Permutaatioryhmien soveltaminen GAP-ohjelmiston avulla

Pro Gradu-tutkielma
Teemu Veikanmaa (1465084)
Matemaattisten tieteiden laitos
Oulun yliopisto
Kevät 2013

Sisältö

Alkusanat

Tämä tutkielma esittelee GAP-ohjelmiston ja tutkii sen käyttöä permutaatioryhmien soveltamisessa. Tutkielman aluksi esitellään permutaatioryhmät ja joitain niiden ominaisuuksia. Toisessa ja kolmannessa luvussa esitellään GAP-ohjelmisto yleisesti ja erityisesti sen käyttö permutaatioryhmien käsittelyssä. Tutkielman loppuosa sisältää esimerkkejä permutaatioryhmien käytöstä GAP-ohjelmistolla.

Permutaatioryhmät ovat tärkeä osa matemaattista ryhmäteoriaa. Niiden tärkeyden perustelee muun muassa Cayleyn lause, joka kertoo jokaiselle ryhmälle löytyvän jonkun sen kanssa rakenteeltaan yhtäläisen permutaatioryhmän. Täten siis kaikkia permutaatioryhmien ominaisuuksia voidaan soveltaa yleisesti kaikkiin ryhmiin.

GAP-ohjelmisto (Groups, Algorithms and Programming) on monipuolinen kokoelma diskreetin matematiikan tarpeisiin soveltuvia ohjelmia. Sen mukana tulee myös kirjastoja jotka sisältävät tietoa erityyppisistä ryhmistä, mm. perustiedot kaikista pienistä ryhmistä (kertaluku ≤ 2000). GAP on laajennettavissa sen omalla ohjelmointikielellä tehdyillä ohjelmakirjastoilla.

Esimerkkisovellukset esitellään ensin yleisesti, jonka jälkeen niiden ominaisuuksia tutkitaan GAP-ohjelmiston avulla.

1 Permutaatioryhmät

Tämä luku sisältää perustiedot ryhmäteoriasta, permutaatioista ja permutaatioryhmistä sekä myöhemmissä luvuissa esiteltävien sovellusten ymmärtämiseen tarvittavat teoreettiset tiedot. Lähteinä on käytetty abstraktia algebraa käsitteleviä perusteoksia [3], [5] ja [8].

1.1 Ryhmäteorian perusteita

Ryhmä on yksi modernin algebran peruskäsitteistä. Ryhmä on seuraavan määritelmän mukainen joukko, johon on liitetty operaatio. Ryhmän operaatio määrittelee sen alkioiden väliset suhteet.

Määritelmä 1.1. Ryhmä $< G, * >$ on joukko G, joka on varustettu operaatiolla $*$ ja toteuttaa seuraavat ominaisuudet.

a) $a * b \in G$ aina, kun $a, b \in G$.

b) Kaikille $a, b, c \in G$ pätee

$(a * b) * c = a * (b * c)$.

c) On olemassa sellainen $i \in G$, että kaikilla $x \in G$

$i * x = x * i = x$.

d) Jokaista $a \in G$ kohti on olemassa yksikäsitteinen a^{-1} siten, että

$a * a^{-1} = a^{-1} * a = i$

Määritelmän 1.1 mukaista alkiota i sanotaan neutraalialkioksi, ja alkiota a^{-1} alkion a käänteisalkioksi.

Ensimmäinen ominaisuus takaa sen, että ryhmä on suljettu operaationsa suhteen, eli kaikki ryhmän alkioiden väliset laskutoimitukset tuottavat tulokseksi ryhmän alkion. Toinen ominaisuus takaa laskutoimituksen tuloksen riippumattomuuden laskuoperaatioiden suoritusjärjestyksestä (assosiatiivisuus) ja viimeiset kaksi ominaisuutta takaavat käänteisalkioiden olemassaolon ja siten muodostavat ryhmiin mielenkiintoisia rakenteita.

Huomautus 1.2. Ryhmää merkitään usein pelkällä joukon symbolilla G, jos operaatio oletetaan tunnetuksi.

Määritelmä 1.3. Ryhmän $< G, * >$ alkioiden lukumäärää $| < G, * > | = |G|$ sanotaan ryhmän G kertaluvuksi.

Määritelmä 1.4. Joukko A on ryhmän $< G, * >$ aliryhmä, jos ja vain jos $A \subseteq G$ ja A on ryhmä operaation $*$ suhteen. Tällöin merkitään $A \leq G$.

Ryhmän määrittelevät ominaisuudet ovat verrattain tuttuja eri lukujoukkojen osalta ja siksi on luonnollista esittää seuraavat kaksi esimerkkiä.

Esimerkki 1.5. *Kokonaislukujen joukko $\mathbb{Z}$ varustettuna yhteenlaskulla $< \mathbb{Z}, + >$ on ryhmä yhteenlaskun suhteen ja parilliset luvut on yksi sen aliryhmä. Neutraalialkio näissä ryhmissä on* 0.

Esimerkki 1.6. *Rationaalilukujen joukko ilman nollaa eli $\mathbb{Q}^*$ muodostaa ryhmän kertolaskun suhteen. Tämän ryhmän neutraalialkio on* 1.

Määritelmä 1.7. Ryhmää G kutsutaan kommutatiiviseksi ryhmäksi, eli Abelin ryhmäksi, mikäli kaikille $a, b \in G$ pätee $a * b = b * a$.

Esimerkki 1.8. *Esimerkkien 1.5 ja 1.6 ryhmät ovat Abelin ryhmiä.*

Seuraava lause esittää erään, joskus varsin käyttökelpoisen, tavan tutkia onko ryhmän osajoukko myös sen aliryhmä.

Lause 1.9 (Aliryhmäkriteeri)**.** *Jos $H \subseteq G$ ja $a * b^{-1} \in H$, $\forall a, b \in H$, niin $H \leq G$.*

Todistus. Olkoon $a * b^{-1} \in H$ $\forall a, b \in H$. Todistetaan että Määritelmän 1.1 ominaisuudet ovat voimassa.

c) Sijoittamalla oletukseen $b := a$, saadaan $a * a^{-1} = i \in H$.

d) Koska ylläolevan nojalla $i \in H$, niin $i * a^{-1} = a^{-1} \in H$, $\forall a \in H$.

a) Koska jokaisella alkiolla on käänteisalkio, $a * b = a * (b^{-1})^{-1}$ ja $a, b^{-1} \in H$, niin $a * b \in H$ $\forall a, b \in H$.

b) Koska laskutoimitus $*$ on sama joukossa H kuin ryhmässä G ja laskutoimitukset ovat assosiatiivisia ryhmässä G, niin ne ovat sitä myös joukossa H.

