

Kisakoodarin käsikirja

Antti Laaksonen

4. joulukuuta 2016

Sisältö

Alkusanat	ix
I Perusasiat	1
1 Johdanto	3
1.1 Ohjelmointikielet	3
1.2 Syöte ja tuloste	4
1.3 Lukujen käsittely	6
1.4 Koodin lyhentäminen	8
1.5 Virheen etsiminen	10
1.6 Matematiikka	11
2 Aikavaativuus	15
2.1 Laskusäännöt	15
2.2 Vaativuusluokkia	18
2.3 Tehokkuuden arviointi	19
2.4 Suurin alitaulukko	20
3 Järjestäminen	23
3.1 Järjestämisen teoriaa	23
3.2 Järjestäminen C++:ssa	27
3.3 Binäärihaku	29
4 Tietorakenteet	33
4.1 Dynaaminen taulukko	33
4.2 Joukkorakenne	35
4.3 Hakemisto	36
4.4 Iteraattorit ja välit	37
4.5 Muita tietorakenteita	39
4.6 Vertailu järjestämiseen	41
5 Täydellinen haku	43
5.1 Osajoukkojen läpikäynti	43
5.2 Permutaatioiden läpikäynti	45
5.3 Peruuttava haku	46
5.4 Haun optimointi	47
5.5 Puolivälihaku	49

6	Ahneet algoritmit	51
6.1	Kolikkotehtävä	51
6.2	Aikataulutus	52
6.3	Tehtävät ja deadlinet	54
6.4	Keskiluvut	55
6.5	Huffmanin koodaus	56
7	Dynaaminen ohjelmointi	59
7.1	Kolikkotehtävä	59
7.2	Pisin nouseva alijono	64
7.3	Reitinhaku ruudukossa	65
7.4	Repunpakkaus	66
7.5	Editointietäisyys	68
7.6	Laatoitukset	69
8	Tasoitettu analyysi	71
8.1	Kaksi osoitinta	71
8.2	Lähin pienempi edeltäjä	74
8.3	Liukuvan ikkunan minimi	75
9	Välikyselyt	77
9.1	Staattiset kyselyt	77
9.2	Binääri-indeksipuu	80
9.3	Segmenttipuu	83
9.4	Lisätekniikoita	87
10	Bittien käsittely	89
10.1	Luvun bittiesitys	89
10.2	Bittioperaatiot	90
10.3	Joukon bittiesitys	92
10.4	Dynaaminen ohjelmointi	94
II	Verkkoalgoritmit	97
11	Verkkojen perusteet	99
11.1	Käsitteitä	99
11.2	Verkko muistissa	103
12	Verkon läpikäynti	107
12.1	Syvyyshaku	107
12.2	Leveyshaku	109
12.3	Sovelluksia	111
13	Lyhimmät polut	113
13.1	Bellman-Fordin algoritmi	113
13.2	Dijkstran algoritmi	116
13.3	Floyd-Warshallin algoritmi	119

14 Puiden käsittely	123
14.1 Puun läpikäynti	124
14.2 Lämpimä	125
14.3 Solmujen etäisyydet	126
14.4 Binääripuut	128
15 Virittävät puut	129
15.1 Kruskalin algoritmi	130
15.2 Union-find-rakenne	132
15.3 Primin algoritmi	134
16 Suunnatut verkot	137
16.1 Topologinen järjestys	137
16.2 Dynaaminen ohjelmointi	139
16.3 Tehokas eteneminen	142
16.4 Syklin tunnistaminen	143
17 Vahvasti yhtenäisyys	145
17.1 Kosarajun algoritmi	146
17.2 2SAT-ongelma	148
18 Puukyselyt	151
18.1 Tehokas nouseminen	151
18.2 Solmutaulukko	152
18.3 Alin yhteinen esivanhempi	155
19 Polut ja kierrokset	159
19.1 Eulerin polku	159
19.2 Hamiltonin polku	163
19.3 De Bruijnin jono	165
19.4 Ratsun kierros	166
20 Virtauslaskenta	167
20.1 Ford-Fulkersonin algoritmi	168
20.2 Rinnakkaiset polut	172
20.3 Maksimiparitus	173
20.4 Polkupeitteet	176
III Lisäaiheita	179
21 Lukuteoria	181
21.1 Alkuluvut ja tekijät	181
21.2 Modulolaskenta	185
21.3 Yhtälönratkaisu	188
21.4 Muita tuloksia	189

