IOI 1999 Olympiavalmennustehtäviä

Antti Vähäkangas

Yleistä tehtävistä

Useisiin tehtäviin on monia eri ratkaisujamalleja. Pyri kehittelemään vaihtoehtoisia tekniikoita ensimmäiselle keksimällesi ratkaisulle.

Jotkin ongelmat voidaan palauttaa suoraan tunnetuiksi ongelmiksi (tai niiden erikoistapauksiksi), joihin on tästä syystä olemassa valmiita ratkaisualgoritmeja (koskee erityisesti *verkkoteoreettisia* ongelmia).

Eräät ongelmat ovat luonnostaan vaikeita — niille ei välttämättä kannata etsiä polynomiallista ratkaisualgoritmia. Näille ongelmille on tyypillisesti olemassa työlääseen hakuun perustuva ratkaisu, jota voidaan parantaa oleellisesti esimerkiksi organisoimalla hakuavaruus järkevästi tai taulukoimalla osaratkaisuja.

Oman luokkansa muodostavat myös ongelmat, joiden ratkaisualgoritmi on todella yksinkertainen (usein jonkinlainen lineaarinen algoritmi), mutta ongelman muodostaa tarvittavan tekniikan keksiminen. Tyypillisesti tarvittava tekniikka on myös varsin yksinkertainen. Tekniikan keksiminen voi kuitenkin olla yllättävän vaikeaa. Näille ongelmille on myös usein olemassa ilmeisiä, mutta kovin tehottomia ratkaisuja, jotka perustuvat esimerkiksi koko hakuavaruuden läpikäymiseen. **IOI:n ensimmäisen päivän I tehtävä on usein kuulunut juuri tähän kategoriaan.**

Tehtäviin liittyvien parametrien arvoalueita ja alkioiden lukumääriä on aina rajoitettu. Raja-arvoista voi päätellä, kuinka tehokasta algoritmia kulloinkin kannattaa etsiä.

1. Usean kasan Nim

Pelissä on N $(1 \leq N \leq 1000)$ kasaa, kussakin kasassa $a * k$ $(1 \leq a, k \leq 25)$ tikkua. Pelaajat ottavat $(1 \cdots k-1)$ tikkua vuorotellen valitsemastaan kasasta (kuitenkin enintään niin monta tikkua, kuin kasassa on jäljellä). *Voittaja* on se, joka ottaa viimeisen tikun viimeisestä jäljelläolevasta kasasta.

Onko toisella pelaajista voittostrategia (jos on, mikä)?

Variantti:

- Tikkujen lukumäärä kasoissa on $a * k + 1$ ja viimeisen tikun ottaja *häviää* pelin.

2. Turnauksen aikataulu

Turnaukseen osallistuu $N = 2^k$ $(1 \leq k \leq 16)$ pelaajaa. Kullekin pelaajalle on määritelty, montako peliä pelaaja voi *maksimissaan* pelata päivässä. Tehtävänä on löytää pelaajille aikataulu, joka täyttää ehdot:

- Jokainen pelaaja pelaa jokaista vastustajaansa vastaan *kerran* (yhteensä siis $N(N-1)/2$ peliä pelataan).
- Yksikään pelaaja ei ylitä pelikapasiteettiaan päivää kohden.
- Pelipäivien lukumäärän on oltava *pienin mahdollinen.*

Anna tehokas algoritmi, joka etsii turnauksen aikataulun.

3. **Luvun ositus**

Annettu kokonaisluku N ($1 \leq N \leq 50$). Anna tehokas algoritmi, joka laskee, kuinka monella tavalla N voidaan osittaa nollaa suuremmiksi kokonaisluvuiksi. T.s. ratkaisee, monellako tavalla voidaan valita jono

$$N = x_1 + \cdots + x_n \mid x_i \in Z^+$$

jossa jäsenien järjestyksellä ei ole merkitystä.
Variantti:

- Jäsenien järjestys on erotteleva ominaisuus.

4. **Lautanpakkaus (Datatähti '94 loppukilpailutehtävä)**

Annettu lautta jonka syvyys on d ($1 \leq d \leq 25$) ja jossa on w ($1 \leq w \leq 4$) kaistaa. Annettu myös autot ($a_1, \cdots, a_n \mid 1 \leq n \leq 100$) (tai oikeammin niiden pituudet) tietyssä järjestyksessä, jota ei saa muuttaa. Tehtävänä on määrittää montako autoa voidaan pakata lauttaan s.e.

