Tehtävä 4

Muodosta äärellinen automaatti D jolle löytyy...

(a) kaksi syötettä a ja b siten, että ab ei ole sallittu syöte.

(b) kaksi sallittua syötettä x ja y jotka tuottavat tulosteet $p = D(x)$ sekä $q = D(y)$ siten, että xy on myös sallittu syöte ja $D(xy) \neq pq$.

Ratkaisu:

(a) Olkoon $D = (Q, \Sigma, \delta, q_0, F)$ äärellinen automaatti, missä $Q = \{s_0, s_1, s_2, s_3, s_4, s_5\}$, $\Sigma = \{0, 1\}$, $q_0 = s_0$, $F = \{s_3\}$ ja $\delta : Q \times \Sigma \to Q \times (\Sigma \cup \{\varepsilon\})$ on funktio siten, että

$$\begin{aligned}
\delta(s_0, 0) &= (s_1, \varepsilon), & \delta(s_0, 1) &= (s_2, \varepsilon),\\
\delta(s_1, 0) &= (s_1, \varepsilon), & \delta(s_1, 1) &= (s_3, \varepsilon),\\
\delta(s_2, 0) &= (s_3, \varepsilon), & \delta(s_2, 1) &= (s_2, \varepsilon),\\
\delta(s_3, 0) &= (s_4, \varepsilon), & \delta(s_3, 1) &= (s_5, \varepsilon),\\
\delta(s_4, 0) &= (s_4, \varepsilon), & \delta(s_4, 1) &= (s_4, \varepsilon),\\
\delta(s_5, 0) &= (s_5, \varepsilon), & \delta(s_5, 1) &= (s_5, \varepsilon).
\end{aligned}$$

Tällöin $D(01)$ tuottaa tilat (s_0, s_1, s_3) ja on siten sallittu syöte. Vastaavasti $D(10)$ tuottaa tilat (s_0, s_2, s_3) ja on siten sallittu syöte. Kuitenkin $D(0110)$ tuottaa $(s_0, s_1, s_3, s_5, s_5)$ eikä siten ole sallittu koska $s_5 \notin F$.

(b) Olkoon $D = (Q, \Sigma, \delta, q_0, F)$ äärellinen automaatti, missä $Q = \{s_0, s_1, s_2, s_3, s_4\}$, $\Sigma = \{0, 1\}$, $q_0 = s_0$, $F = Q \setminus \{s_0\}$ ja $\delta : Q \times \Sigma \to Q \times (\Sigma \cup \{\varepsilon\})$ on funktio siten, että

$$\begin{aligned}
\delta(s_0, 0) &= (s_1, p), & \delta(s_0, 1) &= (s_2, q),\\
\delta(s_1, 0) &= (s_3, \alpha), & \delta(s_1, 1) &= (s_3, \beta),\\
\delta(s_2, 0) &= (s_4, \gamma), & \delta(s_2, 1) &= (s_4, \Delta),\\
\delta(s_3, 0) &= (s_3, \varepsilon), & \delta(s_3, 1) &= (s_3, \varepsilon),\\
\delta(s_4, 0) &= (s_4, \varepsilon), & \delta(s_4, 1) &= (s_4, \varepsilon).
\end{aligned}$$

Tällöin $D(0) = p$ ja $D(1) = q$ ovat sallittuja syötteitä. Lisäksi $D(01) = p\beta$ on sallittu syöte ($s_3 \in F$), mutta $D(01) = p\beta \neq pq$.

Tehtävä 5

Lataa tietokoneelle seuraavat tekstitiedostot:

(a) Iliad ja Odysseia `http://www.gutenberg.org/files/59306/59306-0.txt`

(b) Moby Dick `http://www.gutenberg.org/files/2701/2701-0.txt`

(c) Miljoona piin desimaalia `https://www.pi2e.ch/blog/wp-content/uploads/2017/03/pi_dec_1m.txt`

(d) Miljoona (pseudo)satunnaista merkkiä `https://luisto.fi/PseudoRandom.txt`

Vertaa kaikissa tiedostoissa alkuperäistä tiedostokokoa sekä kompressoimasi (esim. zip) tiedoston kokoa. Mitä huomaat? Miksi piin desimaalit tiivistyvät lähes yhtä hyvin kuin englanninkielinen teksti?

Ratkaisu:

Tiedostojen koot eri kompressointialgoritmeilla on esitetty alla olevassa taulukossa yksiköissä tavu.

