null;
}
```

3. Give the contents of a linear-probing hash table that results when you insert the keys E A S Y Q U T I O N in that order into an initially empty table of initial size M = 4 that is expanded with doubling whenever half full. Use the hash function 11 $k\%M$ to transform the kth letter of the alphabet into a table index.

Daemi3.java

```
key hass gildi
  E   3    0

0 null      2 null
1 null      3 E0

key hass gildi
  A   3    1

0 A1      2 null
1 null    3 E0

Hasstafla af tvöfaldri stærð og M verður að 8

0 null      4 null
1 null      5 null
2 null      6 null
3 null      7 null

endurinnsetning A1
key hass gildi
  A    3   1

0 null      4 null
1 null      5 null
2 null      6 null
3 A1        7 null

endurinnsetning E0
key hass gildi
  E    7   0

0 null      4 null
1 null      5 null
2 null      6 null
3 A1        7 E0

key hass gildi
  S   1    2

0 null      4 null
1 S2        5 null
2 null      6 null
3 A1        7 E0

key hass gildi
  Y   3    3

0 null      4 Y3
1 S2        5 null
2 null      6 null
3 A1        7 E0

Hasstafla af tvöfaldri stærð og M verður að 16

```

```
 0 null      8 null
 1 null      9 null
 2 null     10 null
 3 null     11 null
 4 null     12 null
 5 null     13 null
 6 null     14 null
 7 null     15 null

endurinnsetning A1
key hass gildi
  A   11   1

 0 null      8 null
 1 null      9 null
 2 null     10 null
 3 null     11 A1
 4 null     12 null
 5 null     13 null
 6 null     14 null
 7 null     15 null

endurinnsetning E0
key hass gildi
  E    7   0

 0 null      8 null
 1 null      9 null
 2 null     10 null
 3 null     11 A1
 4 null     12 null
 5 null     13 null
 6 null     14 null
 7 E0       15 null

endurinnsetning S2
key hass gildi
  S    1   2

 0 null      8 null
 1 S2        9 null
 2 null     10 null
 3 null     11 A1
 4 null     12 null
 5 null     13 null
 6 null     14 null
 7 E0       15 null

endurinnsetning Y3
key hass gildi
  Y    3   3

 0 null      8 null
 1 S2        9 null
 2 null     10 null
 3 Y3       11 A1
 4 null     12 null
 5 null     13 null
 6 null     14 null
 7 E0       15 null

```

```
key hass gildi
  Q   11   4

 0 null      8 null
 1 S2        9 null
 2 null     10 null
 3 Y3       11 A1
 4 null     12 Q4
 5 null     13 null
 6 null     14 null
 7 E0       15 null

key hass gildi
  U    7   5

 0 null      8 U5
 1 S2        9 null
 2 null     10 null
 3 Y3       11 A1
 4 null     12 Q4
 5 null     13 null
 6 null     14 null
 7 E0       15 null

key hass gildi
  T   12   6

 0 null      8 U5
 1 S2        9 null
 2 null     10 null
 3 Y3       11 A1
 4 null     12 Q4
 5 null     13 T6
 6 null     14 null
 7 E0       15 null

key hass gildi
  I    3   7

 0 null      8 U5
 1 S2        9 null
 2 null     10 null
 3 Y3       11 A1
 4 I7       12 Q4
 5 null     13 T6
 6 null     14 null
 7 E0       15 null

Hasstafla af tvöfaldri stærð og M verður að 32

 0 null     16 null
 1 null     17 null
 2 null     18 null
 3 null     19 null
 4 null     20 null
 5 null     21 null
 6 null     22 null
 7 null     23 null
 8 null     24 null
 9 null     25 null
10 null     26 null
```

```
178  11 null     27 null
179  12 null     28 null
180  13 null     29 null
181  14 null     30 null
182  15 null     31 null
183
184  endurinnsetning A1
185  key hass gildi
186    A   11   1
187
188   0 null     16 null
189   1 null     17 null
190   2 null     18 null
191   3 null     19 null
192   4 null     20 null
193   5 null     21 null
194   6 null     22 null
195   7 null     23 null
196   8 null     24 null
197   9 null     25 null
198  10 null     26 null
199  11 A1       27 null
200  12 null     28 null
201  13 null     29 null
202  14 null     30 null
203  15 null     31 null
204
205  endurinnsetning E0
206  key hass gildi
207    E   23   0
208
209   0 null     16 null
210   1 null     17 null
211   2 null     18 null
212   3 null     19 null
213   4 null     20 null
214   5 null     21 null
215   6 null     22 null
216   7 null     23 E0
217   8 null     24 null
218   9 null     25 null
219  10 null     26 null
220  11 A1       27 null
221  12 null     28 null
222  13 null     29 null
223  14 null     30 null
224  15 null     31 null
225
226  endurinnsetning I7
227  key hass gildi
228    I    3   7
229
230   0 null     16 null
231   1 null     17 null
232   2 null     18 null
233   3 I7       19 null
234   4 null     20 null
235   5 null     21 null
236   6 null     22 null
237   7 null     23 E0
238   8 null     24 null
```