- Minkään kaistan kapasiteetti (d) ei ylity. (Kaistalle i pakattujen autojen kustannus on niiden yhteenlaskettu pituus.)
- Jos auto a_i on pakattu lauttaan, myös autot $a_1, \cdots, a_{i-1}$ on pakattu lauttaan.

Anna (mahdollisimman) tehokas algoritmi, joka ratkaisee lautanpakkausongelman.

5. **Dominojono**

Dominolaatta on järjestämätön pari ($a, b \mid 1 \leq a, b \leq 20000$). Kaksi dominolaattaa voidaan *asettaa peräkkäin* joss niiden liittymäkohdan vastakkaiset luvut ovat samat. Esimerkiksi dominolaatat $(2, 4)$ ja $(2, 6)$ voidaan asettaa peräkkäin kahdella eri tavalla:

- $(4, 2)$, $(2, 6)$
- $(6, 2)$, $(2, 4)$.

Anna tehokas algoritmi, joka ratkaisee voidaanko annetut N ($1 \leq N \leq 1000$) dominolaattaa asettaa peräkkäin jonoon.
Variantteja:

- Dominojonon etsiminen.
- Eri dominojonojen lukumäärän laskeminen.

6. **Kohtaaminen**

Annettu tasoverkko $G = (V, E)$, kaksi sen solmua (u ja v) sekä kustannusfunktio $w : E \mapsto \{1, \cdots, 10\}$. Tehtävänä on etsiä G:stä yksinkertainen polku ($u \mapsto c \mapsto v \mid c \in V$) s.e. osapolun ($u \mapsto c$) kustannus on sama kuin osapolun ($c \mapsto v$) kustannus. (Jos polulla on kaaret ($e_1, \cdots, e_n$) niin sen kustannus $= \sum_{i=1}^{n} w(e_i)$.)

(Verkko on tasoverkko, jos se voidaan piirtää tasolle ilman että yksikään kaaripari leikkaa toisensa. Yksinkertaisella polulla kukin solmu esiintyy maksimissaan yhden kerran.) Anna (mahdollisimman) tehokas algoritmi tällaisen polun etsimiseen. Voit olettaa, että solmuja on verkossa korkeintaan 200 kpl.

7. **Ornamentti**

Ornamentti koostuu pienistä *palloista* sekä palloja yhdistävistä *naruista*. Kuhunkin palloon liittyy yksi tai useampi naru ja yksi naru yhdistää aina kaksi palloa. Kunkin narun toinen pää on punainen ja toinen keltainen. Kullakin narulla on tietty pituus (kahdella eri narulla voi olla eri pituus). Ornamentti täytyy ripustaa seinälle seuraavia kohtia noudattaen:

- Yksi tai useampi pallo kiinnitetään naulalla seinään.
- Jokaisen narun punainen pää on ylempänä kuin keltainen pää.
- Jokainen naru on suorana ja täsmälleen pystysuorassa.
  (Useampi pallo voi olla samalla korkeudella rikkomatta edellistä ehtoa.)

Anna tehokas algoritmi, joka luettelee pallojen järjestyksen (ylhäältä alas) kun ornamentti on ripustettu seinälle. Voit olettaa, että ornamentti voidaan aina ripustaa seinälle (sääntöjä noudattaen) ja että palloja on korkeintaan 1000.

8. **2-SAT**

Annettu lausekalkyylin kaava *2-normaalimuodossa*. 2-normaalimuotoinen kaava on muotoa:

$$\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \cdots \wedge \alpha_n,$$

jossa kukin tekijä α_i on muotoa:

$$(x \vee y)$$

(x, y on muuttuja tai sen negaatio.) Esimerkki 2-normaalimuotoisesta lausekalkyylin kaavasta:

$$(x_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3) \wedge (x_1 \vee \overline{x}_3),$$

jonka muuttujilla on *toteuttava* (lausekkeen arvoksi tulee 1) totuusarvoasetus:

$$\begin{cases} x_1 = 1 \\ x_2 = 1 \\ x_3 = 1. \end{cases}$$

Tehtävänä on määrittää, onko muuttujilla kaavan toteuttavaa totuusarvoasetusta. Anna tehokas algoritmi kaavan toteutuvuuden testaamiseen. Voit olettaa, että muuttujia on korkeintaan 1000 kappaletta.