22 Kombinatoriikka	193
22.1 Binomikerroin	194
22.2 Catalanin luvut	196
22.3 Inklusio-eksklusio	198
22.4 Burnsiden lemma	200
22.5 Cayleyn kaava	201
23 Matriisit	203
23.1 Laskutoimitukset	203
23.2 Lineaariset rekursioyhtälöt	206
23.3 Verkkojen käsittely	208
24 Todennäköisyys	211
24.1 Tapahtumat	212
24.2 Satunnaismuuttuja	214
24.3 Markovin ketju	216
24.4 Satunnaisalgoritmit	217
25 Peliteoria	221
25.1 Pelin tilat	221
25.2 Nim-peli	223
25.3 Sprague–Grundyn lause	224
26 Merkkijonoalgoritmit	229
26.1 Trie-rakenne	230
26.2 Merkkijonohajautus	231
26.3 Z-algoritmi	234
27 Neliöjuorialgoritmit	239
27.1 Eräkäsittely	240
27.2 Tapauskäsittely	241
27.3 Mo’n algoritmi	241
28 Lisää segmenttipuusta	243
28.1 Laiska eteneminen	244
28.2 Dynaaminen toteutus	247
28.3 Tietorakenteet	249
28.4 Kaksiulotteisuus	250
29 Geometria	253
29.1 Kompleksiluvut	254
29.2 Pisteet ja suorat	256
29.3 Monikulmion pinta-ala	259
29.4 Etäisyysmitat	260

30 Pyyhkäisyviiva	263
30.1 Janojen leikkauspisteet	264
30.2 Lähin pistepari	265
30.3 Konvekssi peite	266

Osa I

Perusasiat

Bittien käsittely

10.1 Luvun bittiesitys

89

```
int x = -43;
unsigned int y = x;
cout << x << "\n"; // -43
cout << y << "\n"; // 4294967253
```

etumerkillistä lukua $x = -43$ vastaa etumerkitön luku $y = 2^{32} - 43$.

Jos luvun suuruus menee käytössä olevan bittiesityksen ulkopuolelle, niin luku pyörii ympäri. Etumerkillisessä bittiesityksessä luvusta $2^{n-1} - 1$ seuraava luku on -2^{n-1} ja vastaavasti etumerkittömässä bittiesityksessä luvusta $2^n - 1$ seuraava luku on 0. Esimerkiksi koodissa

```
int x = 2147483647
cout << x << "\n"; // 2147483647
x++;
cout << x << "\n"; // -2147483648
```

muuttuja x pyörii ympäri luvusta $2^{31} - 1$ lukuun -2^{31} .

10.2 Bittiooperaatiot

And-operaatio

And-operaatio $x \& y$ tuottaa luvun, jossa on ykkösbitti niissä kohdissa, joissa molemmissa luvuissa x ja y on ykkösbitti. Esimerkiksi $22 \& 26 = 18$, koska

$$\begin{array}{rcl} & 10110 & (22) \\ \& & 11010 & (26) \\ \hline = & 10010 & (18) \end{array}$$

And-operaation avulla voi tarkastaa luvun parillisuuden, koska $x \& 1 = 0$, jos luku on parillinen, ja $x \& 1 = 1$, jos luku on pariton.