```
 9 null     25 null
10 null     26 null
11 A1       27 null
12 null     28 null
13 null     29 null
14 null     30 null
15 null     31 null

endurinnsetning Q4
key hass gildi
  Q   27   4

 0 null     16 null
 1 null     17 null
 2 null     18 null
 3 I7       19 null
 4 null     20 null
 5 null     21 null
 6 null     22 null
 7 null     23 E0
 8 null     24 null
 9 null     25 null
10 null     26 null
11 A1       27 Q4
12 null     28 null
13 null     29 null
14 null     30 null
15 null     31 null

endurinnsetning S2
key hass gildi
  S   17   2

 0 null     16 null
 1 null     17 S2
 2 null     18 null
 3 I7       19 null
 4 null     20 null
 5 null     21 null
 6 null     22 null
 7 null     23 E0
 8 null     24 null
 9 null     25 null
10 null     26 null
11 A1       27 Q4
12 null     28 null
13 null     29 null
14 null     30 null
15 null     31 null

endurinnsetning T6
key hass gildi
  T   28   6

 0 null     16 null
 1 null     17 S2
 2 null     18 null
 3 I7       19 null
 4 null     20 null
 5 null     21 null
 6 null     22 null
```

```
 7 null     23 E0
 8 null     24 null
 9 null     25 null
10 null     26 null
11 A1       27 Q4
12 null     28 T6
13 null     29 null
14 null     30 null
15 null     31 null

endurinnsetning U5
key hass gildi
  U    7   5

 0 null     16 null
 1 null     17 S2
 2 null     18 null
 3 I7       19 null
 4 null     20 null
 5 null     21 null
 6 null     22 null
 7 U5       23 E0
 8 null     24 null
 9 null     25 null
10 null     26 null
11 A1       27 Q4
12 null     28 T6
13 null     29 null
14 null     30 null
15 null     31 null

endurinnsetning Y3
key hass gildi
  Y   19   3

 0 null     16 null
 1 null     17 S2
 2 null     18 null
 3 I7       19 Y3
 4 null     20 null
 5 null     21 null
 6 null     22 null
 7 U5       23 E0
 8 null     24 null
 9 null     25 null
10 null     26 null
11 A1       27 Q4
12 null     28 T6
13 null     29 null
14 null     30 null
15 null     31 null

key hass gildi
  O    5   8

 0 null     16 null
 1 null     17 S2
 2 null     18 null
 3 I7       19 Y3
 4 null     20 null
 5 O8       21 null
```

```
 6 null     22 null
 7 U5       23 E0
 8 null     24 null
 9 null     25 null
10 null     26 null
11 A1       27 Q4
12 null     28 T6
13 null     29 null
14 null     30 null
15 null     31 null

key hass gildi
  N   26   9

 0 null     16 null
 1 null     17 S2
 2 null     18 null
 3 I7       19 Y3
 4 null     20 null
 5 O8       21 null
 6 null     22 null
 7 U5       23 E0
 8 null     24 null
 9 null     25 null
10 null     26 N9
11 A1       27 Q4
12 null     28 T6
13 null     29 null
14 null     30 null
15 null     31 null
```

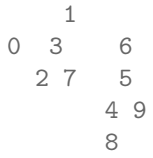
4. *Cuckoo hashing*. Develop a symbol-table implementation that maintains two hash tables and two hash functions. Any given key is in one of the tables, but not both. When inserting a new key, hash to one of the tables; if the table position is occupied, replace that key with the new key and hash the old key into the other table (again kick- ing out a key that might reside there). If this process cycles, restart. Keep the tables less than half full. This method uses a constant number of equality tests in the worst case for search (trivial) and amortized constant time for insert.
Hér þarf klára útfærsluna á get, put og delete í skránni CuckooST.java

Daemi4.java

```
1 // code
```

5. Draw the tree corresponding to the id[] array depicted at right. Can this be the result of running weighted quick-union? Explain why this is impossible or give a sequence of operations that results in this array.

i	0	1	2	3	4	5	6	7	8	9
id[i]	1	1	3	1	5	6	1	3	4	5



Þetta tré getur ekki komið úr running quick-union. Hægra tréið: 1, 6, 5, 4, 9 , 8 hefur hæðina 4 sem er meira en $logN$.

6. *Random connections.* Develop a UF client ErdosRenyi that takes an integer value N from the command line, generates random pairs of integers between 0 and N-1 , calling connected() to determine if they are connected and then union() if not (as in our development client), looping until all sites are connected, and printing the number of connections generated. Package your program as a static method count() that takes N as argument and returns the number of connections and a main() that takes N from the command line, calls count() , and prints the returned value. *Erdös-Renyi model.* Use your client from Exercise 1.5.17 to test the hypothesis that the number of pairs generated to get one component is 1/2N ln N.
Hér þarf bara að skila einu forriti sem leysir seinna verkefnið.

Daemi6.java

```java
import edu.princeton.cs.algs4.StdOut;
import java.util.ArrayList;
import java.util.List;

public class Daemi6 {
    private class Tilraun {
        int fjoldi;
        int por;
        public Tilraun(int fjoldi, int por) {
            this.fjoldi = fjoldi;
            this.por = por;
        }
    }
    public static void main(String[] args) {
        new Daemi6().geraTilraun();
    }
    private void geraTilraun() {
        List<Tilraun> tilraunir = new ArrayList<>();
        int fjoldi = 16;
        for(int i = 0; i < 10; i++) {
            int porTil = count(fjoldi, false);
            Tilraun experiment = new Tilraun(fjoldi, porTil);
            tilraunir.add(experiment);
            fjoldi *= 2;
        }
    }
    private static int count(int fjoldi, boolean oskar) {
        int tengingar = 0;
        UnionFind unionFind = new UnionFind(fjoldi);
        while(unionFind.count() != 1) {
            int stradi = StdRandom.uniform(fjoldi);
            int viktor = StdRandom.uniform(fjoldi);
            tengingar++;
            if (oskar) {
                StdOut.println("Tengingar: " + stradi + " - " + viktor);
            }
            if (!unionFind.connected(stradi, viktor)) {
                unionFind.union(stradi, viktor);
            }
        }
        return tengingar;
    }
}
```