Kotitehtävä 2

Euroopan unionin yhteisvaluutta euro on ollut tilivaluuttana käytössä jo jonkin aikaa, mutta eurokolikot ja setelit tulevat käyttöön vuoden 2002 alussa. Käyttöön tulevat 1 ja 2 euron kolikot sekä 5, 10, 20, 50, 100, 200 ja 500 euron setelit.

Eräs hyvyyskriteeri rahojen nimellisarvovalikoimalle on, millaisia rahasummia voidaan maksaa siten, että vain pieni määrä rahoja (kolikoita tai seteleitä) vaihtaa omistajaa. Esimerkiksi 46 euroa voidaan maksaa 50 euron setelillä ja saada kaksi 2 euron kolikkoa takaisin. Se voidaan myös maksaa 50 euron setelillä ja 1 euron kolikolla ja saada 5 euron kolikko takaisin.

$L(k, m)$ on suurin L, jolle voidaan löytää k nimellisarvon nimellisarvovalikoima $W=\{w_1, \ldots, w_k\}$, jolla jokainen rahasumma $1, 2, 3, \ldots, L$ voidaan maksaa niin, että enintään m rahaa vaihtaa omistajaa. Toisin sanoen, $L(k, m)$ on suurin sellainen L, että löytyy nimellisarvovalikoima, jolla jokainen $1, \ldots, L$ voidaan kirjoittaa muodossa $\sum_{i=1}^{k} m_i w_i$, missä $\sum_{i=1}^{k} |m_i| \leq m$. (kaikki kokonaislukuja)

- Mikä on pienin rahasumma, jonka maksamiseen eurorahoilla ei riitä, että kolme rahaa vaihtaa omistajaa?
- Millainen nimellisarvovalikoima Euroopan unionissa olisi tullut ottaa käyttöön? Laadi (ei-triviaaliin) heuristiikkaan perustuva tietokoneohjelma, joka etsii mahdollisimman hyvän alarajan ja vastaavan nimellisarvovalikoiman $L(9, 3)$:lle. Tarkastele myös muita tapauksia $L(k, 3)$. Voit tutkia, löytääkö ohjelmasi $L(k, m)$:n tunnettuja arvoja $L(1, 3) = 3$, $L(2, 3) = 12$, $L(3, 3) = 24$, $L(4, 3) = 47$, $L(5, 3) = 73$ ja $L(6, 3) = 110$ vastaavia nimellisarvovalikoimia.

Erityisen ansiokkaista ratkaisuista voi saada bonuspisteen tenttiin. Bonuspisteen tavoittelemiseksi voi esimerkiksi esittää erinomaisen tehokkaan heuristiikan ja löytää tarvittavaan laskenta-aikaan suhteutettuna erinomaisia alarajoja $L(k, m)$:lle eri k:n ja m:n arvoilla tai vertailla ainakin kahden toisistaan poikkeavan algoritmin tehokkuutta.

Palautukseen tulee sisältyä ratkaisun sanallinen kuvaus, ohjelmien lähdekoodit ja kuvaus löydetyistä $L(k, m)$:n alarajoista vastaavine nimellisarvojoukkoineen sekä tarvitusta laskenta-ajasta.

Palautus: Palautus assistentille (Petteri Kaski, Petteri.Kaski@hut.fi, TB358). Kotitehtävän voi palauttaa myös sähköpostitse esim. tekstinä tai hyvälaatuisena Postscriptinä. Viimeinen palautuspäivä on 22.3.2000.

Vihjeitä ja ohjeita

Nimellisarvovalikoiman hyvyyttä arvioitaessa ilmeisin valinta kohdefunktioksi olisi suurin L, jolla kaikki summat $1, \ldots, L$ voidaan maksaa m rahalla. Saattaa olla tehokkaampaa muotoilla ongelma optimointiongelman sijaan hakuongelmana: kiinnitetään ensin L, ja koetetaan löytää arvovalikoima, jolla kaikki summat $1, \ldots, L$ voidaan maksaa. Tällöin kohdefunktio koostuisi vain sakkotermeistä ja se voisi olla yksinkertaisesti peittämättömien summien lukumäärä. Kohdefunktio voi toki olla aivan muunlainenkin.

Alla annetaan rutiini, joka laskee, mitkä rahasummat voidaan maksaa annetulla nimellisarvovalikoimalla. Voitte halutessanne käyttää sitä apuna kohdefunktiota muotoillessanne. Aliohjelman parametrit kuvataan seuraavassa.

arvot on taulukko, jonka alkiot $0, \ldots, k-1$ ovat nimellisarvot.

k on nimellisarvojen lukumäärä.

m on omistajaa vaihtavien rahojen suurin sallittu määrä.

i on muuttuja, jossa pidetään tähän asti kiinnitettyjen m_i-muuttujien määrää. Aliohjelmaa ulkopuolelta kutsuttaessa tämän parametrin tulee olla 0.

curr on muuttuja, jossa aliohjelma pitää tähän asti kiinnitettyjen rahojen summan. Käyttäjän kutsuessa aliohjelmaa parametrin arvon tulee olla 0.

peitetty on globaali taulukko, johon lasketaan, kuinka monella tavalla kukin summa voidaan maksaa.

maxl on peitetty-taulukon viimeisen alkion indeksi.

Aliohjelmakutsun jälkeen **peitetty[i]** kertoo, monellako tavalla i euroa voidaan maksaa käytettävissä olevalla nimellisarvovalikoimalla.

Algorithm 1 peitto.py

```
def peitto(arvot, k, m, i, curr):
    if i==0:
        x = 1
        while x <= maxl:
            peitetty[x]=0
            x=x+1
    if i==k:
        if curr>0 and curr <= maxl:
            peitetty[curr]=peitetty[curr]+1
    else:
        x = -m
        while x <= m:
            peitto(arvot, k, m-abs(x), i+1, curr+x*arvot[i])
            x=x+1
```