Tuukka Kurtti
tuukkak123@gmail.com
Oulun yliopisto
28.12.2014

1. Yleistä

Yksi haasteellisimpia asioita rahasumman jako mielivaltaisiin seteleihin. Ensisilmäyksellä voi tuntua, että asia on perin yksinkertainen, mutta ongelmia tulee eteen nopeasti. Lopullinen ratkaisu menee aika pitkälle lukuteoriaan. Kehittelin useita omia termejä, ja on mahdollista, että samoista asioista on olemassa virallinen termi. En ala tässä dokumentoinnissa todistamaan, että algoritmit toimivat jokaisessa tilanteessa.

2. Suurin yhteinen tekijä

Rahanlajittelualgoritmi alkaa seteleiden suurimman yhteinen tekijän etsimisellä heti ohjelman käynnistyessä. Jos käyttäjän syöttämä rahasumma ei ole jaollinen setelien suurimmalla yhteisellä tekijällä, ei sitä voida antaa tasan, vaan se täytyy pyöristää alaspäin. Esimerkiksi jos kaikki setelit ovat jaollisia kymmenellä (eli niiden syt on 10), ei summaa 55 voida antaa, vaan se pyöristetään alaspäin lukuun 50, joka on jaollinen suurimmalla yhteisellä tekijällä.

Kun setelien suurin yhteinen tekijä tiedetään, voidaan setelien summat jakaa sillä, jolloin päästää tilanteeseen, että setelien syt on yksi. Kulissien takana ohjelma käsittelee seteleitä tässä muodossa, jolloin tilanne yksinkertaistuu. Tällöin voidaan hyödyntää algoritmeja, jotka vaativat lukujen olevan jaottomia keskenään kuten frobeniuksen luvun tapauksessa. Seteleiden syt tallennetaan muistiin, jolloin kaikki käyttäjältä kysytyissä ja käyttäjälle tulostettavissa lukuarvoissa voidaan tämä muutos ottaa huomioon.

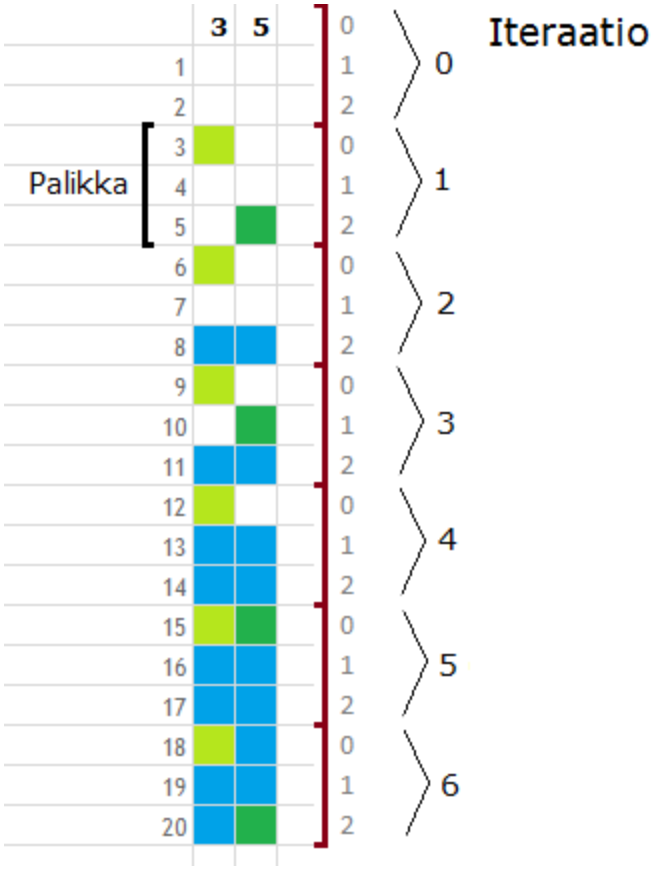
Kaikkien seteleiden keskinäinen suurin yhteinen tekijä haetaan algoritmilla vertailemalla jokaista vierekkäistä seteliä laskemalla niiden kahdenkeskisen suurimman yhteisen tekijän arvo. Näin saadaan yhteensä n-1 syt:in arvoa, joista pienin kuvaa kaikkien seteleiden suurinta yhteistä tekijää (n on erilaisten setelien lukumäärä).