Kaikki ryhmän määrittelevät ehdot pitävät paikkansa joukossa H ja $H \subseteq G$, joten $H \leq G$.

□

1.2 Permutaatiot

Määritelmä 1.10. Bijektiota $f : A \to A$ sanotaan joukon A permutaatioksi.

Permutaatio on siis funktio, joka kuvaa kunkin joukon alkioista joksikin saman joukon alkioksi siten, että kaksi eri alkiota kuvautuvat kahdeksi eri alkioksi. Jos permutoitava joukko on järjestetty, mikä tahansa funktio joka kuvaa joukon toiseksi samoista alkioista koostuvaksi järjestetyksi joukoksi, on permutaatio.

Esimerkki 1.11. *Olkoon $A = \{1, 2, 3, 4\}$. Kuvaus f, jolle $f(1) = 2$, $f(2) = 3$, $f(3) = 1$ ja $f(4) = 4$, on eräs joukon A permutaatio.*

Määritelmä 1.12. Permutaatiot voidaan esittää edellisen esimerkin tapaan listana, tai muodossa

$$p = \begin{pmatrix} 1 & 2 & 3 & \cdots \\ p(1) & p(2) & p(3) & \cdots \end{pmatrix}, \tag{1}$$

jossa ylärivillä joukon alkiot on merkitty kokonaisluvuilla $1, 2, \ldots$ ja alarivillä on alkio, joksi permutaatio kuvaa ylärivin alkion.

Esimerkki 1.13. *Esimerkin 1.11 permutaatio esitettynä edellisen määritelmän kuvaamassa muodossa on*

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}. \tag{2}$$

Lemma 1.14. *Olkoot f ja g joukon A permutaatioita. Tällöin yhdistetty funktio $f \circ g$ on myöskin joukon A permutaatio.*

Todistus. Olkoon f ja g joukon A permutaatioita. Tällöin f ja g ovat bijektioita. Jokaiselle $a \in A$ on tällöin olemassa $f^{-1}(a) \in A$ ja $g^{-1}(f^{-1}(a)) \in A$. Täten myös $f \circ g$ on bijektio $A \to A$ eli joukon A permutaatio. □

Määritelmä 1.15. Merkitään kahden permutaation p ja q yhdistettyä funktiota $f \circ g$ kirjoittamalla permutaatiot peräkkäin pq ja kutsutaan tätä permutaatioiden tuloksi.

Huomautus 1.16. Koska permutaatioiden tulo pq on lyhennysmerkintä yhdistetystä funktiosta $f \circ g$, se permutoi alkiot ensin permutaation q ja sitten permutaation p mukaisesti. Permutaatioiden tulossa operaatiot suoritetaan siis oikealta vasemmalle.

Määritelmä 1.17. Jokainen joukon A permutaatio p jakaa joukon A alkiot pistevieraisiin osajoukkoihin $A_i \subseteq A, i \in \mathbb{Z}$, joille $a, b \in A_i$, jos ja vain jos $p^n(a) = b$ jollain $n \in \mathbb{Z}$. Osajoukkoa johon alkio $a \in A$ kuuluu, kutsutaan alkion a radaksi permutaatiossa p.

Jokainen alkio kuuluu siis tarkalleen yhteen rataan, joka sisältää kaikki alkiot, joille permutaatio voi kuvata tarkasteltavan alkion.

Määritelmä 1.18. Permutaatiota S, jolla on vain yksi kahdesta tai useammasta alkiosta koostuva rata, kutsutaan sykliksi. Syklin alkioiden lukumäärää kutsutaan syklin pituudeksi. Sykliä merkitään $S = (a_1\ a_2\ \ldots\ a_n)$, jossa $S(a_i) = a_{i+1}$, $\forall i \in 1, 2, \ldots, n-1$ ja $S(a_n) = a_1$.

Koska kaikki permutaatiot ovat jaettavissa pistevieraisiin ratoihin, ja siis koostuvat erillisistä sykleistä, permutaatiot on kätevää esittää ns. syklimuodossa.

Määritelmä 1.19 (Sykliesitys)**.** Permutaatiot voidaan esittää syklimuodossa kirjoittamalla kaikki permutaation sisältämät syklit peräkkäin. Jos permutaatio on identiteettikuvaus, eli se kuvaa jokaisen alkion itsekseen, merkitään (1).

Esimerkki 1.20. *Olkoon permutaatio* $p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 1 & 2 \end{pmatrix}$. *Tällöin sen radat ovat* $\{1, 3, 4\}$ *ja* $\{2, 5\}$. *Permutaatio voidaan esittää syklimuodossa* $p = (1\ 3\ 4)(2\ 5)$.

Esimerkki 1.21. *Esimerkin 1.11 permutaatio esitettynä syklimuodossa on* $(1\ 2\ 3)$.

Syklimuotoisessa esityksessä permutaatio saadaan esitettyä yhdellä rivillä ja kukin joukon alkio esiintyy esityksessä enintään kerran. Koska syklin määritelmässä vaaditaan syklin pituudeksi vähintään 2, alkiot, jotka permutaatio kuvaa itsekseen, jätetään kokonaan merkitsemättä sykliesitykseen.