Or-operaatio

Or-operaatio $x \mid y$ tuottaa luvun, jossa on ykkösbitti niissä kohdissa, joissa ainakin toisessa luvuista x ja y on ykkösbitti. Esimerkiksi $22 \mid 26 = 30$, koska

$$\begin{array}{rcl} & 10110 & (22) \\ \mid & 11010 & (26) \\ \hline = & 11110 & (30) \end{array}$$

Xor-operaatio

Xor-operaatio $x \wedge y$ tuottaa luvun, jossa on ykkösbitti niissä kohdissa, joissa tarkalleen toisessa luvuista x ja y on ykkösbitti. Esimerkiksi $22 \wedge 26 = 12$, koska

$$\begin{array}{rcl} & 10110 & (22) \\ \wedge & 11010 & (26) \\ \hline = & 01100 & (12) \end{array}$$

Not-operaatio $\sim x$ tuottaa luvun, jossa kaikki x :n bitit on muutettu käänteisiksi. Operaatiolle pätee kaava $\sim x = -x - 1$, esimerkiksi $\sim 29 = -30$.

$$\begin{array}{rcl} x & = & 29 \quad 000000000000000000000000011101 \\ \sim x & = & 30 \quad 1111111111111111111111111100010 \end{array}$$

Vasen bittisiirto $x \ll k$ tuottaa luvun, jossa luvun x bittejä on siirretty k askelta vasemmalle (luvun loppuun tulee k nollabittiä). Oikea bittisiirto $x \gg k$ tuottaa puolestaan luvun, jossa luvun x bittejä on siirretty k askelta oikealle (luvun lopusta lähtee pois k viimeistä bittiä).

Huomaa, että vasen bittisiirto $x \ll k$ vastaa luvun x kertomista 2^k :lla ja oikea bittisiirto $x \gg k$ vastaa luvun x jakamista 2^k :lla alaspäin pyöristäen.

Luvun bitit indeksoidaan oikealta vasemmalle nolasta alkaen. Luvussa $1 < k$ on tarkalleen yksi ykkösbitti kohdassa k , joten sen avulla voi käsitellä muiden lukujen yksittäisiä bittejä.

Lauseke $x \& (x-1)$ muuttaa luvun x viimeisen ykkösbitin nollassi, ja lauseke $x \& -x$ nolaa luvun x kaikki bitit paitsi viimeisen ykkösbitin. Lauseke $x \mid (x-1)$ vuorostaan muuttaa kaikki viimeisen ykkösbitin jälkeiset bitit ykkösiksi.

GCC:n g++-kääntäjä sisältää mm. seuraavat funktiot bittien käsittelyyn:

- 91

Nämä funktiot käsittelevät int-lukuja, mutta funktioista on myös long long -versiot, joiden lopussa on päätte ll.

Seuraava koodi esittelee funktioiden käyttöä:

```
int x = 5328; // 000000000000000000001010011010000
cout << __builtin_clz(x) << "\n"; // 19
cout << __builtin_ctz(x) << "\n"; // 4
cout << __builtin_popcount(x) << "\n"; // 5
cout << __builtin_parity(x) << "\n"; // 1
```

10.3 Joukon bittiesitys

Joukon $\{0, 1, 2, \dots, n-1\}$ jokaista osajoukkoa vastaa n -bittinen luku, jossa ykkös-bitit ilmaisevat, mitkä alkiot ovat mukana osajoukossa. Esimerkiksi joukkoa $\{1, 3, 4, 8\}$ vastaa bittiesitys 100011010 eli luku $2^8 + 2^4 + 2^3 + 2^1 = 282$.

Joukon bittiesitys vie vähän muistia, koska tieto kunkin alkion kuulumisesta osajoukkoon vie vain yhden bitin tilaa. Lisäksi bittimuodossa tallennettua joukkoa on tehokasta käsitellä bittioperaatioilla.