3. Frobeniuksen luku

Ehkä haasteellisin osa seteleihin jakoa on frobeniuksen lukuun liittyvät algoritmit. Frobeniuksen luku on suurin luku, jota ei voida esittää taulukon arvojen positiivisten kokonaislukukertoimien avulla. Itse frobeniuksen luku ei ole rahanjakoalgoritmin kannalta olennaisin osa, vaan algoritmiin liittyvän taulukon, jota kutsun palikaksi, avulla pystytään

määrittelemään, voiko minkä tahansa summan jakaa tasan seteleihin. Palikan avulla voidaan määrittää helposti se luku, joka on lähimpänä haluttua lukua, jos haluttu luku ei mene tasan.

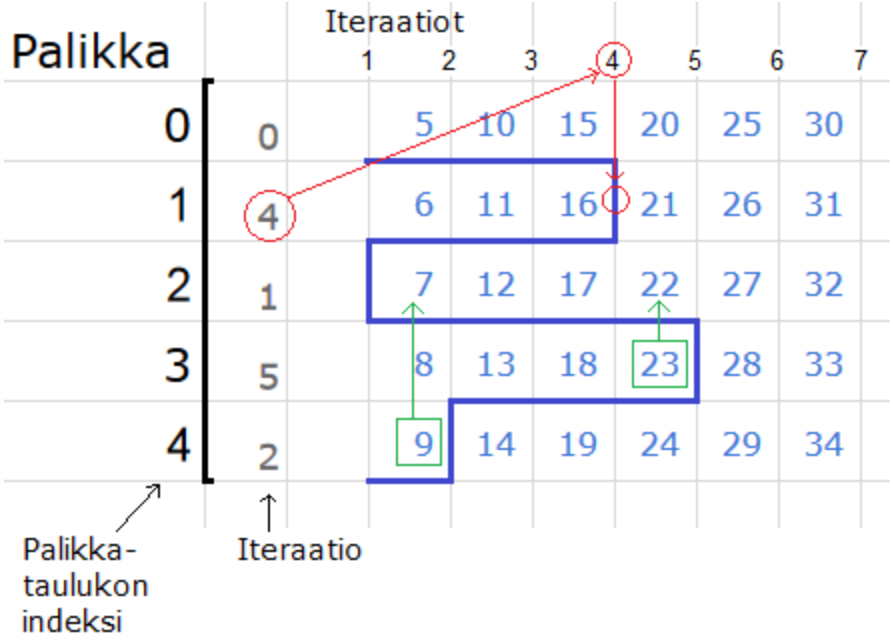
Tarkastellaan lukuja 3 ja 5:



Kuvaan on vaalean vihreällä merkitty arvot, jotka voidaan jakaa tasan kolmella, ja tummalla vihreällä arvot, jotka voidaan jakaa tasan viidellä. Sinisellä on merkitty arvot, jotka voidaan ilmaista yhdistelemällä lukuja 3 ja 5. Lukuja 1, 2, 4 ja 7 ei pystytä esittämään lukujen 3 ja 5 summakombinaationa. Näistä frobeniuksen luku on siis suurin eli 7.

Kuvasta näkee, että palikan indeksit nolla ja kaksi täyttyy iteraatiossa 1, indeksi yksi taas vasta iteraatiossa 3. Tarkalleen ottaen indeksi nolla täyttyy aina iteraatiossa nolla, koska nolla voidaan antaa aina tyhjänä summana.

Tarkastellaan palikan tarkoitusta kuvan avulla. Alla kuvataan palikkaa, joka muodostuu setelien ollessa luvut 5 ja 7, joista pohjaksi on valittu luku 5, koska se on pienin.



Sinisen viivan vasemmalla puolella olevia lukuja ei voi esittää lukujen 5 ja 7 summakombinaatioina. Sininen viiva muodostuu palikassa olevien täyttymisiteraatioden arvojen perusteella. (luvut 0-4 eivät näy kuvassa, mutta näistä aina ainoastaan nolla voidaan esittää summana, ja sekin on triviaalitapaus.)

Jos käyttäjä syöttää haluamakseen summaksi luvun, jota ei voida jakaa setelien summaksi, käydään palikkaa läpi ylöspäin, kunnes päästään lukuun, joka menee tasan seteleiden summaksi. Esimerkiksi kuvan tapauksessa käyttäjän syöttäessä summaksi 9 tippuu annettujen setelien summa lukuun 7 tai 23 lukuun 22 (merkitty kuvassa vihreällä). Jokainen luku voidaan jakaa summaksi viimeistään palikan indeksillä nolla.