1.3 Symmetrinen ryhmä

Lause 1.22. *Olkoon A epätyhjä joukko ja S_A kaikkien joukon A permutaatioiden muodostama joukko. Tällöin S_A on ryhmä funktioiden yhdistämisen suhteen.*

Todistus. Olkoon f, g ja h joukon A permutaatioita ja $*$ funktioiden yhdistämisoperaatio. Osoitetaan että S_A toteuttaa ryhmän määritelmässä 1.1 vaaditut ominaisuudet.

a) Koska sekä f että g ovat bijektioita $A \to A$, niin myös yhdistetty funktio $f * g$ on bijektio $A \to A$ eli permutaatio ja $f * g \in S_A$.

b) Olkoon $a \in A$. Koska f, g ja h ovat bijektioita, ne ovat määritellyt kaikilla joukon A alkioilla. Tällöin $((f * g) * h)(a) = (f * g)(h(a)) = f(g(h(a))) = f(g * h(a)) = (f * (g * h))(a)$.

c) Olkoon $P : A \to A$, $P(a) = a, \forall a \in A$. Tällöin P on bijektio, ja joukon A permutaatio, jolle pätee $P * f(a) = P(f(a)) = f(a)$ ja $f * P(a) = f(a)$, joten P on neutraalialkio operaation $*$ suhteen joukossa S_A.

d) Olkoon g permutaatio $A \to A$. Funktio g on siis bijektio, joten sillä on käänteisfunktio $g^{-1} : A \to A$, joka on myöskin permutaatio. Nyt $g, g^{-1} \in S_A$ ja $g * g^{-1} = g^{-1} * g = (1) = i$.

□

Määritelmä 1.23. Olkoon $A = \{1, 2, \cdots, n\}$ äärellinen joukko. Lauseen 1.22 nojalla sen kaikkien permutaatioiden joukon S_A alkiot muodostavat ryhmän funktioiden yhdistämisen suhteen. Tätä ryhmää sanotaan astetta n olevaksi symmetriseksi ryhmäksi ja merkitään S_n.

Huomautus 1.24. Symmetrisen ryhmän S_n alkioiden lukumäärä eli kertaluku on $n!$.

Esimerkki 1.25. *Astetta* 2 *oleva symmetrinen ryhmä S_2 on alkioiden $\{1, 2\}$ kaikkien permutaatioiden muodostama ryhmä. Tämän ryhmän alkiot ovat siis* (1) *ja* (1 2).

Esimerkki 1.26. *Symmetrisen ryhmän S_3 alkiot ovat* (1), (1 2), (1 3), (2 3), (1 2 3) *ja* (1 3 2).

Koska esimerkiksi (1 2)(1 2 3) = (2 3) *ja* (1 2 3)(1 2) = (1 3), *S_3 ei ole kommutatiivinen eli Abelin ryhmä. Itse asiassa S_3 on kertaluvultaan pienin ei-kommutatiivinen ryhmä.*

1.4 Alternoivat ryhmät

Määritelmä 1.27. Transpoosi on sellainen permutaatio, joka kuvaa kaksi alkiota toisikseen, jättäen kaikki muut käsiteltävän joukon alkiot ennalleen. Transpoosi, joka kuvaa joukon alkiot a ja b toisikseen, on syklimuodossa esitettynä $(a\ b)$.

Lause 1.28. *Jokainen permutaatio voidaan esittää transpoosien tulona.*

Todistus. Jokainen sykli $(x_1\ x_2\ \dots\ x_n)$ voidaan esittää transpoosien tulona

$$(x_1\ x_n)(x_1\ x_{n-1})\dots(x_1\ x_2)$$

ja jokainen permutaatio voidaan esittää erillisten syklien tulona. □

Huomautus 1.29. Transpoosiesitys ei ole yksikäsitteinen, vaan esimerkiksi minkä tahansa transpoosin lisääminen kahdesti peräkkäin mihin tahansa kohtaan transpoosiesitystä tuottaa saman permutaation.

Lause 1.30. *Permutaation transpoosiesityksen transpoosien määrä on joko parillinen tai pariton.*

Todistus. Tehdään vastaoletus: on olemassa sellainen joukon $A = \{1, 2, \dots, n\}$ permutaatio p joka voidaan saada aikaan sekä parittoman, että parillisen transpoosimäärän tulona.

Olkoon matriisi M sellainen $n \times n$ matriisi, joka saadaan $n \times n$ yksikkömatriisista järjestämällä sen vaakarivit $i_1, i_2, \dots, i_n$ järjestykseen $i_{p(1)}, i_{p(2)}, \dots, i_{p(n)}$.

Kukin transpoosi, eli matriisin vaakarivien vaihtaminen keskenään, muuttaa matriisin determinantin vastakkaismerkkiseksi. Koska permutaatio p voidaan saadaan sekä parillisen että parittoman transpoosimäärän tulona, voidaan matriisi M saada luotua yksikkömatriisista sekä parillisella että parittomalla määrällä vaakarivien vaihtoja. Täten matriisin M determinantti on sekä $+1$ että -1. Koska matriisin determinantti on yksikäsitteinen, päädytään ristiriitaan eli vastaoletus on väärä. □

Määritelmä 1.31. Jokaisella permutaatiolla on pariteetti, joka on joko parillinen tai pariton. Permutaation pariteetti vastaa sen transpoosiesityksen pariteettia.

Yksittäinen transpoosi on siis pariton. Permutaatio, joka voidaan esittää parillisen transpoosijoukon tulona on parillinen ja permutaatio, joka voidaan esittää parittoman transpoosijoukon tulona, on vastaavasti pariteetiltaan pariton. Koska identiteettikuvaus (1) saadaan minkä tahansa transpoosin tulona itsensä kanssa, identiteettikuvaus on pariteetiltaan parillinen.

Lemma 1.32. *Symmetrisen ryhmän $S_n, n \geq 2$ sisältämien parillisten ja parittomien permutaatioiden määrä on sama.*

Todistus. Olkoot $n \geq 2$, A kaikkien symmetrisen ryhmän S_n parillisten ja B sen kaikkien parittomien permutaatioiden joukko ja kuvaus $f(x) = (1\ 2)x$.

Koska kuvaus f lisää permutaation transpoosiesitykseen yhden transpoosin, se kuvaa joukon A parittomien permutaatioiden joukoksi $f(A) \subseteq B$. Vastaavasti joukko B kuvautuu parillisten permutaatioiden joukoksi $f(B) \subseteq A$. Koska $f^{-1}(x) = (1\ 2)^{-1}x = (1\ 2)x = f(x)$, f on bijektio $A \to B$ ja $|A| = |B|$. □

Koska symmetrisen ryhmän parillisten permutaatioiden joukko sisältää identiteettikuvauksen ja parillisten permutaatioiden tulot ja käänteisalkiot ovat parillisia, parilliset permutaatiot muodostavat ryhmän.

