Tölvunarfræði II
Skiladæmi 11

Sesar Hersisson - seh32

Dæmi 1:

$$
\begin{array}{ccc}
E & \xrightarrow{5*11\%5} & 0 \\
A & \xrightarrow{1*11\%5} & 1 \\
S & \xrightarrow{19*11\%5} & 4 \\
Y & \xrightarrow{25*11\%5} & 0 \\
Q & \xrightarrow{17*11\%5} & 2 \\
U & \xrightarrow{21*11\%5} & 1 \\
T & \xrightarrow{20*11\%5} & 0 \\
I & \xrightarrow{9*11\%5} & 4 \\
O & \xrightarrow{15*11\%5} & 0 \\
N & \xrightarrow{14*11\%5} & 4
\end{array}
$$

$$
\begin{array}{c|c|ccccccc}
0 & \boxed{\text{E}} & \xrightarrow{.next} & \text{Y} & \xrightarrow{.next} & \text{T} & \xrightarrow{.next} & \text{O} \\
1 & \boxed{\text{A}} & \xrightarrow{.next} & \text{U} & & & & \\
2 & \boxed{\text{Q}} & & & & & & \\
3 & \boxed{\phantom{\text{Q}}} & & & & & & \\
4 & \boxed{\text{S}} & \xrightarrow{.next} & \text{I} & \xrightarrow{.next} & \text{N} & &
\end{array}
$$

Dæmi 2:

```java
public class perfectHash
{
  public static void main(String[] args)
  {
    ST<Character, Integer> st = new ST<Character, Integer>();
    st.put('S', 19); st.put('E', 5); st.put('A', 1); st.put('R', 18);
    st.put('C', 3); st.put('H', 8); st.put('X', 24); st.put('M', 13);
    st.put('P', 16); st.put('L', 12);


    int m = 10;
    int[] funct = new int[st.size()];
    boolean iterate = true;
    while(iterate)
    {
      for(int i = 1; i < m; i++)
      {
        int j = 0; boolean bol = true;
        for(Character staf: st.keys())
        {
          funct[j] = (i*st.get(staf))%m;
          for(int t = 0; t < j && bol; t++)
          {
            if(funct[t] == funct[j])  bol = false;
          }
          if(!bol) break;
          j++;
        }
        if(bol)
        {
          System.out.println("m: " + m + " a: " + i);
          iterate = false;
          i = m;
        }
      }
      m++;
    }
  }
}
```

Interactions | Console | Compiler Output

```
Welcome to DrJava.  Working directory is /Users/sessihers/OneDrive/Tolfr102
> run perfectHash
m: 20 a: 1
>
```

Running main Method of Current Document

Dæmi 3:

Við vitum að í lokin mun $m = 32$ þar sem þegar það eru kominn 2 stök í fylkið stækkum við fylkið í 8, svo þegar 4 stök eru kominn stækkum við í 16 og svo loks í 32 þegar 8 stök hafa verið sett í fylkið.

$$
\begin{array}{lcl}
E & \xrightarrow{5*11\%32} & 23 \\
A & \xrightarrow{1*11\%32} & 11 \\
S & \xrightarrow{19*11\%32} & 17 \\
Y & \xrightarrow{25*11\%32} & 19 \\
Q & \xrightarrow{17*11\%32} & 27 \\
U & \xrightarrow{21*11\%32} & 7 \\
T & \xrightarrow{20*11\%32} & 28 \\
I & \xrightarrow{9*11\%32} & 3 \\
O & \xrightarrow{15*11\%32} & 5 \\
N & \xrightarrow{14*11\%32} & 26
\end{array}
$$

0	
1	
2	
3	I
4	
5	O
6	
7	U
8	
9	
10	
11	A
12	
13	
14	
15	
16	
17	S
18	
19	Y
20	
21	
22	
23	E
24	
25	
26	N
27	Q
28	T
29	
30	
31	

