TÖL101G - Tölvunarfræði 1

Vikublað 8

Lausn

Cloudcoder æfingar

3.1.9

```
public class ValidDNA {

    // Notkun: c = isValidDNA(s);
    // Fyrir:  ekkert
    // Eftir:  c er true ef einu stafirnir í s eru A,C,G,T
    public static boolean isValidDNA(String s) {
        boolean valid = true;
        for (int i = 0; i < s.length(); i++) {
            // I: valid is true if all characters from 0 to i-1
            //    in s are one of A,C,G or T
            char c = s.charAt(i);
            if (c != 'A' && c != 'C' && c != 'G' && c != 'T') {
                valid = false;
            }
        }
        return valid;
    }

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

3.1.15

Þessi lausn byggist á að fyrst bera saman lengdina á `s` og `t`, ef þeir eru ekki jafnlangir þurfum við ekki að gera neitt og skilum `false`. Annars leggjum við `s` við sjálft sig. Ef `t` er hring-hliðrun á `s` þá er til `i` þ.a. `s.substring(i) + s.substring(0,i) == t`. Ef `ss = s + s` þá er `t` hlutstrengur í `ss` þ.e. `ss.substring(i,i+t.length()) == t` og `ss.indexOf(t)` skilar ekki -1.

```
public class CircularShift {

    // Notkun: c = isCircular(s,t)
    // Fyrir:  ekkert
    // Eftir:  c er satt ef til er i þ.a.
    //         s.substring(i) + s.substring(0,i) == t
    public static boolean isCircular(String s, String t) {
        return (s.length() == t.length()) && (s + s).indexOf(t) != -1;
    }
```

```
    public static void main(String[] args) {
        System.out.println(isCircular(args[0],args[1]));
    }
}
```

3.1.17

Ein leið til að skipta upp strengnum eftir því hvar punktarnir eru er að nota `indexOf` og `substring`. Það er hins vegar til fall `split` sem gerir nákvæmlega þetta. Ef við köllum á `s.split(".")` geris hins vegar ekkert. Ástæðan er sú að strengurinn er túlkaður sem regular expression og punktur hefur sérstaka þýðingu þar. Því þarf að setja \á undan punktinum, reyndar tvö til að java haldi ekki að við séum að reyna að tala um einhvern spes staf. Þegar strengurinn er kominn í bita röðum við þeim saman aftur á bak og munum eftir að setja punktinn á sinn stað aftur.

```
public class reverseDomain {
    // Notkun: t = reverseDomain(s)
    // Fyrir:  ekkert
    // Eftir:  t er strengurinn s, þegar röð allra hlutstrengja
    //         í s, afmörkuðum með '.', hefur verið snúið við
    public static String reverseDomain(String s) {
        String[] a = s.split("\\.");
        String ret = "";
        for (int i = a.length-1; i >= 0; i--) {
            ret = ret + a[i];
            if (i > 0) {
                ret  = ret + ".";
            }
        }
        return ret;
    }

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

Æfingar

3.1.18

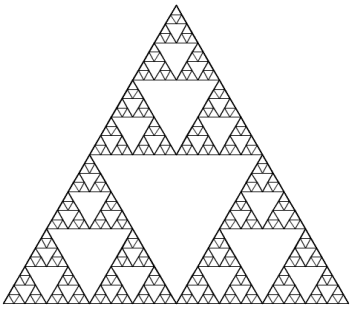
Fallið `mystery` snýr við strengnum s. Þetta sést á því hvað gerist fyrir tóman streng og streng með einum staf (grunntilfellið), í endurkvæma tilfellinu er seinni partinum b, snúið við og settur á undan fyrri partinum a sem er einnig snúið við.

Verkefni

Sierpinski

Við lausn þessa verkefnis er nauðsynlegt að negla niður hvað á að teiknast þegar kallað er á fallið með x, y hnitum og hliðarlengd sz. Um leið og það er komið á hreint er hægt að byrja að forrita fallið. Til að einfalda kóðann reiknum við út $sq3 = Math.sqrt(3.0)/2.0$ sem fær gildið $\frac{\sqrt{3}}{2}$ sem er hæð þríhyrningsins.

Þegar kallað er á forritið með viðfanginu 6 birtist þessi mynd



```
public class Sierpinski {

    // Notkun: draw(n,sz,x,y)
    // Fyrir: n >= 0, 0 <= x,y,sz <= 1
    // Eftir: Teiknar Sierpinsky þríhyrning með neðra vinstra horn
    //        í x,y með hliðarlengd sz og af dýpt n
    public static void draw(int n, double sz, double x, double y) {
        if (n == 0) return;
        double haed = Math.sqrt(3.0)/2.0;

        StdDraw.line(x,y,x+sz,y); // neðri línan
        StdDraw.line(x,y,x+sz/2.0, y + sz*haed); // efri vinstri línan
        StdDraw.line(x+sz,y,x+sz/2.0, y+ sz*haed); // efri hægri línan

        draw(n-1,sz/2.0,x,y); // neðri vinstri þríhyrningur
        draw(n-1,sz/2.0,x+sz/2.0,y); // neðri hægri þríhyrningur
        draw(n-1,sz/2.0,x+sz/4.0,y+sz*haed/2.0); // topp þríhyrningur
    }

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        draw(n,1.0,0.0,0.0);
    }
}
```

Hamming

Hér eru prentaðir allir strengir í Hamming fjarlægð k frá strengnum s. Í endurkvæmu lausninni er haldið utan um forskeytið `prefix` og við viljum prenta alla strengi í hamming

fjarlægð k frá `prefix+s` sem byrja á `prefix`. Í upphafi er kallað á fallið með `prefix` sem tóma strenginn. Tilfellin eru eftirfarandi.

- k er 0, þá er bara strengurinn `s` sjálfur í fjarlægð 0 og við prentum út `prefix+s`
- k er stærra en lengdin á `s`. Þá er enginn slíkur strengur til og við getum hætt.
- $0 \leq k \leq$`s.length()`. Við finnum út hver fyrsti bitinn í s og hver afgangurinn af s er. Köllum endurkvæmt á hamming með nýtt `prefix` viðfang og afgangnum af strengnum s. Annars vegar þegar við látum bitann vera og eigum þá eftir að flippa k bitum, hins vegar þegar við flippum fyrsta bitanum og eigum þá k-1 bita eftir.

```
public class Hamming {

  // Notkun: hamming(p,s,k)
  // Fyrir:  0 <= k, p og s eru 0-1 strengir
  // Eftir:  Allir strengir p + t hafa verið prentaðir,
  //         þar sem t er í hamming fjarlægð k frá s
  public static void hamming(String prefix, String s, int k) {
    if (s.length() < k) {
      return; // enginn strengur til
    }

    if (k == 0) {
      // einn strengur í fjarlægð 0 frá s, s sjálfur.
      System.out.print(prefix + s + " ");
    } else {
      int d = Integer.parseInt(s.substring(0,1));
      String rest = s.substring(1,s.length());
      hamming(prefix + d, rest, k);    // ekkert flip, k eftir
      hamming(prefix + (1-d), rest,k-1); // flip, k-1 eftir
    }
  }

  // Notkun: hamming(s,k)
  // Fyrir:  k >= 0, s er 0-1 strengur
  // Eftir:  allir strengir í hamming fjarlægð k frá s
  //         hafa verið prentaðir út
  public static void hamming(String s, int k) {
    hamming("",s,k);
  }

  public static void main(String[] args) {
    String s = args[0];
    int k = Integer.parseInt(args[1]);
    hamming(s,k);
    System.out.println();
  }
}
```