Määritelmä 1.33. Astetta $n \geq 2$ oleva alternoiva ryhmä A_n on symmetrisen astetta n olevan ryhmän parillisten permutaatioiden muodostama ryhmä. Alternoivan ryhmän A_n kertaluku on $\frac{1}{2}n!$.

1.5 Cayleyn lause

Cayleyn lause perustelee permutaatioryhmien tärkeyden. Sen mukaan jokainen äärellinen ryhmä on rakenteeltaan samankaltainen jonkin permutaatioryhmän kanssa. Tämä mahdollistaa esimerkiksi kaikkien permutaatioryhmien ominaisuuksien käyttämisen minkä tahansa ryhmän käsittelyyn.

Lause 1.34. *Jokainen äärellinen ryhmä on isomorfinen jonkin permutaatioryhmän kanssa.*

Todistus. Olkoon G äärellinen ryhmä, jonka kertaluku on n. Merkitään ryhmän G alkioita $x_1, x_2, \ldots, x_n$. Olkoon P joukko kuvauksia $P = \{f_1, f_2, \ldots, f_n\}$, joille $f_i(g) = x_i g$, $\forall i \in \{1, 2, \ldots, n\}\ \forall g \in G$.

Merkitään $a_{ij} = x_i^{-1}x_j$. Koska $x_i(x_i^{-1}x_j) = x_i a_{ij} = x_j$, eli jokaiselle ryhmän G alkioparille (x_i, x_j) on olemassa $a_{ij} \in G$, siten että $x_i a = x_j$, niin $x_i g$ käy läpi kaikki ryhmän G alkiot, kun g käy ne läpi.

Täten kukin kuvaus $f_i : G \to G$, $f_i(g) = x_i g$ on bijektio ja määrittelee permutaation

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_i x_1 & x_i x_2 & \cdots & x_i x_n \end{pmatrix} \in S_n \tag{3}$$

.

Olkoon kuvaus $h : G \mapsto P$, $h(x_i) = f_i$, $\forall i \in \{1, 2, \ldots, n\}$. Koska $h(x_a)h(x_b)(g) = f_a f_b(g) = x_a x_b g = h(x_a x_b)(g)$, niin h on homomorfismi $G \to P$.

Täten G on isomorfinen permutaatioryhmän $P < S_n$ kanssa.

□

1.6 Burnsiden lemma

Seuraavan lemman avulla voidaan laskea erilaisten ominaisuusyhdistelmien määrää, kun osa ominaisuuksista säilyy symmetrioiden ansiosta joissain permutaatioissa. Tavallinen käyttökohde Burnsiden lemmalle on symmetristen esineiden oleellisesti erilaisten kuviointien laskeminen, joita on sekä tämän luvun esimerkeissä, että myöhempien lukujen sovelluksissa.

Lemma 1.35 (Burnsiden lemma)**.** *Olkoon G permutaatioryhmä ja N_g niiden ominaisuusyhdistelmien joukko, jotka permutaatio $g \in G$ pitää ennallaan. Kun sellaisia ominaisuusyhdistelmiä, jotka saadaan toisistaan jollain permutaatiolla, pidetään samoina, niin tutkittavan joukon eri ominaisuusyhdistelmien määrä on*

$$N = \frac{1}{|G|} \sum_{g \in G} |N_g|.$$

Esimerkki 1.36. *Tehtävä: Kun neliö jaetaan lävistäjiensä suuntaisilla viivoilla osiin ja jokainen syntynyt ala väritetään joko mustaksi tai valkoiseksi, montako erilaista väritystä on olemassa, kun sellaisia värityksiä, jotka saadaan toisistaan neliötä pyörittämällä pidetään samoina?*

Olkoon neliön sivut numeroitu ylimmäisestä alkaen myötäpäivään luvuin $1-4$.Neliön rotaatioista muodostettu ryhmä on tällöin $G = \{(1), (1\ 2\ 3\ 4), (1\ 3)(2\ 4), (1\ 4\ 3\ 2)\}$.

Permutaatio (1) in identiteettikuvaus, joka säilyttää kaikki mahdolliset väritykset sellaisenaan. Näitä värityksiä on 2^4.

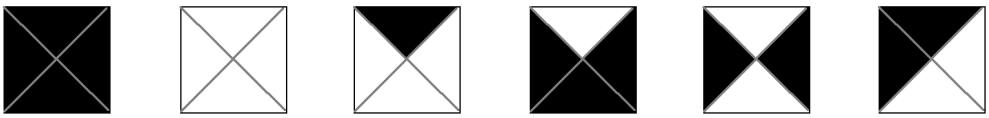
Permutaatiot $(1\ 2\ 3\ 4)$ *ja* $(1\ 4\ 3\ 2)$ *vastaavat* 90 *asteen rotaatiota vasta- ja myötäpäivään. Molemmat näistä säilyttävät värityksen vain yksivärisissä neliöissä, joita on* 2 *erilaista.*

Permutaatio $(1\ 3)(2\ 4)$ *vastaa 180 asteen rotaatiota. Se säilyttää kaikki neliöt, joiden vastakkaisilla sivuilla olevat alat on väritetty samalla värillä. Tällaisia värityksiä on* 2^2 *erilaista.*

Burnsiden lemman mukaan erilaisten väritysten määrä on

$$N = \frac{1}{4}\sum_{g\in G}|N_g| = \frac{1}{4}(2^4 + 2^2 + 2\times 2) = \frac{1}{4}(16+4+4) = \frac{24}{4} = 6.$$

Nämä erilaiset väritykset ovat



Esimerkki 1.37. *Jos edellisen esimerkin neliön voi myös kääntää ympäri, ja osat ovat samanväriset molemmin puolin, permutaatioryhmän* G *kertaluku on kaksinkertainen eli* $|G| = 8$

Edellisen esimerkin permutaatioiden lisäksi ryhmässä on

- *180 asteen pyöräytys vastakkaisten sivujen keskipisteiden määrittelemän akselin ympäri eli permutaatiot* (13) *ja* (24) *jotka säilyttävät värityksen* 2^3 *neliössä*
- *180 asteen pyöräytys vastakkaisten kulmien määrittelemän akselin ympäri eli permutaatiot* $(12)(34)$ *ja* $(14)(23)$ *jotka säilyttävät värityksen* 2^2 *neliössä*

Nyt väritysvaihtoehtojen määräksi saadaan Burnsiden lemmaa käyttäen

$$N = \frac{1}{8}\sum_{g\in G}|N_g| = \frac{1}{8}(2^4 + 2\times 2^3 + 3\times 2^2 + 2\times 2) = \frac{1}{8}(16+16+12+4) = \frac{48}{8} = 6,$$

joka on odotetusti sama kuin edellisessä esimerkissä.