Joukon käsittely

Seuraavan koodin muuttuja x sisältää joukon $\{0, 1, 2, \dots, 31\}$ osajoukon. Koodi lisää luvut 1, 3, 4 ja 8 joukkoon ja tulostaa joukon sisällön.

```
// x on tyhjä joukko
int x = 0;
// lisää luvut 1, 3, 4 ja 8 joukkoon
x |= (1<<1);
x |= (1<<3);
x |= (1<<4);
x |= (1<<8);
// tulostetaan joukon sisältö
for (int i = 0; i < 32; i++) {
    if (x&(1<<i)) cout << i << " ";
}
cout << "\n";
```

Koodin tulostus on seuraava:

```
1 3 4 8
```

Nyt joukko-operaatiot voi toteuttaa bittioperaatioilla:

- $a \& b$ on joukkojen a ja b leikkaus $a \cap b$ (tämä sisältää alkiot, jotka ovat kummassakin joukossa)
- $a \mid b$ on joukkojen a ja b yhdiste $a \cup b$ (tämä sisältää alkiot, jotka ovat ainakin toisessa joukossa)

- $a \& (\sim b)$ on joukkojen a ja b erotus $a \setminus b$ (tämä sisältää alkiot, jotka ovat joukossa a mutta eivät joukossa b)

Seuraava koodi muodostaa joukkojen $\{1, 3, 4, 8\}$ ja $\{3, 6, 8, 9\}$ yhdisteen:

```
// joukko {1, 3, 4, 8}
int x = (1<<1)+(1<<3)+(1<<4)+(1<<8);
// joukko {3, 6, 8, 9}
int y = (1<<3)+(1<<6)+(1<<8)+(1<<9);
// joukkojen yhdiste
int z = x|y;
// tulostetaan yhdisteen sisältö
for (int i = 0; i < 32; i++) {
    if (z&(1<<i)) cout << i << " ";
}
cout << "\n";
```

Koodin tulostus on seuraava:

```
1 3 4 6 8 9
```

Osajoukkojen läpikäynti

Seuraava koodi käy läpi joukon $\{0, 1, \dots, n-1\}$ osajoukot:

```
for (int b = 0; b < (1<<n); b++) {
    // osajoukon käsittely
}
```

Seuraava koodi käy läpi osajoukot, joissa on k alkia:

```
for (int b = 0; b < (1<<n); b++) {
    if (__builtin_popcount(b) == k) {
        // osajoukon käsittely
    }
}
```

Seuraava koodi käy läpi bittiesitystä x vastaavan joukon osajoukot:

```
int b = 0;
do {
    // osajoukon käsittely
} while (b=b-x&x);
```

Yllä olevien koodien tavoin tämä koodi käy osajoukot läpi bittiesityksen suuruusjärjestyksessä.

10.4 Dynaaminen ohjelmointi

Dynaamisen ohjelmoinnin avulla on usein mahdollista muuttaa permutaatioiden läpikäynti osajoukkojen läpikäynniksi. Tällöin dynaamisen ohjelmoinnin tilana on joukon osajoukko sekä mahdollisesti muuta tietoa.

Tekniikan hyötynä on, että n -alkioisen joukon permutaatioiden määrä ($n!$) on selvästi suurempi kuin osajoukkojen määrä (2^n). Esimerkiksi jos $n = 20$, niin $n! = 2432902008176640000$, kun taas $2^n = 1048576$. Niinpä sopivilla n :n arvoilla permutaatioita ei ehdi käydä läpi mutta osajoukot ehtii käydä läpi.

Tutustumme tekniikkaan seuraavan tehtävän kautta:

Tehtävä: Montako permutaatiota voit muodostaa luvuista $\{0, 1, \dots, n-1\}$ niin, että missään kohdassa ei ole kahta peräkkäistä lukua? Esimerkiksi kun $n = 4$, ratkaisuja on 2: $(1, 3, 0, 2)$ ja $(2, 0, 3, 1)$.

Merkitään $f(x, k)$:llä, monellako tavalla osajoukon x luvut voi järjestää niin, että viimeinen luku on k ja missään kohdassa ei ole kahta peräkkäistä lukua. Esimerkiksi $f(\{0, 1, 3\}, 1) = 1$, koska voidaan muodostaa permutaatio $(0, 3, 1)$, ja $f(\{0, 1, 3\}, 3) = 0$, koska 0 ja 1 eivät voi olla peräkkäin alussa.