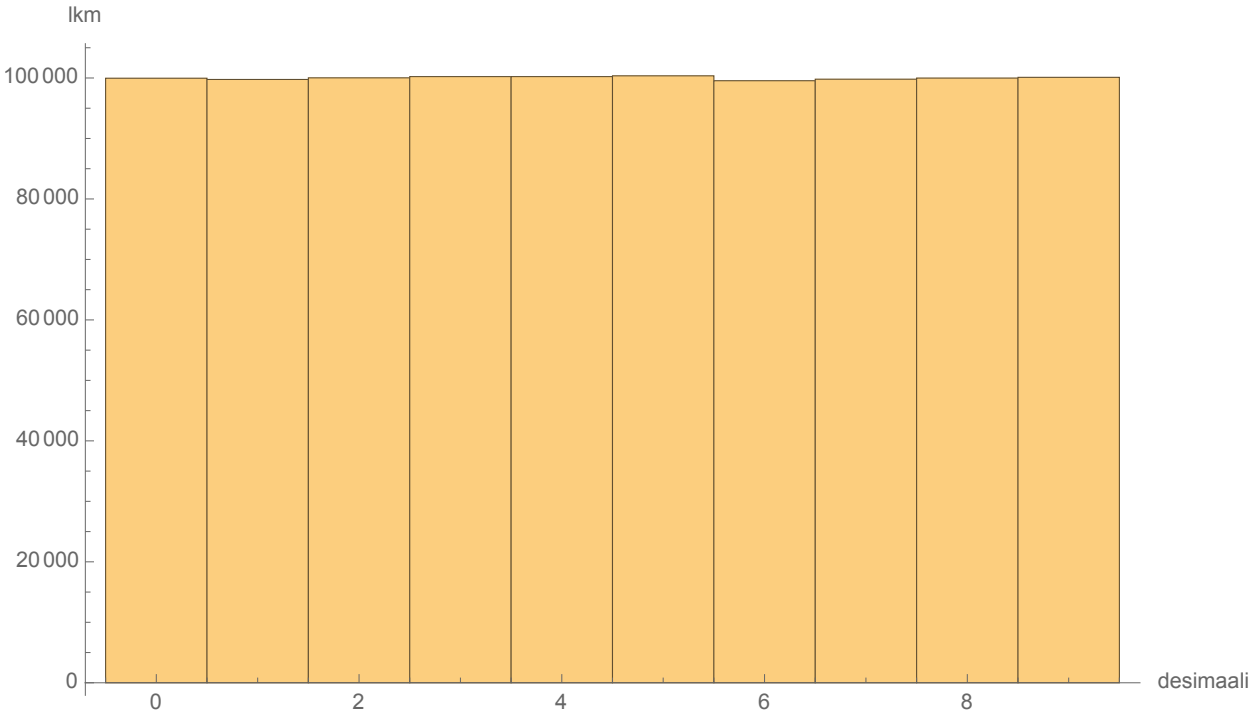
Tiedosto	Kompressoimaton	zip (fast)	lzip (slow)
Iliad ja Odyssey	525 670 (100%)	215 816 (41%)	173 420 (33%)
Moby Dick	1 276 201 (100%)	534 266 (42%)	417 647 (33%)
Miljoona π:n desimaalia	1 000 002 (100%)	475 977 (48%)	434 908 (43%)
Miljoona satunnaista merkkiä	1 000 000 (100%)	838 829 (84%)	838 996 (84%)

Huomiota:

- lzip on parempi pakkausalgoritmi tekstille.
- Parempi algoritmi pystyy pakkaamaan “aitoa” kieltä (tässä Iliad ja Odyssey sekä Moby Dick) tehokkaammin (esim. Moby Dickin kohdalle algoritmin vaihto zip $\rightarrow$ lzip pienentää vielä merkittävästi tiedoston kokoa), kun taas piin desimaalien kohdalla ero on jo pienempi ja satunnaismerkkien kohdalla olematon.
- Satunnaismerkkijonoa on vaikeampi pakata kuin kieltä, sillä puhutuissa kielissä eri kirjainten esiintymistodennäköisyys ei ole tasajakauma, kun taas satunnaismerkkijonoissa todennäköisyysjakauma on (tai sen ainakin pitäisi olla) täsmälleen tasajakauma

Piin desimaalit ovat pakkausalgoritmeille (käytännössä) satunnaisia. Tämä johtuu siitä, että piin kymmenkantaiset desimaalit muodostavat lähes tasajakauman (kuva 1). Kuitenkin kun piin desimaalit esitetään ASCII-tekstinä, jokaiseen merkkiin on käytetty 8 bittiä. Piin desimaalin tapauksessa (desimaalierotinta lukuunottamatta) tämä on haaskausta, koska koko ASCII-merkistön sijaan tarvitaan vain numerot 0–9. Tämän esittämiseen tarvitaan vain 4 bittiä tietoa, joten tällaisenkin naiivin pakkausalgoritmin kompressiosuhteen tulisi lähestyä lukua 1/2. Informaatioteorian näkökulmasta

desimaalien esittämiseen vaaditaan $h = -\log_2(1/10) \approx 3.3$ bit, joten kehittyneemmät algoritmit voivat pakata piin desimaalit vielä tehokkaammin. Teoreettisella rajalla tiedostonkoko on pienimmillään $S = Nh \approx 415$ kilotavua. On huomattavaa, että lzip pääsee jo melko lähelle tätä rajaa.



