EDIN01 – Project 3

Tony Jin (to1643ji-s@student.lu.se)

December 12, 2018

1 Exercise 1

```
package main

import (
  "fmt"
  "strings"
  "strconv"
  "os"
  "io"
)

func WriteStringToFile(filepath, s string) error {
fo, err := os.Create(filepath)
if err != nil {
return err
}
defer fo.Close()

_, err = io.Copy(fo, strings.NewReader(s))
if err != nil {
return err
}

return nil
}

func mod(a int, n int) int {
  if a % n < 0 {
    return (a % n) + n
  } else {
    return a % n
  }
}

func hamming(u []int, z []int) int {
  if len(u) != len(z) {
    panic("Error: ‘u‘ and ‘z‘ not of the same length")
    return -1
  }

  d := 0
  for i := 0; i < len(u); i++ {
    if u[i] != z[i] {
      d = d + 1
    }
  }
  return d
}

func SeqSplit(s string) []int {
  a := strings.Split(s, "")
  b := make([]int, len(a))
  for i, v := range a {
    b[i], _ = strconv.Atoi(v)
  }
  return b
}

func LFSR(poly []int, state *[]int, n int) (out int, in int) {
  for i := 0; i < len(poly); i++ {
    in = in - poly[i] * (*state)[i]
  }

  out = (*state)[0]
  in = mod(in, n)
  *state = append((*state)[1:], in)

  return out, in
}

func Cycle(p []int, init []int, clock int) []int {
  initCopy := init
  seq := make([]int, 0)

  for i := 0; i < clock; i++ {
    out, _ := LFSR(p, &initCopy, 2)
    seq = append(seq, out)
  }

  return seq
}

func Generator(p []int, init []int, size int) [][]int {
  initCopy := init
  zero := make([]int, len(p))
  trials := [][]int{zero}

  for i := 0; i < size; i++ {
    trials = append(trials, initCopy)
    LFSR(p, &initCopy, 2)
  }

  return trials
}

func MinimizeP(p []int, trials [][]int, z []int) (int, []int) {
  trialsCopy := trials
  N := len(z)

  var minD int
  var minU []int
  for i := 0; i < len(trialsCopy); i++ {
    u := Cycle(p, trialsCopy[i], N)
    if i == 0 || hamming(u, z) < minD {
      minD = hamming(u, z)
      minU = trialsCopy[i]
    }
  }

  return minD, minU
}

func main() {
  z := SeqSplit("1100100111111111010110110100001101001101001101111000011100111001011101111000
  C1 := []int{1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1}
  C2 := []int{1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0}
  C3 := []int{1, 1, 0, 0, 1, 0, 0, 1, 0, 8, 0, 0, 1, 1, 0, 1, 0}

  // p(x) = x^13 + x^4 + x^3 + x^1 + 1
  p13 := []int{1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1}
  gen13 := []int{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}
  trials13 := Generator(p13, gen13, 8191)

  // p(x) = x^15 + x^1 + 1
  p15 := []int{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}
  gen15 := []int{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}
  trials15 := Generator(p15, gen15, 32767)

  // p(x) = x^17 + x^3 + 1
  p17 := []int{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0}
  gen17 := []int{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}
  trials17 := Generator(p17, gen17, 131071)

  _, K1 := MinimizeP(C1, trials13, z)
  _, K2 := MinimizeP(C2, trials15, z)
  _, K3 := MinimizeP(C3, trials17, z)

  prediction := make([]int, 0)
  for i := 0; i < 193; i++ {
    out1, _ := LFSR(C1, &K1, 2)
    out2, _ := LFSR(C2, &K2, 2)
    out3, _ := LFSR(C3, &K3, 2)

    if out1 + out2 + out3 > 1 {
      prediction = append(prediction, 1)
    } else {
      prediction = append(prediction, 0)
    }
  }

  if err := WriteStringToFile(
    "prediction",
    strings.Trim(strings.Join(strings.Fields(fmt.Sprint(prediction)), ""), "[]")); err != nil
    panic(err)
  }
}
```

The key we get is

$$K = (1000111111110, 111010011100101, 11101000011101111).$$

The probabilities for our inital states K_1, K_2, K_3 are $1 - \frac{45}{193}, 1 - \frac{45}{193}, 1 - \frac{68}{193}$, respectively.

2 Exercise 2

Let T seconds be the time it takes to calculate the $2^{13}+2^{15}+2^{17}$ states. Then, calculating 2^{45} states will take

$$\frac{2^{45}}{2^{13}+2^{15}+2^{17}} \approx 2367T \text{ days} \approx 6.4T \text{ years.}$$