Funktion f avulla ratkaisu tehtävään on summa

$$\sum_{i=0}^{n-1} f(\{0, 1, \dots, n-1\}, i).$$

Dynaamisen ohjelmoinnin tilat voi tallentaa seuraavasti:

```
long long d[1<<n][n];
```

Perustapauksena $f(\{k\}, k) = 1$ kaikilla k :n arvoilla:

```
for (int i = 0; i < n; i++) d[1<<i][i] = 1;
```

Tämän jälkeen muut funktion arvot saa laskettua seuraavasti:

```
for (int b = 0; b < (1<<n); b++) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (abs(i-j) > 1 && (b&(1<<i)) && (b&(1<<j))) {
                d[b][i] += d[b^(1<<i)][j];
            }
        }
    }
}
```

Muuttujassa b on osajoukon bittiesitys, ja osajoukon luvuista muodostettu permutaatio on muotoa (\dots, j, i) . Vaatimukset ovat, että lukujen i ja j etäisyyden tulee olla yli 1 ja lukujen tulee olla osajoukossa b .

Lopuksi ratkaisujen määrän saa laskettua näin muuttujaan s :


```

long long s = 0;
for (int i = 0; i < n; i++) {
    s += d[(1<<n)-1][i];
}

```

Dynaamisen ohjelmoinnin avulla voi ratkaista myös seuraavan tehtävän:

Tehtävä: Tarkastellaan n -alkioisen joukon osajoukkoja. Jokaista osajoukkoa x vastaa arvo $c(x)$. Tehtäväsi on jokaiselle osajoukolle x summa

$$s(x) = \sum_{y \subset x} c(y)$$

eli bittimuodossa ilmaistuna

$$s(x) = \sum_{y \& x = y} c(y).$$

Seuraavassa on esimerkki funktioiden arvoista, kun $n = 3$:

x	$c(x)$	$s(x)$
000	2	2
001	0	2
010	1	3
011	3	6
100	0	2
101	4	6
110	2	5
111	0	12

Esimerkiksi $s(110) = c(000) + c(010) + c(100) + c(110) = 7$.

Tehtävä on mahdollista ratkaista ajassa $O(2^n n)$ laskemalla arvoja funktiolle $f(x, k)$: mikä on lukujen $c(y)$ summa, missä y :n saa x :stä muuttamalla halutulla tavalla bittien $0, 1, \dots, k$ joukossa ykkösbittejä nollabiteiksi. Tämän funktion avulla ilmaistuna $s(x) = f(x, n - 1)$.

Funktion f voi laskea rekursiivisesti seuraavasti:

$$f(x, k) = \begin{cases} c(x) & \text{jos } k = -1 \\ f(x, k - 1) & \text{jos } x\text{:n bitti } k \text{ on } 0 \\ f(x, k - 1) + f(x \wedge (1 \ll k), k - 1) & \text{jos } x\text{:n bitti } k \text{ on } 1 \end{cases}$$

Pohjatapauksena $f(x, -1) = c(x)$, koska mitään bittejä ei saa muokata. Muuten jos kohdan k bitti on nolla, se säilyy nollana, ja jos kohdan k bitti on ykkönen, se joko säilyy ykkösenä tai muuttuu nolaksi.

Seuraava koodi laskee kaikki funktion s arvot taulukkoon s olettaen, että funktion c arvot ovat taulukossa c .

```
for (int b = 0; b < (1<<n); b++) s[b] = c[b];
for (int k = 0; k < n; k++) {
    for (int b = 0; b < (1<<n); b++) {
        if (b & (1<<k)) s[b] += s[b^(1<<k)];
    }
}
```

Koodi laskee ensin kaikki arvot funktiolle $f(x, 0)$, sitten kaikki arvot funktiolle $f(x, 1)$, jne.