2 GAP-ohjelmisto

2.1 Yleistä

GAP (Groups, Algorithms and Programming) on vapaa, avoin ja laajennettava ohjelmistokokoelma diskreettien algebrallisten rakenteiden laskennalliseen tutkimukseen ja soveltamiseen. GAP-ohjelmiston yhteydessä vapaalla tarkoitetaan ilmaista ja vapaasti levitettävää, avoimella mahdollisuutta tutkia ja muunnella lähdekoodia ja laajennettavuudella mahdollisuutta liittää omia ohjelmiaan ja funktioitaan osaksi GAP-ohjelmiston toimintaa.[4]

GAP on saatavissa ohjelmiston kotisivuilta osoitteesta `http://www.gap-system.org/`. GAP sisältyy myös monien Linux-jakeluiden käyttämiin pakettivarastoihin.

2.2 Historia

GAP-ohjelmiston kehitys alkoi RWTH-Aachenin Matemaattisten tieteiden laitoksella (Lehrstuhl D für Mathematik) vuonna 1985 Joachim Neubüserin johdolla. Nykyään ohjelmiston kehitys on laajentunut ja kansainvälistynyt, ja sen eri versioiden kehitystä on johdettu eri puolilta maailmaa.

2.3 Ohjelmiston rakenne

Perusohjelmisto koostuu neljästä pääosasta, jotka ovat

1) C-kielellä toteutettu ydin

- automaattinen ja dynaaminen tiedonhallinta
- peruslaskutoimitukset kokonaisluvuilla, äärellisissä kunnissa, permutaatioilla ja sanoilla
- perustietorakenteet ja niiden operaatiot
- GAP-olio-ohjelmointikielen tulkki
- pieni joukko systeemifunktioita tiedostonkäsittelyn ja ulkoisten ohjelmien ajamisen toiminnan yhtenäistämiseksi eri käyttöjärjestelmissä
- ohjelmointityökaluja testaukseen ja debuggaukseen sekä ajastusalgoritmit
- käyttöliittymä, jolla ohjelmistolle voidaan syöttää käskyjä, ja jonka avulla ohjelmisto tulostaa palautteen käyttäjälle

2) GAP-ohjelmointikielellä kirjoitettu kirjasto, joka sisältää ydintä laajemman joukon algoritmeja

3) kirjasto, joka sisältää teoreettista tietoa algebrallisista rakenteista

4) dokumentaatio sekä tulostettavassa, että digitaalisesti selattavassa HTML-muodossa

Ohjelmistoa voidaan laajentaa ns. laajennuspaketeilla. Laajennuspaketit on kirjoitettu GAP-ohjelmointikielellä ja ne sisältävät GAP-koodin lisäksi dokumentaation ja mahdollisesti myös erillisiä ohjelmia, joita hyödynnetään paketin sisältämän koodin avulla.

Algebrallisten rakenteiden kirjasto sisältää mm. pienten ryhmien (kertaluku ≤ 2000 poislukien 1024), primitiivisten permutaatioryhmien (kertaluku ≤ 2500) ja äärellisten täydellisten ryhmien (kertaluku $\leq 10^6$ muutamaa kertalukua lukuunottamatta) tiedot, joita voi käyttää ryhmiä tutkittaessa.

2.4 Ohjelmiston käyttöliittymä

Ohjelmisto käynnistetään ja sitä käytetään perinteiseen tapaan komentoriviltä. Käynnistettäessä GAP näyttää ohjelman perustiedot sekä tiedot käytössä olevista komponenteista ja kirjastoista. Tämän jälkeen ohjelman näyttää komentokehotteensa

```
gap>
```

johon käyttäjä voi kirjoittaa haluamansa komennon. Kaikki komennot päättyvät kaksoispisteeseen (;), ja niitä, sekä silmukka- ja toistorakenteita, ketjuttamalla voidaan luoda pidempiä kokonaisuuksia; algoritmeja ja ohjelmia. Ohjelmiston käyttö lopetetaan ja siitä poistutaan komennolla `quit;`.

GAP-ohjelmistolle on olemassa myös ikkunoitu käyttöliittymä GGAP, joka helpottaa tiedostojen hallintaa ja sisältää avustetun käskyjen syötön ja erillisen ikkunan ohjeille. GGAP toimii pääasiassa samalla tavalla kuin komentoriviversiokin, mutta työtilojen hallinta sekä virheellisten komentojen korjaaminen on huomattavasti helpompaa.

2.5 Operaattorit

2.5.1 Aritmeettiset operaattorit

GAP-ohjelmiston aritmeettiset operaattorit ovat $+$, $-$, $*$, $/$, *mod* ja $\hat{}$. Ne kaikki ovat kaksipaikkaisia ja toteuttavat yhteen-, vähennys-, kerto- ja jakolaskun, kongruenssin ja potenssiinkorotuksen.

