TÖL101G - Tölvunarfræði 1

Vikublað 6

Lausn

Cloudcoder Æfingar

2.1.2

Einföld lausn væri að telja fjölda breyta sem eru `true`, en ef við notum **XOR** virkjann $\wedge$ getum við einfaldað kóðann með

```
public class CountOdd {
  // Notkun: r = odd1(a,b,c)
  // Fyrir: ekkert
  // Eftir: r er true ef oddatölufjöldi af a,b og c eru true, annars false
  public static boolean odd1(boolean a, boolean b, boolean c) {
    return a ^ b ^ c;
  }

  // Notkun: r = odd2(a,b,c)
  // Fyrir: ekkert
  // Eftir: r er true ef oddatölufjöldi af a,b og c eru true, annars false
  public static boolean odd2(boolean a, boolean b, boolean c) {
    int num = 0;
    if (a) { num++;}
    if (b) { num++;}
    if (c) { num++;}
    return ((num%2)==1);
  }

  public static void main(String[] args) {
    boolean []a = new boolean[args.length];
    for (int i = 0; i < a.length; i++) {
      a[i] = Boolean.parseBoolean(args[i]);
    }
    System.out.println(odd1(a[0],a[1],a[2]));
    System.out.println(odd2(a[0],a[1],a[2]));
  }
}
```

2.1.6

Hér var villa í bókinni, fallið sigmoid er í raun $\frac{1}{1+e^{-x}}$ en ekki $\frac{1}{1-e^{-x}}$.

```
public class Sigmoid {
  // Notkun: y = Sigmoid(x)
  // Fyrir:  ekkert
  // Eftir:  y = 1/(1+e^(-x))
```

```
  public static double Sigmoid(double x) {
    return 1/(1+Math.exp(-x));
  }

  public static void main(String[] args) {
    System.out.println(Sigmoid(Double.parseDouble(args[0])));
  }
}
```

2.1.12

```
public class Signum {
  public static int signum(int N) {
    if (N < 0) return -1;
    if (N == 0) return 0;
    return 1;
  }

  public static void main(String[] args) {
    System.out.println(signum(Integer.parseInt(args[0])));
  }
}
```

Verkefni 1

Forritskóðinn sjálfur er ekki upp á marga fiska, í raun bara umritun á því sem er í bókinni. Hins vegar skiptir meira máli hvernig hann er settur saman. Í main fallinu eru viðföng lesin inn, og séð til þess að þau séu skynsamleg. Öll vinnan fer fram inni í föllunum, main skilgreinir breytur og kallar á föll til að búa þær til og prenta út. Ef einhver sér aðeins main fallið og lýsingarnar á hinum föllunum þá á það að vera nóg til að skilja forritið.

Takið eftir að hin föllin "vita" ekki um hvert annað, t.d. er `printArray` alveg sama hvaðan fylkið kemur eða hvað er í því, þetta eykur líkur á því að við getum endurnýtt fallið.

```
public class SampleArray {


    // Notkun: x = readArray(N)
    // Fyrir:  N >= 0
    // Eftir:  Les inn N tölur af staðalinntaki og geymir
    //         þær í fylkinu x í þeirri röð sem þær komu
    public static int[] readArray(int N) {
        int[] a = new int[N];
        for (int i = 0; i < a.length; i++) {
            a[i] = StdIn.readInt();
        }
        return a;
```

```
}

// Notkun: b = sampleArray(a,M)
// Fyrir:  M >= 0 og a.length >= M
// Eftir:  b er fylki með M stök sem valin voru af handahófi
//         úr M stöðum í fylkinu a
public static int[] sampleArray(int[] a, int M) {
    // ATH kóðinn í bókinni eyðileggur a, svo við tökum afrit
    // af fylkinu og geymum í x
    int[] x = new int[a.length];
    for (int i = 0; i < x.length; i++) {
        x[i] = a[i];
    }

    for (int i = 0; i < M; i++) {
        int r = i + (int) (Math.random() * (x.length-i));
        int t = x[r];
        x[r] = x[i];
        x[i] = t;
    }

    // geymum M fyrstu gildin í x í nýju fylki b
    int[] b = new int[M];
    for (int i = 0; i < M; i++) {
        b[i] = x[i];
    }
    return b;
}

// Notkun: printArray(a)
// Fyrir:  ekkert
// Eftir:  búið er að prenta út stökin á a
public static void printArray(int[] a) {
    System.out.print("{");
    for (int i = 0; i < a.length; i++) {
        System.out.print(" " + a[i]);
    }
    System.out.println(" }");
}

public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    int M = Integer.parseInt(args[1]);
    if (N < 0 || M < 0 || M > N) {
        System.out.println("Wrong arguments: N = " + N + ", M = " + M);
        System.exit(1); // drepur forritið og hættir keyrslu
    }
```

```
        int[] a = readArray(N);
        int[] b = sampleArray(a,M);
        printArray(b);
    }
}
```

Verkefni 2

Hér notum við eina breytu til að halda utan um hversu oft i-ta stakið hefur komið fyrir í fylkinu. Þessi breyta er annað hvort hækkuð um 1 eða endurstillt á 1. Aðeins ef þessi breyta fær gildið 2 hækkum við `total` breytuna. Þetta kemur í veg fyrir að telja tölur sem koma oftar en tvisvar of oft. Hér setjum við fyrirskilyrðið að fylkið sé raðað, annað hvort í minnkandi eða vaxandi röð. Hins vegar mun forritið virka ef allar endurteknar tölur eru í hóp, t.d. myndi forritið virka fyrir inntakið `{2 2 1 3 3 3}`.

Fastayrðing lykkju tengir saman breyturnar `count` og `total` við þann hluta fylkisins sem við höfum skoðað þ.e. `a[0,...,i-1]`

```
public class CountTwo {

  // Notkun: x = countTwo(a)
  // Fyrir:  a er raðað fylki
  // Eftir:  x er fjöldi talna í a sem koma a.m.k tvisvar fyrir
  public static int countTwo(int[] a) {
    int total = 0;
    if (a.length <= 1) {
      return 0;
    }
    int count = 1;
    for(int i = 1; i < a.length; i++) {
      // total er fjöldi talna í a[0],..,a[i-1] sem koma fyrir
      //  oftar en einu sinni
      // a[i-1] kemur fyrir count sinnum í a[0],...,a[i-1]
      if (a[i] == a[i-1]) {
        count++; // tala endurtekin
      } else {
        count=1; // ný tala
      }
      if (count == 2) {
        total++; //
      }
    }

    return total;
  }

  public static void main(String[] args) {
    int[] a = new int[args.length];
    for (int i = 0; i < a.length; i++) {
```

```
      a[i] = Integer.parseInt(args[i]);
    }
    System.out.println(countTwo(a));
  }
}
```