Osa II

Verkkoalgoritmit

Osa III

Lisäaiheita

Hakemisto

- #define, 8
- bitset, 39
- complex, 254
- deque, 39
- map, 36
- multiset, 35
- next_permutation, 45
- priority_queue, 40
- queue, 40
- random_shuffle, 37
- reverse, 37
- set, 35
- sort, 27, 37
- stack, 40
- string, 34
- typedef, 8
- unordered_map, 36
- unordered_multiset, 35
- unordered_set, 35
- vector, 33
- 2SAT-ongelma, 148
- aakkosto, 229
- ahne algoritmi, 51
- aikavaativuus, 15
- alijono, 229
- alin yhteinen esivanhempi, 155
- alkuluku, 181
- alkulukupari, 183
- alkuosa, 229
- alkutekijähajotelma, 181
- Andrew'n algoritmi, 267
- aritmeettinen summa, 13
- aste, 101
- Bellman-Fordin algoritmi, 113
- binääri-indeksipuu, 80
- binäärihaku, 29
- binäärikoodi, 56
- binääripuu, 128
- binomijakauma, 215
- binomikerroin, 194
- bittiesitys, 89
- bittijoukko, 39
- Burnsiden lemma, 200
- Catalanin luku, 196
- Cayleyn kaava, 201
- de Bruijnin jono, 165
- determinantti, 205
- Dijkstran algoritmi, 116, 141
- Dilworthin lause, 178
- Diofantoksen yhtälö, 188
- Diracin lause, 164
- dynaaminen ohjelmointi, 59
- dynaaminen segmenttipuu, 247
- editointietäisyys, 68
- Edmonds-Karpin algoritmi, 170
- ehdollinen todennäköisyys, 213
- Eratostheneen seula, 184
- esijärjestys, 128
- etäisyysmitta, 260
- Eukleideen algoritmi, 184, 188
- Eukleideen kaava, 190
- Euklidinen etäisyys, 260
- Eulerin kierros, 160
- Eulerin lause, 186
- Eulerin polku, 159
- Eulerin totienttifunktio, 185
- Fermat'n pieni lause, 186
- Fibonaccin luku, 14, 190, 206
- Floyd-Warshallin algoritmi, 119
- Floydin algoritmi, 144
- Ford-Fulkersonin algoritmi, 168
- funktionaalinen verkko, 142
- geometria, 253
- geometrinen jakauma, 216

geometrinen summa, 13
 Goldbachin konjektuuri, 183
 Grundy-luku, 225
 Grundyn peli, 227

 häviötila, 221
 hajautus, 231
 hajautusarvo, 231
 hakemisto, 36
 Hallin lause, 174
 Hamiltonin kierros, 164
 Hamiltonin polku, 163
 harmoninen summa, 14, 184
 harva segmenttipuu, 247
 Heronin kaava, 253
 heuristiikka, 166
 Huffmanin koodaus, 56

 indeksien pakkaus, 87
 inklusio-eksklusio, 198
 inversio, 25
 iteraattori, 37

 jälkijärjestys, 128
 järjestäminen, 23
 jakaja, 181
 jakauma, 215
 jakso, 229
 jaollisuus, 181
 jono, 40
 joukko, 11, 35

 käänteismatriisi, 206
 Königin lause, 175
 kaari, 99
 kaarilista, 105
 kaksi osoitinta, 71
 kaksijakoisuus, 102, 112
 kaksiulotteinen segmenttipuu, 250
 keko, 40
 kekojärjestäminen, 26
 kierto, 229
 kiinalainen jäännöslause, 189
 Kirchhoffin lause, 209
 kofaktori, 205
 kokonaisluku, 6
 kombinatoriikka, 193
 kompleksiluku, 254

 konvekksi peite, 266
 koodi, 56
 koodisana, 56
 Kosarajun algoritmi, 146
 Kruskalin algoritmi, 130
 kuplajärjestäminen, 24