Dæmi 4:

```
public class CuckooST<Key, Value> {
  private static final int INIT_CAPACITY = 16;

  private int n;           // number of key-value pairs in the symbol
     table
  private int m;           // size of the tables
  private Key[][] keys;      // the keys
  private Value[][] vals;    // the values


  // taken from MurmurHash3 http://sites.google.com/site/murmurhash/
  // and
     http://facebook.github.io/jcommon/util/jacoco/com.facebook.util.digest/MurmurHas

  private long rotateLeft64(long x, int r) {
    return (x << r) | (x >>> (64 - r));
  }

  private long fmix(long k) {
    k ^= k >>> 33;
    k *= 0xff51afd7ed558ccdL;
    k ^= k >>> 33;
    k *= 0xc4ceb9fe1a85ec53L;
    k ^= k >>> 33;
    return k;
  }

  public long murmurhash(long data, long seed) {
    long c1 = 0x87c37b91114253d5L;
    long c2 = 0x4cf5ad432745937fL;

    long h1 = seed, h2 = seed;

    long k1 = data;
    k1 *= c1;
    k1 = rotateLeft64(k1, 31);
    k1 *= c2;
    h1 ^= k1;

    h1 ^= 8;
    h2 ^= 8;

    h1 += h2;
    h2 += h1;

    return (fmix(h1) + fmix(h2));
  }

  public CuckooST() {
    this(INIT_CAPACITY);
  }

  public CuckooST(int capacity) {
    m = capacity;
    n = 0;
```

```
    keys = (Key[][]) new Object[2][];
    keys[0] = (Key[]) new Object[m];
    keys[1] = (Key[]) new Object[m];

    vals = (Value[][]) new Object[2][];
    vals[0] = (Value[]) new Object[m];
    vals[1] = (Value[]) new Object[m];
  }

  public int size() {
    return n;
  }

  public boolean isEmpty() {
    return size() == 0;
  }

  public boolean contains(Key key) {
    if (key == null) throw new IllegalArgumentException("argument to
        contains() is null");
    return get(key) != null;
  }


  private void resize(int capacity) {
    CuckooST<Key, Value> temp = new CuckooST<Key, Value>(capacity);
    for (int i = 0; i < m; i++) {
      if (keys[0][i] != null) {
        temp.put(keys[0][i], vals[0][i]);
      }
      if (keys[1][i] != null) {
        temp.put(keys[1][i], vals[1][i]);
      }
    }
    keys = temp.keys;
    vals = temp.vals;
    m    = temp.m;
  }

  public void put(Key key, Value val) {
    if (key == null) throw new IllegalArgumentException("first
        argument to put() is null");

    if (val == null) {
      delete(key);
      return;
    }

    if (n > (0.8*m)) {
      resize(2*m);
    }

    boolean iterate = true;
    //Ef true skiptum vid ut stokum i fyrra fylkinu, annars seinna
    boolean last = true;
    //Ef vid rekumst aftur a key og viljum skipta honum ut
    //erum vid i hringras og thurfum ad resize-a
    Key[] checked = (Key[]) new Object[m];
```

```
int pos = 0;
while(iterate)
{
  long h = murmurhash(key.hashCode(), 42);
  int h1 = ((int)(h & 0x7fffffff)) % m;
  int h2 = ((int)((h >> 32) & 0x7fffffff)) % m;
  if(keys[0][h1] == null)
  {
    keys[0][h1] = key;
    vals[0][h1] = val;
    n++;
    iterate = false;
  }
  else if(keys[1][h2] == null)
  {
    keys[1][h2] = key;
    vals[1][h2] = val;
    n++;
    iterate = false;
  }
  else
  {
    boolean loop = false;
    if(last)
    {
      Key tempKey = keys[0][h1];
      Value tempVal = vals[0][h1];
      for(int i = 0; i < pos; i++)
      {
        if(tempKey == checked[i])
        {
          resize(2*m);
          i = pos;
          loop = true;
        }
      }
      if(!loop)
      {
        keys[0][h1] = key;
        vals[0][h1] = val;
        checked[pos] = key;
        pos++;
        key = tempKey;
        val = tempVal;
        last = !last;
      }
    }
    else
    {
      Key tempKey = keys[1][h2];
      Value tempVal = vals[1][h2];
      for(int i = 0; i < pos; i++)
      {
        if(tempKey == checked[i])
        {
          resize(2*m);
          i = pos;
          loop = true;
```

```
          }
        }
        if(!loop)
        {
          keys[1][h2] = key;
          vals[1][h2] = val;
          checked[pos] = key;
          key = tempKey;
          val = tempVal;
          last = !last;
        }
      }
    }
  }
}


public Value get(Key key) {
  if (key == null) throw new IllegalArgumentException("argument to
      get() is null");
  long h = murmurhash(key.hashCode(), 42);
  int h1 = ((int)(h & 0x7fffffff)) % m;
  int h2 = ((int)((h >> 32) & 0x7fffffff)) % m;

  if(keys[0][h1] == key) return vals[0][h1];
  if(keys[1][h2] == key) return vals[1][h2];
  else return null;


}

public void delete(Key key) {
  if (key == null) throw new IllegalArgumentException("argument to
      delete() is null");
  long h = murmurhash(key.hashCode(), 42);
  int h1 = ((int)(h & 0x7fffffff)) % m;
  int h2 = ((int)((h >> 32) & 0x7fffffff)) % m;

  if(keys[0][h1] == key)
  {
    keys[0][h1] = null;
    vals[0][h1] = null;
    n--;
    return;
  }
  if(keys[1][h2] == key)
  {
    keys[1][h2] = null;
    vals[1][h2] = null;
    n--;
    return;
  }
  return;
}

public Iterable<Key> keys() {
  Queue<Key> queue = new Queue<Key>();
  for (int i = 0; i < m; i++) {
```