Kuva 1: Miljoonan piin desimaalin jakauma on lähes tasajakauma. Täsmällisesti tätä ei ole vielä kyetty osoittamaan, sillä ei tiedetä, onko pii normaali luku vai ei.

Tehtävä 6

Koodaa Quine.

Ratkaisu:

Alla esitettävä ohjelma on koodattu Swift–kielellä ja testattu versioilla 5.0.1 ja 5.1.2. Alle kirjoitettu ohjelma on kommentoitu versio eikä siis lopullinen vastaus (ohjelma ei tulosta kommentteja tai rivinvaihtoja). Oikean ohjelman voi ajaa osoitteesta `https://repl.it/repls/ChillyWanTranslations`.

```
// --- KOODI ALKAA ---
import Foundation // Tuodaan standard library, joka sisältää esim. Unicode-merkistön.

let a = "105 109 112 111 114 116 32 70 111 117 110 100 97 116 105 111 110 59 108
         101 116 32 97 32 61 32 34 35 34 59 108 101 116 32 98 32 61 32 85 110 105
         99 111 100 101 83 99 97 108 97 114 40 51 53 41 33 59 108 101 116 32 120
         32 61 32 97 46 99 111 109 112 111 110 101 110 116 115 40 115 101 112 97
         114 97 116 101 100 66 121 58 32 34 32 34 41 46 109 97 112 32 123 32 83
         116 114 105 110 103 40 67 104 97 114 97 99 116 101 114 40 85 110 105 99
         111 100 101 83 99 97 108 97 114 40 73 110 116 40 36 48 41 33 41 33 41 41
         32 125 46 106 111 105 110 101 100 40 41 59 112 114 105 110 116 40 120 46
         114 101 112 108 97 99 105 110 103 79 99 99 117 114 114 101 110 99 101 115
         40 111 102 58 32 83 116 114 105 110 103 40 98 41 44 32 119 105 116 104 58
         32 97 41 41 10" // Osa lähdekoodista Unicode-merkkien desimaaleina,
                            välilyönneillä eroteltuna.

let b = UnicodeScalar(35)! // Antaa merkin #, kirjoitettu Unicode-merkkinä koska
                              muuten myöhemmät vaiheet korvaisivat myös tämän merkin,
                              mitä ei tietenkään haluta. Huutomerkki lopussa on
                              Swiftin ominaisuus jolla ei ole tässä väliä.

let x = a.components(separatedBy: " ") // Muuntaa stringin listaksi muotoon
                                          ["105", "109", ...].
    .map { // Käy jokaisen listan alkion läpi ja suorittaa sille tämän blokin.
              $0 tarkoittaa senhetkistä alkiota eli ensimmäisellä
              iteraatiolla $0 = "105", toisella $0 = "109" jne.

        String(Character(UnicodeScalar(Int($0)!)!)) // Muunnetaan Unicode-numero $0
                                                       stringiksi. Huutomerkit voi
                                                       jälleen jättää huomiotta.

}.joined() // Tällä hetkellä x on muotoa ["i", "m", ...], joten laitetaan kaikki
              kirjainalkiot yhteen pötköön muotoon "im...".
```

```
print( // Tulostetaan stdoutiin.

    // Alla oleva rivi korvaa stringin x kaikki #-merkit muuttujalla a. Näin saadaan
       lopulliseen tulostukseen muuttuja a määriteltyä kuten se tässä koodissa on ilman
       että a:n määritelmää pitäisi ottaa huomioon a:n alkuperäisessä määritelmästä.
       Itse asiassa koko koodipätkän idea on, että muuttujaa a voi muuttaa täysin
       erillään muusta koodista. Muu koodi oikeastaan vain dekoodaa a:n normaali-
       aakkosille ja kikkailee a:n kohdalle takaisin alkuperäisen määritelmän.
    x.replacingOccurrences(of: String(b), with: a)
)
// --- KOODI LOPPUU ---
```

Tämä ohjelma on siis itse kirjoitettu monen tunnin pohdinnan jälkeen. Ei toki ole yllättävää, että internetistä löytyy paljon yksinkertaisempia ohjelmia:

```
let s=";print(\"let s=\"+s.debugDescription+s)";
print("let s="+s.debugDescription+s)
```

Tämä ei kuitenkaan ole `codegolf.stackexchange.com`, joten koodin pituudella ei ole väliä :)