 lähin pienempi edeltäjä, 74
 lähin pistepari, 265
 läpimitta, 125
 Lagrangen lause, 189
 laiska eteneminen, 244
 laiska segmenttipuu, 244
 Las Vegas -algoritmi, 217
 laskemisjärjestäminen, 27
 Legendren konjektuuri, 183
 leikkaus, 168
 leikkauspiste, 257, 264
 leksikografinen järjestys, 230
 leveyshaku, 109
 liukuluku, 7
 liukuvan ikkunan minimi, 75
 logaritmi, 12
 logiikka, 11
 lomitussjärjestäminen, 25
 loppuosa, 229
 lukuteoria, 181
 lyhin polku, 113

 makro, 8
 maksimiparitus, 173
 maksimivirtaus, 167
 Manhattan-etäisyys, 260
 Markovin ketju, 216
 matriisi, 203
 matriisipotenssi, 205
 matriisitulo, 204, 218
 merkkijono, 34, 229
 merkkijonohajautus, 231
 mex-funktio, 225
 minimileikkaus, 168
 Mo'n algoritmi, 241
 modulolaskenta, 6
 Monte Carlo -algoritmi, 217
 multinomikerroin, 196
 muutoshistoria, 248

 naapuri, 101

neliöjuuri, 239
 neliömatriisi, 203
 nim-peli, 223

 odotusarvo, 214
 ohjelmointikieli, 3
 Oren lause, 164
 osajono, 229
 osajoukko, 43

 pakka, 39
 permutaatio, 45
 persistentti segmenttipuu, 248
 peruuttava haku, 46
 Pickin lause, 260
 pienin yhteinen moninkerta, 184
 pikajärjestäminen, 26
 pino, 40
 pisin nouseva alijono, 64
 piste, 254
 polkupeite, 176
 polynomien hajautus, 231
 Prüfer-koodi, 201
 prefiksi, 229
 Primin algoritmi, 134
 prioriteettijono, 40
 puolivälihaku, 49
 puu, 123
 Pythagoraan kolmikko, 190
 pyyhkäisyviiva, 263

 ratsun kierros, 166
 rekursioyhtälö, 60, 206
 repunpakkaus, 66
 reuna, 230
 riippumaton joukko, 175
 riippumattomuus, 214
 ristitulo, 256

 satunnaisalgoritmi, 217
 satunnaismuuttuja, 214
 segmenttipuu, 83, 243
 seuraajaverkko, 142
 sisäjärjestys, 128
 solmu, 99
 solmupeite, 175
 SPFA-algoritmi, 116
 Sprague–Grundyn lause, 224

suffiksi, 229
 suhteellinen alkuluku, 185
 suljettu muoto, 193
 sulkulauseke, 196
 summataulukko, 78
 suurin alitaulukko, 20
 suurin yhteinen tekijä, 184
 syöte ja tuloste, 4
 sykli, 111, 137, 143
 syklin tunnistaminen, 143
 syntymäpäiväparadoksi, 233
 syvyyshaku, 107

 täydellinen luku, 182
 törmäys, 233
 tasajakauma, 215
 tasoitettu analyysi, 71
 tekijä, 181
 tietorakenne, 33
 todennäköisyys, 211
 topologinen järjestys, 137
 transpoosi, 203
 trie, 230

 union-find-rakenne, 132

 välikysely, 77
 väritys, 102, 219
 vaativuusluokka, 18
 vahvasti yhtenäisyys, 145
 vektori, 33, 203, 254
 verkko, 99
 vieruslista, 103
 vierusmatriisi, 104
 virittävä puu

- pienin ja suurin, 129
- yhteismäärä, 209

 virtaus, 167
 voittotila, 221

 Warnsdorffin sääntö, 166
 Wilsonin lause, 191

 yhtenäisyys, 100, 111
 ykkösmatriisi, 204

 Z-algoritmi, 234
 Z-taulukko, 234
 Zeckendorfin lause, 190