Peruslaskutoimitukset ($+$, $-$, $*$ ja $/$) antavat luvuille tavallisten laskusääntöjen mukaiset tulokset. Plus- ja miinusoperaattorit toimivat myös yksipaikkaisina lukuarvojen edessä, jolloin miinus tuottaa käänteisarvon ja plus ei tee mitään. Peruslaskutoimituksia on määritelty myös muille tietotyypeille ja olioille. Seuraavassa esimerkkejä peruslaskutoimitusten käytöstä.

```
gap> 123+456;
579
gap> 123-456;
-333
gap> 123*456;
56088
gap> 123/456;
41/152
gap> 12+5/2.1;
14.381
gap> 1/3 + 6/5;
23/15
```

Operaattori `mod` laskee jäännösluokat eri lukutyypeille ja sillä voidaan ratkaista myös kokonaisluvun modulaariaritmeettinen käänteisluku. Alla esimerkkejä operaattorin käytöstä. Alin esimerkki laskee käänteisluvun luvulle 19 (*mod* 7).

```
gap> 456 mod 123;
87
gap> 12.3 mod 4.56;
3.18
gap> 1/19 mod 7;
3
```

Potenssiinkorotus voi tuottaa suuria lukuja, mutta ne eivät tuota ongelmia GAP:ssä.

```
gap> 2^16;
65536
gap> 123^456;
992500687720988567008314620574696326372959408198869005198162988813828671
047493990779211286614261446380554242369362718724928003527416499021181438
196726015699981001207904967595176364654458956257416098662099005001984071
532446047789680169630280503102614176159144687299182406854878786176459769
390634643579861657117309763994785076492286863414669671679101266533421349
427448514638999274870924866109771461127635671016726459531321964814393398
730170881404146612711985003332557130961423351514146306516830655187840812
036784877030028020820912366035190262568806244996817813872275740354848312
715156831237421490955692604636096559777009388445806119312464951662086955
403136981400116380273225662526897808381363518287953142721621112222311709
017156123557013475523715300136938553798348656670600146433024591004297836
539669137830022907842834556282833554705299329560514844771293338811599302
127586876027950885792304316616960102321873904366016141456032419023866634
42520160735566561
```

2.5.2 Vertailuoperaattorit ja loogiset operaattorit

GAP-ohjelmiston vertailuoperaattorit ovat =(yhtäsuuruus), <>(erisuuruus), <(pienempi kuin), >(suurempi kuin), <=(pienempi tai yhtäsuuri kuin), >=(suurempi tai yhtäsuuri kuin). Operaattorit palauttavat totuusarvon `true` tai `false`.

```
gap> 123 < 456;
true
gap> 23 > 45;
false
gap> "abc" <> "efg";
true
gap> 43=43;
true
gap> "abc" <= "def";
true
```

Käytössä olevat loogiset operaattorit ovat `or`, `and` ja `not`.

```
gap> true or false;
true
gap> true and false;
false
gap> not true;
false
```

2.6 Muuttujat

GAP-ohjelmistossa muuttujalle annetaan arvo operaattorilla `:=`. Muuttujat eivät ole tyypitettyjä, vaan ne toimivat lyhennysmerkintöinä kaikkille käytetyille tietotyypeille ja olioille.

```
gap> a:=12;
12
gap> luku:=34;
34
gap> tot:=a=b;
false
gap> luku2:=a+luku;
46
gap> a+luku+luku2;
92
gap> a=a^a;
false
gap> a:=a^a;
8916100448256
gap> luku;
34
```

2.7 Listat

Listat ovat järjestyksensä säilyttäviä jonoja, jotka koostuvat yksitttäisistä vakioista. Vakioiden tietotyyppejä ei tarkisteta listaa luodessa, vaan vasta vääränlainen käyttö antaa virheilmoituksen. Muuttujan tallennus listaan tallentaa muuttujan senhetkisen arvon.

```
gap> lista:=["a","b"];
[ "a", "b" ]
gap> lista2:=[3,1,2];
[ 3, 1, 2 ]
```

Säännöllisiä kokonaislukulistoja voidaan määritellä antamalla lukuvälin alku- ja loppuarvot. Mikäli perättäisten listan lukujen väliksi halutaan joku muu kuin 1, on listan kaksi ensimmäistä lukua määriteltävä alla olevien esimerkkien mukaisesti.

```
gap> [1..10];
[ 1 .. 10 ]
gap> parilliset:=[2,4..20];
[ 2, 4 .. 20 ]
gap> Length(parilliset);
10
gap> tasakymmenet:=[10,20..1000];
[ 10, 20 .. 1000 ]
gap> Length(tasakymmenet);
100
gap> [20,19..0];
[ 20, 19 .. 0 ]
```

Listoja voidaan luoda myös toisista listoista kuvaamalla niiden kukin alkio jonkin määritellyn funktion avulla allaolevalla tavalla. Jos funktiolla on vain yksi argumentti, riittää funktion nimi kuvaukseksi.

```
gap> kertomat:=List([1..5], Factorial);
[ 1, 2, 6, 24, 120 ]
gap> neliot:=List([1..8], x -> x^2);
[ 1, 4, 9, 16, 25, 36, 49, 64 ]
```

Listoja käytetään myös vektoreiden esittämisessä. Matriisit esitetään listoista muodostettuina listoina.

```
ap> vektori:=[4,2,-1];
[ 4, 2, -1 ]
gap> skalaaritulo:=vektori*vektori;
21
```

Funktion `ForAll` avulla voidaan tutkia päteekö annettu ehtolause kaikilla listan alkioilla.

```
gap> ForAll(lista, x -> x<"c");
true
gap> ForAll(lista2, x -> x>2);
false
```

Listojen ominaisuuksien tutkimiseen on monia funktioita, joista alla esitetään muutama. Lisäksi listoina esitettävien vektoreiden ja matriisien tutkimiseen ja käsittelyyn on runsaasti valmiita funktioita. Lisätietoa niistä löytyy GAP dokumentaatiosta [4].

```
gap> IsSortedList(lista);
true
gap> IsSortedList(lista2);
false
gap> IsDenseList(lista);  ## Kaikki määritellyt alkiot peräkkäin
true
gap> IsDenseList([0, 1, , , , 1]);
false
gap> IsHomogeneousList(lista); ## Kaikki alkiot saman tyyppisiä
true
gap> IsHomogeneousList([1, 'a']);
false
```

2.8 Ehtolauseet ja silmukat

GAP:n ehtolauseet annetaan muodossa

```
if [EHTO] then [TOIMINTO1]
elif [EHTO2] then [TOIMINTO2]
else [TOIMINTO3]
fi;
```

jossa kukin `[EHTOX]` on totuusarvo tai sen palauttava lauseke ja kukin `[TOIMINTOX]` on komentosarja, joka ehdon toteutuessa suoritetaan. Rivit `elif` ja `else` eivät ole ehtolauseen pakollisia osia. Osa `elif` voi esiintyä ehtolauseessa useamman kerran, mutta `else` vain kerran. Osan `else` sisältämät komennot suoritetaan mikäli sitä edeltävä ehto (`if` tai `elif`) ei toteudu. Osa `elif` on vastaava kuin `else`, mutta se vaatii lisäksi oman ehtolauseensa toteutumisen.

```
gap> luku:=5;
5
gap> if luku<5 then Print("alle viisi");
> elif luku>5 then Print("yli viisi");
> else Print("tasan viisi"); fi;
tasan viisi
```

Käytettävissä on myös kolme silmukkarakennetta: `for`, `while` ja `repeat`. Näistä `while` ja `repeat` ovat ehdollisia ja `for` käy läpi sille annetun listan.