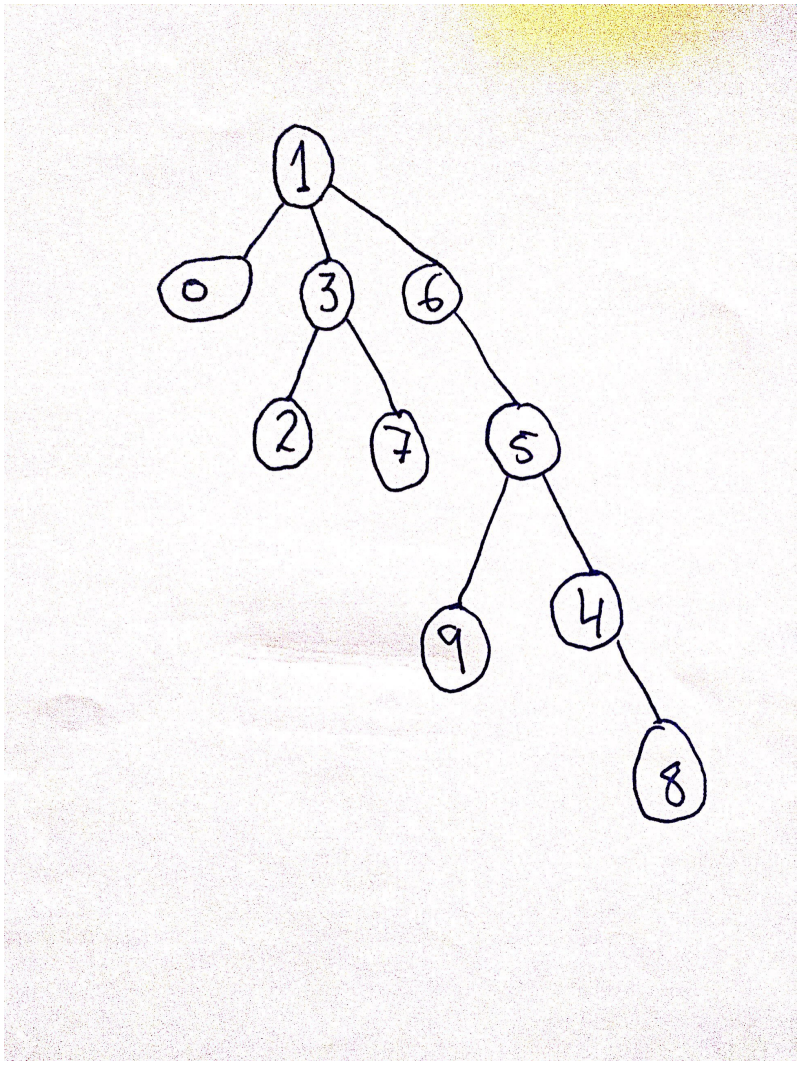
```java
      if (keys[0][i] != null) queue.enqueue(keys[0][i]);
      if (keys[1][i] != null) queue.enqueue(keys[1][i]);
    }
    return queue;
  }


  public static void main(String[] args) {
    CuckooST<Integer, Integer> st = new CuckooST<Integer, Integer>();
    for (int i = 0; i < 100; i++) {
      st.put(i, i*i);
    }

    // print keys
    for (Integer i : st.keys()) {
      StdOut.println(i + " " + st.get(i));
    }
    for (int i = 0; i < 100; i++) {
      if (!st.contains(i)) {
        System.out.println("Error: Key not found " + i );
      }
      Integer j = st.get(i);
      if (!j.equals(i*i)) {
        System.out.println("Error: Key found = " + i + ", wrong value
            " + j );
      }

    }
  }
}
```

Dæmi 5: Athugum að þetta er ekki hægt að fá út frá weighted quick union find þar sem hæðin á trénu er 5 en stærsta mögulega hæð trés með 9 stökum út frá weighted quick union find er $log9 \approx 4$



Dæmi 6:

```java
public class ErdosRenyi
{
  public static int count(int N)
  {
    WeightedQuickUnionUF uf = new WeightedQuickUnionUF(N);
    int counter = 0;
    while(uf.count() > 1)
    {
      counter++;
      int p = (int)(Math.random()*N);
      int q = (int)(Math.random()*N);
      if(!uf.connected(p,q))
      {
        uf.union(p,q);
      }
    }
    return counter;
  }

  public static void main(String[] args)
  {
    int N = Integer.parseInt(args[0]);
    int sum = 0;
    for(int i = 0; i < 1000; i++)
    {
      sum += count(N);
    }
    double counter = (double)(sum)/1000.0;
    double hype = 0.5*N*Math.log(N);
    System.out.println("Average number of pairs generated: " + counter
        + " and 1/2NlnN = " + hype + ".");
  }
}
```

```
> run ErdosRenyi 1000
Average number of pairs generated: 3698.983 and 1/2NlnN = 3453.8776394910683.
> run ErdosRenyi 10000
Average number of pairs generated: 48663.159 and 1/2NlnN = 46051.701859880915.
```