Breakable Hash

July 21, 2021

Samuel Ballan

sb7875@nyu.edu

Hashing algorithm based on work by by Professor Daniel J. Bernstein

Our `BreakableHash` module has two methods: `call` executes a hash functions on a string, and `find_collision` will iterate through strings until a collision is found. The character set we iterate through is the standard 36 character set in Ruby, comprised of characters 0-9 and a-z.

[144]:
```
module BreakableHash
  extend self

  StartPrime = 5381
  BitShift = 7

  attr_accessor :mod_prime

  def call(str)
    hash = StartPrime
    str.each_codepoint do |c|
      hash = ((hash << BitShift) + hash) + c
    end
    output_num = hash % mod_prime
    "#{output_num}".rjust(mod_prime.to_s.length, "0")
  end

  def find_collision(iterations = 1000)
    hashes = {}

    (1..iterations).each do |num|
      str = num.to_s 36
      hash = self.call(str)

      if hashes[hash]
        puts "Found collision between #{str} and #{hashes[hash]} in #{num}␣
↪iterations"
        return hash
      end
      hashes[hash] = str
    end
  end
end
```

[144]:
```
:find_collision
```

We need to set the `mod_prime` value, which is used to make sure we have constant sized output. We'll set it to `97` for now. Note - this also represents the largest hash value we can create.

[145]:
```
BreakableHash.mod_prime = 97
```

[145]:
```
97
```

Let's see how many iterations it takes to find a collision:

[146]:
```
BreakableHash.find_collision
```

```
Found collision between 10 and j in 36 iterations
```

[146]:
```
"26"
```

We found a collision rather easily! If we make our `mod_prime` bigger, we'll reduce the likelihood of getting a collision:

[148]:
```
BreakableHash.mod_prime = 5869
BreakableHash.find_collision
```

```
Found collision between a0 and 3q in 360 iterations
```

[148]:
```
"2711"
```

It takes considerably longer to find a collision!

Source code can be found at https://github.com/sballan/nyu_blockchain_hw2