Listoja läpikäyvä `for`-silmukka annetaan muodossa

```
for [MUUTTUJA] in [LISTA] do [OPERAATIO] od;
```

jossa `[MUUTTUJA]` on merkkijono, jolla nimetään muuttuja, joka käy listan arvot läpi. Tämän muuttujan avulla listan arvoja voidaan käyttää kutakin vuorollaan silmukan eri kierroksilla. `[LISTA]` sisältää listan läpikäytävistä arvoista ja `[OPERAATIO]` sisältää kunkin listan arvon kohdalla suoritettavan komentosarjan.

```
gap> luvut:=[1,1,2,3,5,8,12];
[ 1, 1, 2, 3, 5, 8, 12 ]
gap> for i in luvut do
```

```
> Print(i*2.2," ");
> od;
2.2 2.2 4.4 6.600000000000001 11 17.6 26.4
```

```
gap> for i in [1..10] do
> Print(i," ");
> od;
1 2 3 4 5 6 7 8 9 10
```

Silmukat `while` ja `repeat` annetaan muodoissa

```
while [EHTO] do [OPERAATIO] od;
repeat [OPERAATIO] until [EHTO];
```

joista `while` toteuttaa operaatioitaan, niin kauan kuin ehto on voimassa.

```
gap> index:=0;
0
gap> while index<10 do
> Print(index*3," ");
> index:=index+2;
> od;
0 6 12 18 24
```

Silmukka `repeat` toistaa operaatioitaan, kunnes annettu ehto toteutuu.

```
gap> luku:=1;
1
gap> repeat Print(luku, " "); luku:=luku+7; until luku mod 13=0;
1 8 15 22 29 36 43 50 57 64 71
```

Kaikki silmukat voidaan keskeyttää käskyllä `break`. Käskyllä `continue` keskeytetään silmukan kyseinen kierros ja siirrytään seuraavaan.

2.9 Funktiot

Funktiot määritellään sijoittamalla halutun nimiseen muuttujaan function(muuttujat).

```
gap> kertoma:=function(n)
> local tulo;
> tulo:=1;
> for i in [2..n] do
> tulo:=tulo*i;
> od;
> return tulo;
> end;
function( n ) ... end
```

Käskyllä `return` palautetaan haluttu arvo. Funktio loppuu käskyyn `end`. Käsky `local` määrittelee paikallisesti käytettävät muuttujat, ja estää funktiota muuttamasta samannimisiä globaaleja muuttujia.

Funktiota käytetään syöttämällä funktion nimi ja suluissa lista sen vaatimista parametreista. Jollei parametreja ole, funktion nimen jälkeen syötetään tyhjät sulut.

```
gap> kertoma(5);
120
gap> kertoma(15);
1307674368000
gap>
```

2.10 Lisätietoa

Koska GAP sisältää suuren määrän erilaisia funktioita ja perustoimintojakin voidaan käyttää monin tavoin, tämän tutkielman laajuuteen ei saada mahtumaan kaikkia, mahdollisesti tärkeitäkään, ominaisuuksia. Tässä ja seuraavassa luvussa on kuitenkin pyritty kuvaamaan ne toiminnot, joiden käyttöä myöhempien lukujen esimerkeissä käsitellään.

GAP-ohjelmiston kotisivuilta `http://www.gap-system.org/` löytyvä "The GAP reference manual" sisältää lisätietoa ohjelmiston käytöstä, sekä käskyjen ja funktioiden tarkemmat kuvaukset. Sivuilta löytyy myös muuta materiaalia GAP:n avulla tapahtuvaan laskentaan.

3 Permutaatioryhmien käsittely GAP-ohjelmistolla

3.1 Permutaatiot

Permutaatiot määritellään syöttämällä halutun permutaation erilliset syklit peräkkäin. Syklit muodostetaan sulkeiden sisälle ja niiden alkiot erotetaan toisistaan pilkuilla. GAP näyttää palautteena permutaation, tai syötön epäonnistuessa virheilmoituksen. Identiteettikuvausta (1) merkitään tyhjillä sulkeilla.

```
perm1 := (1,2,3);
(1,2,3)
perm2 := (1,2)(4,5,6);
(1,2)(4,5,6)
perm3 := (2,4)(4,5);
Permutation: cycles must be disjoint and duplicate-free
```

GAP-ohjelmistossa permutoitavia alkioita merkitään positiivisilla kokonaisluvuilla. Suurin permutoitava alkio voi olla $2^{28} - 2$.

Permutaation voi määritellä myös funktion $PermList(list)$ avulla. Funktion argumenttina annetaan lista, jonka alkioina ovat halutun permutaation π kuvat listan indeksin $1, 2, \ldots, n$ arvoilla eli $\pi(1), \pi(2), \ldots, \pi(n)$, jossa n on suurin alkio, jonka permutaatio π kuvaa muuksi kuin itsekseen.

```
PermList([1,3,2,4,6,5,7,8])
(2,3)(5,6)
```

Vastaavasti funktio $ListPerm(\pi)$ palauttaa listan $[\pi(1), \pi(2), \ldots, \pi(n)]$, joka sisältää permutaation π alkioiden $1, 2, \ldots, n$ kuvat. Tässäkin n on suurin alkio, jonka permutaatio π kuvaa muuksi kuin itsekseen.

```
ListPerm(perm2);
[ 2, 1, 3, 5, 6, 4 ]
```