Palikan täyttöalgoritmi suoritetaan etukäteen ohjelman käynnistyksen yhteydessä. Palikka-algoritmin alussa valitaan pohjaksi (eli palikan kooksi) arvo, jonka kannattaa olla pienin setelien arvoista, koska algoritmi on silloin nopein. Palikka täytetään iterointiin ja jakoyhtälöon perustuvan algoritmin avulla. Tilanne menee hieman monimutkaisemmaksi silloin, kun seteleitä on enemmän kuin kaksi. Tarkempi selitys ja esimerkki palikan täyttymisestä löytyy kohdasta 5, joka sisältää esimerkin koko algoritmin toiminnasta.

4. Seteleihin jako

Käyttäjän antaessa nostettavan summan, summa muutetaan ensin muotoon, joka pystytään jakamaan seteleiksi tasan. Tässä käytetään apuna sekä frobenius-palikkaa ja lukujen suurinta yhteistä tekijää.

Tämän jälkeen viimeinen vaihe on summan jako seteleihin, joka täytyy myös suorittaa jokaiselle käyttäjän syöttämälle summalle erikseen. Frobenius-palikka tarkastuksen avulla tiedetään lähin summa, joka pystytään antamaan käyttäjälle. Algoritmi vähentää summasta niin monta suurinta seteliä kuin siihen mahtuu ja sitten yrittää etsiä yhdistelmä muita seteleitä, jotka jakavat jäljelle jääneen summan tasan. Jos jäljellä oleva summa ei jakaannu tasan, vähennetään suurimpien setelien lukumäärästä yksi ja tutkitaan sitten jäljellä olevaa suurempaa summaa. Eri yhdistelmiä käydään näin läpi, kunnes löytyy yhdistelmä, joka jakautuu seteliksi tasan. Tämä toteutus vaatii n kappaletta sisäkkäistä silmukkaa, joka on toteutettu rekursiivisesti itseään kutsuvan funktion avulla.

Seteliyhdistelmiä käydään läpi sellaisessa järjestyksessä, että kun oikea kombinaatio löytyy, suuria seteleitä annetaan mahdollisimman paljon. Joissain harvoissa tilanteissa tämä ei välttämättä ole paras ratkaisu. Esimerkiksi automaatti antaisi summan 1998 euroa tällä tavalla:

1 kpl 1000 euron seteleitä.
998 kpl 1 euron seteleitä.
YHTEENSÄ 1998 euroa.

Parempi tapa olisi antaa se näin:

2 kpl 999 euron seteleitä.
YHTEENSÄ 1998 euroa.

Automaatti ei priorisoi setelien lukumäärää mitenkään, ja antaisi ensimmäisen vaihtoehdon siksi, kun se sisältää suuremman setelin.

5. Esimerkki algoritmin toiminnasta kokonaisuudessaan

Halutaan, että pankkiautomaatti voi antaa asiakkaille 30, 45 ja 100 euron seteleitä. Ensimmäinen vaihe on suurimman yhteisen tekijän etsiminen. Tämä tehdään vertailemalla viereisten lukujen suurimpia yhteisiä tekijöitä.

syt(30, 45) = 15
syt(45, 100) = 5

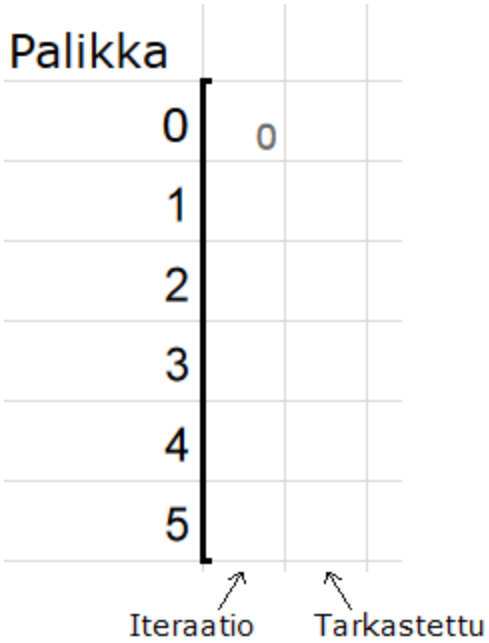
Näistä arvoista luku 5 on pienempi, joten tämä on kolmen setelin suurin yhteinen tekijä. Nyt setelien arvot voidaan yksinkertaistaa seuraavaan muotoon:

30 / 5 = 6
45 / 5 = 9
100 / 5 = 20

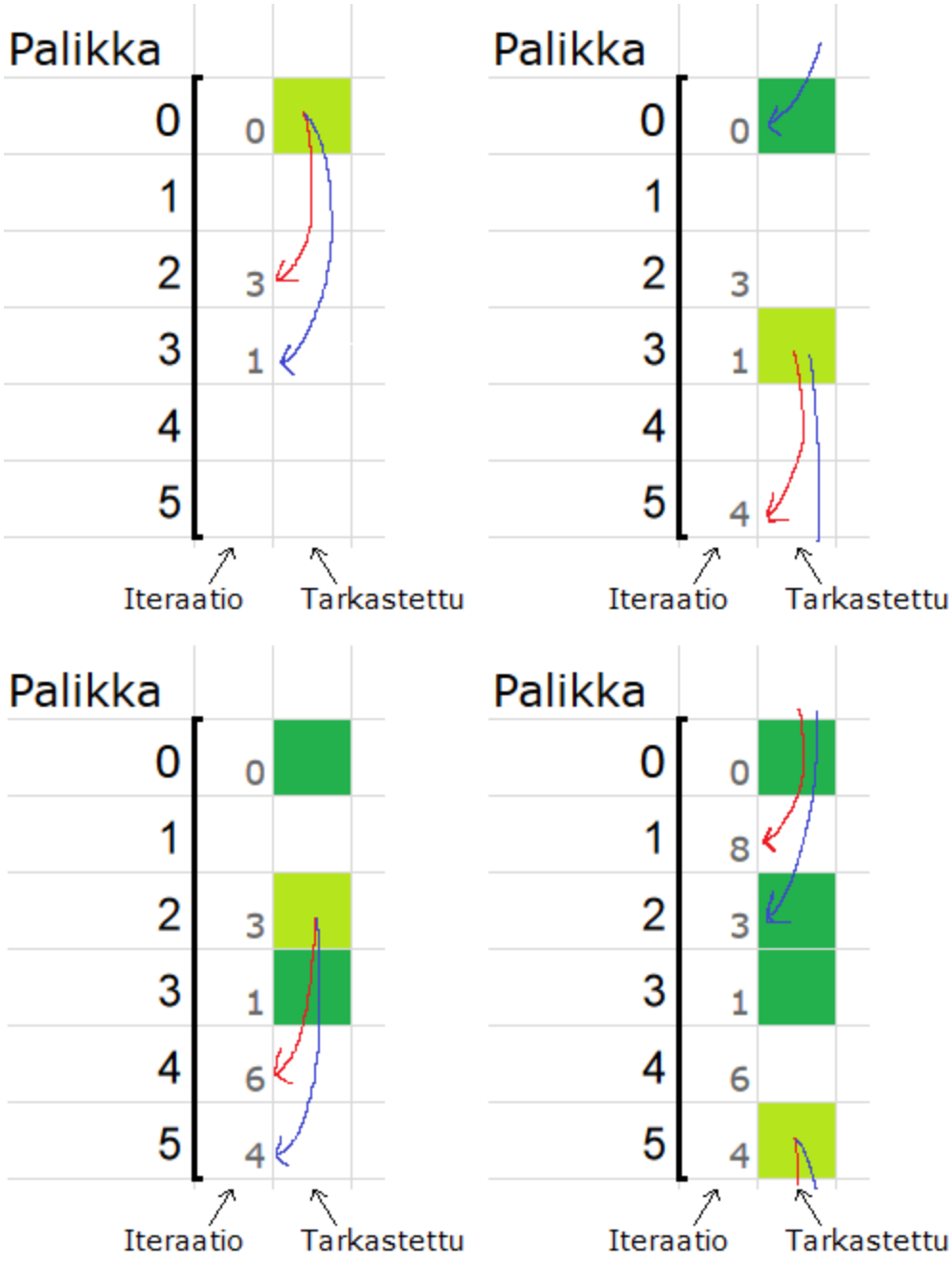
Nyt setelien luvut ovat jaottomia keskenään, joten seuraava vaihe on frobenius-palikan kokoaminen. Valitaan pohjaksi pienin luku eli 6. Tämän jälkeen lasketaan muille luvuille **askel** ja **väli**, jotka saadaan lukujen jakoyhtälöstä pohjan suhteen:

9 = **1** * 6 + **3**
20 = **3** * 6 + **2**

Askel ja väli ovat ainoat tiedot, mitä luvuista tarvitaan seuraavassa vaiheessa, eli frobenius-palikan täyttämisessä. Aluksi alustetaan palikka tyhjäksi ja sijoitetaan ensimmäiseen alkioon palikka[0] täyttymisiteraatioksi luku nolla. Palikan täyttämistä varten tallennetaan jokaista alkiota varten myös totuusarvo, joka ilmoittaa onko kyseistä kohtaa tarkastettu:



Tämän jälkeen toistetaan palikantäyttöalgoritmia, kunnes palikka on valmis. Alussa etsitään palikasta pienin tarkastamaton iteraatio ja merkitään se tarkastetuksi. Tästä palikan kohdasta siirrytään eteenpäin askeleen verran jokaista muuta lukua kohden (kuvassa sininen 9 ja punainen 20), ja kirjoitetaan sinne välin verran suurempi iteraation arvo, jos vanhaa arvoa ei ole tai se on suurempi. Tätä jatketaan, kunnes kaikki palikan kohdat on tarkistettu.



Kohdat 1 ja 4 on vielä tarkastamatta, mutta palikka on jo täynnä eikä nämä tarkastukset muuta sen arvoja. Jos hypätään palikan lopusta takaisin alkuun, tallennetaan iteraation kohdalle yhtä normaalia korkeampi arvo. Kuvassa 4 kohtaan 1 kirjoitetaan iteraation arvo 8 (punainen nuoli), joka muodostuu seuraavien lukujen summana: 4(alkuperäinen iteraation arvo kohdassa 5), 3(väli) ja 1(lisäys, koska hypättiin palikan alkuun). Nyt frobenius-palikka on täytetty ja rahanlajittelijan alustus on valmis.

Oletetaan, että käyttäjä haluaa nostaa summan 173 euroa. Aluksi tämä summa jaetaan setelien suurimmalla yhteisellä tekijällä ja pyöristetään alaspäin:

173 / 5 = 34,6 ~~ 34

Seuraavaksi tutkitaan, jakautuuko summa tasan lukuihin 6, 9 ja 20. Tämäkin onnistuu jakoyhtälön avulla:

34 = **5** * 6 + 4

Luku 34 sijaitsee siis palikan indeksiä **4** vastaavalla rivillä ja iteraatioiden **5** ja 6 välissä (eli viidennellä sarakkeella), sivulla 3 olevan kuvan kaltaisessa tilanteessa.

Koska palikka[**4**] on 6, mikä on suurempi kuin **5**, ei lukua 34 voida esittää tasan summana. Tutkitaan siis edellistä lukua.
Koska palikka[3] on 1, mikä on pienempi tai yhtäsuuri kuin **5**, luku 33 voidaan esittää tasan summana. Viimeinen vaihe on etsiä rekursiivisesti oikea yhdistelmä luvun 33 esittämiseen:

20 menee kerran lukuun 33, jäljelle jää summa 33 - 20 = 13
    9 menee kerran lukuun 13, jäljelle jää summa 4
        6 menee 0 kertaa lukuun 4, mutta 0 * 6 ei ole 4, palataan siis taaksepäin
    9 menee nyt 0 kertaa lukuun 13, jäljelle jää summa 13
        6 menee 2 kertaa lukuun 13, mutta 2 * 6 ei ole 13, palataan siis taaksepäin
    9 menisi nyt -1 kertaa lukuun 13, mikä ei ole mahdollista, palataan siis taaksepäin
20 menee nyt **0 kertaa** lukuun 33, jäljelle jää summa 33
    9 menee **3 kertaa** lukuun 33, jäljelle jää summa 6
        6 menee **kerran** lukuun 6, 1 * 6 on 6, haluttu yhdistelmä on siis löytynyt

Kun summa menee syvimmällä tasolla tasan, löytyy setelien määrät viimeisimmästä kohtaa sen omaa tasoa, jotka on kuvattu eri sisennysasteilla. (20 ei sisennetty, 9 yksi sisennys, 6 kaksi)
Käyttäjälle tulostettavat summat on vielä kerrottava suurimmalla yhteisellä tekijällä:

1 kpl 30 euron seteleitä.
3 kpl 45 euron seteleitä.
YHTEENSÄ: 165 euroa.