Skilaverkefni 9

TÖL203G: Tölvunarfræði 2

Axel Thor Aspelund
ata13@hi.is

17. mars 2022

3.4.4

```
s = "SEARCHXMPL"
M = 10
a = 1

for i in s:
   print((1*ord(i))%20)

l = []

for M in range(10,26):
    for a in range(1,100):
        l = []
        for i in s:
            dec = ord(i)
            elem = (a * dec) % M
            if elem in l:
                l = []
                break
            l.append(elem)
            if len(l) == 10:
                print("success")
                print(l)
                print("a: ",a)
                print("M: ",M)
```

Lægstu a og M sem þetta forrit fann eru $a = 1$ og $M = 20$. Sem gefa indexin

```
 S  E  A  R  C  H  X   M   P  L
[3, 9, 5, 2, 7, 12, 8, 17, 0, 16]
```

2

Tíminn sem það tekur að setja A Tale of Two Cities í eftirfarandi gagnagrindur. (sek.)

```
BST: 0.158
RedBlackBST: 0.134
SeparateChainingHashST: 0.088
LinearProbingHashST: 0.067
```

Hérna eru niðurstöður úr að leita að öllum orðum í War and Peace 10 sinnum.

```
BST: 1.345
RedBlackBST: 1.715
SeparateChainingHashST: 0.566
LinearProbingHashST: 0.283
```

3.4.31

```
private int hash(Key key, int index) {
    long h = murmurhash(key.hashCode(), 42);
    int h1 = ((int)(h & 0x7fffffff)) % m;
    int h2 = ((int)((h >> 32) & 0x7fffffff)) % m;
    int[] hs = {h1, h2};
    return hs[index];
  }

public void put(Key key, Value val) {
    int count = 0;
    if (key == null) {
      throw new IllegalArgumentException("first argument to put() is null");}
    long h = murmurhash(key.hashCode(), 42);
    int h1 = ((int)(h & 0x7fffffff)) % m;
    int h2 = ((int)((h >> 32) & 0x7fffffff)) % m;
    int[] hs = {h1, h2};

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

    if (n > (0.5*m)) {
      resize(2*m);
    }

    if (keys[0][h1] = null) {
      keys[0][h1] = key;
      vals[0][h1] = val;
      n++;
    }
    else {
      Key inTo = keys[0][h1];
      while(inTo != null){
        int h = hash(inTo, count % 2);
        Key tempKey = keys[count % 2][h];
        Value tempVal = vals[count % 2][h]];

        keys[count % 2][hash(key, count % 2)] = key;
        vals[count % 2][hash(key, count % 2)] = val;

        ++count;
        int H = hash(tempKey, count % 2);
        inTo = keys[count % 2][H];

        keys[count % 2][H] = tempKey;
        vals[count % 2][H] = tempVal;
```

```
        key = tempKey;
        val = tempVal;

        if (count >100){
          //max depth
          resize(m + 1);
          put(key, val);
        }
      }
      n++;
    }
  }
```

4.1.4

```
public boolean hasEdge(int v, int w){
    Iterator itr = adj[v].iterator();
    while (itr.hasNext()) {
      if (itr.next() == w) {
        return true;
      }
    }
    return false;
  }
```