9. **Potenssiinkorotus**

Annettu kokonaisluku N ($1 \leq N \leq 100$). Käytössä on ohjelmointikieli, joka tukee *sijoitus-* ja *kertolasku*operaatioita. Tehtävänä on generoida *ohjelmateksti* ko. ohjelmointikielelle, joka laskee ($b = a^N$):n *minimimäärällä* kertolaskuja. Esimerkki generoidusta ohjelmatekstistä, kun $N = 13$

```
X1:=a;
X2:=X1*X1;
X3:=X2*X2;
X4:=X3*X1;
X5:=X3*X3;
X6:=X5*X4;
b:=X6;
```

Anna tehokas algoritmi, joka generoi ohjelmatekstin.

10. **Pisteitä suoralla**

Annettu joukko S ($1 \leq |S| \leq 100$) pisteitä 2D-avaruudessa. Kunkin pisteen paikan avaruudessa määrää (x, y)–koordinaattipari ($x, y \in N$). Voit olettaa, ettei millään kahdella pisteellä ole samaa x–koordinaattia.

Tehtävänä on etsiä sellaisen *maksimaalisen* osajoukon S' ($S' \subseteq S$) *koko*, jonka kaikki pisteet ovat *samalla suoralla*. Anna tehokas algoritmi $|S'|$:n etsimiseksi.

11. Viestiketju

Joukko H ($1 \leq |H| \leq 200$) henkilöitä muodostaa *verkoston* — jokainen henkilö valitsee *yhden* kohdehenkilön, jolle tämä välittömästi viestin saatuaan sen välittää. Muodostunut verkosto on *viestiketju*, joss mille tahansa henkilölle (verkoston ulkopuolelta) tullut viesti välittyy *jokaiselle* henkilölle verkostossa — myös viestin alunperin saaneelle henkilölle.

Syötteenä annetaan eräs verkosto, joka ei välttämättä ole viestiketju. Tehtävänä on laskea verkostoon tehtävien *muutosten minimilukumäärä*, jotta verkosto muodostaisi viestiketjun. Yksi muutos tarkoittaa yhden henkilön *kohdehenkilön vaihtoa*. Anna tehokas algoritmi tarvittavien muutosten minimimäärän laskemiseksi.

12. Lukujen jako

Annettu joukko S lukuja $(\forall i \in S)(1 \leq i \leq 200)$, $|S| \leq 100$. S jaetaan A:han ja B:hen siten, että

$$\begin{cases} A \cup B = S \\ A \cap B = \phi. \end{cases}$$

Tehtävänä on etsiä A ja B siten, että erotus

$$\left| \sum_{i=1}^{|A|} A_i - \sum_{j=1}^{|B|} B_j \right|$$

on *pienin mahdollinen*. Anna (mahdollisimman) tehokas algoritmi optimijaon etsimiseen.

13. Alkulukujono

Alkuluku p on p':s alkuluku (p' ei *välttämättä* ole alkuluku). Alkuluku p *muodostaa alkulukujonon* joss jono

$$p, p', (p')', \cdots, 1$$

on olemassa (koostuen siis vain alkuluvuista sekä luvusta 1).

Annettu luku N ($2 \leq N \leq 10000$). Tehtävänä on etsiä pienin p siten, että $p \geq N$ ja p muodostaa alkulukujonon. Anna tehokas algoritmi ongelman ratkaisemiseksi. (Alkuluku on luku, joka on jaollinen vain itsellään ja 1:llä; ensimmäiset alkuluvut ovat 2,3,5,7.)

Lähteitä

1. Ei erityistä lähdettä.
2. Alkuperäinen tehtävä (hieman yksinkertaisempana) teoksessa Mark Allen Weiss: Data Structures and Algorithm Analysis. s. 411.
3. Yleistä: http://www.astro.virginia.edu/ẽww6n/math/PartitionFunctionP.html.
4. Datatähti '94 loppukilpailutehtävä.
6. Testiaineistoja ja kuvaus: http://www.ioi99.org.tr/BOI/11/index.html.
7. Testiaineistoja ja kuvaus: http://www.ioi99.org.tr/BOI/22/index.html.
8. Tehtävä mainittu teoksessa Steven S. Skiena: The Algorithm Design Manual. s. 267.
9. IOI-1990 tehtävä 6. ftp://ftp.win.tue.nl/pub/ioi/ioi90/tasks90.txt.

10. Kuvaus: http://www.ifmo.ru/neerc/ProblemSetArchive/B_EU_AARC/1998/Lines.html.
11. Kuvaus: http://olympiads.win.tue.nl/ioi/misc/other.html: CEOI'95 Day 1 - problem B.
12. Kuvaus: http://olympiads.win.tue.nl/ioi/misc/other.html: CEOI'95 Day 2 - problem C.
13. Ei erityistä lähdettä.