Alkiota voidaan permutoida kaksipaikkaisella operaattorilla ^, jonka ensimmäisenä argumenttina annetaan permutoitava alkio a ja toisena permutaatio π. Tuloksena saadaan alkio $\pi(a)$, eli annettu alkio permutoituna annetulla permutaatiolla.

```
5^perm1;
5
5^perm2;
6
```

Toinen permutaatioiden perusoperaatio on kaksipaikkainen operaatio $*$, joka määrittelee permutaatioiden tulon eli yhdistetyn funktion jossa sisäfunktiona on ensimmäisenä argumenttina annettu permutaatio. Permutointi suoritetaan siis vasemmalta oikealle.

```
perm1*perm2;
(2,3)(4,5,6)
perm2*perm1;
(1,3)(4,5,6)
```

Funktiot $SmallestMovedPoint(X)$ ja $LargestMovedPoint(X)$ palauttavat pienimmän ja suurimman alkion jonka permutaatio kuvaa muulle kuin itselleen. Funktion argumenttina X voi olla joko yksi permutaatio tai permutaatiojoukko. Identiteettikuvaukselle funktiot palauttavat arvot 0 ja $infinity$.

```
gap> SmallestMovedPoint(perm1);LargestMovedPoint(perm2);
1
6
gap> SmallestMovedPoint(());LargestMovedPoint(());
infinity
0
```

Permutaatioilla on myös vertailuoperaattorit $=$ ja $<$. Permutaatiolle π ja ρ vertailu $\pi = \rho$ palauttaa totuusarvon $true$, jos ne permutoivat samoja alkioita ja kaikkien näiden alkioiden kuva on sama. Vertailuoperaattori $<$ palauttaa totuusarvon permutaatioiden järjestyksestä. Järjestys on määritelty siten että $\pi < \rho$, jos $\pi(i) < \rho(i)$, jossa i on pienin alkio, jolle $\pi(i) \neq \rho(i)$.

```
perm1=perm2;perm1<perm2;
false
false
```

Permutaation syklirakenne saadaan selvitettyä funktiolla CycleStructurePerm(π), joka palauttaa listan permutaation π eripituisten permutaatioiden lukumäärästä. Lista alkaa 2-sykleistä ja päättyy permutaation suurimpaan sykliin.

```
CycleStructPerm((1,2,3,4,5)(6,7,8));
[ , 1,, 1 ]
```

Permutaation p pariteetti saadaan selvitettyä funktiolla $SignPerm(p)$. Funktio palauttaa arvon $(-1)^k$, jossa k on permutaation parittomien syklien määrä. funktion tuloksena on siis parillisilla permutaatioilla arvo 1 ja parittomilla -1.

```
SignPerm(perm1);
1
SignPerm(perm1*perm2);
-1
```

3.2 Permutaatioryhmät

Permutaatioryhmät määritellään ryhmän generoivien permutaatioiden tai valmiiden funktioiden avulla.

```
G := Group([perm1,perm2]);
Group([ (1,2,3), (1,2)(4,5,6) ])
S5 := Group([(1,2),(1,2,3,4,5)]);
Group([ (1,2), (1,2,3,4,5) ])
gap> Sym5:=SymmetricGroup(5);
Sym( [ 1 .. 5 ] )
```

GAP sisältää tiedot pienistä ryhmistä (kertaluku ≤ 2000). Kunkin pienen ryhmän tiedot, eli kertaluvun ja indeksin ryhmäkirjastossa, saa funktion `IdGroup` avulla. Nämä tiedot soveltuvat myös pienten ryhmien vertailuun.

```
gap> IdGroup(G);
[ 18, 3 ]
gap> IdGroup(Sym5);
[ 120, 34 ]
gap> IdGroup(Sym5)=IdGroup(S5);
```

```
true
```

Ryhmien radat, kertaluvut ja generaattorit saadaan funktioilla `Orbits`, `Order` ja `GeneratorsOfGroup`.

```
gap> Order(G);Orbits(G);
18
[ [ 1, 2, 3 ], [ 4, 5, 6 ] ]
gap> Orbits(S5);
[ [ 1, 2, 3, 4, 5 ] ]
gap> GeneratorsOfGroup(Sym5);
[ (1,2,3,4,5), (1,2) ]
```

Määritellyn ryhmän aliryhmät voidaan määritellä aliryhmän funktiolla `Subgroup`, jolle annetaan argumentteina ryhmä ja aliryhmän siinä generoivat alkiot. Ryhmän kommutaattorialiryhmän saa komennolla `DerivedSubgroup`.

```
gap> ali:=Subgroup(G,[(1,2),(2,3)]);
Group([ (1,2), (2,3) ])
gap> Order(ali);
6
gap> A5:=DerivedSubgroup(S5);
Group([ (1,2,3), (2,3,4), (3,4,5) ])
gap> Order(A5);
60
```

Konjugointiluokat saadaan seuraavasti.

```
gap> ConjugacyClasses(A5);
[ ()^G, (1,2)(3,4)^G, (1,2,3)^G, (1,2,3,4,5)^G, (1,2,3,5,4)^G ]
gap> ConjugacyClasses(S5);
[ ()^G, (1,2)^G, (1,2)(3,4)^G, (1,2,3)^G, (1,2,3)(4,5)^G, (1,2,3,4)^G,
(1,2,3,4,5)^G ]
gap> ConjugacyClasses(G);
[ ()^G, (4,5,6)^G, (4,6,5)^G, (2,3)^G, (2,3)(4,5,6)^G, (2,3)(4,6,5)^G,
(1,2,3)^G, (1,2,3)(4,5,6)^G, (1,2,3)(4,6,5)^G ]
```

Funktion `Factorization` avulla saadaan permutaatiot esitettyä generaattoreiden avulla. Esimerkiksi ryhmän A_5 alkion (1 5 2) esitys generaattoreiden avulla on

```
gap> GeneratorsOfGroup(A5);
[ (1,2,3), (2,3,4), (3,4,5) ]
gap> Factorization(A5,(1,5,2));
x1^-1*x3^-1*x2
```

jossa $xi, x2$ ja $x3$ ovat generaattorit (1 2 3), (2 3 4) ja (3 4 5). Kun muistetaan, että GAP esittää permutaatiot vasemmalta oikealle, nähdään että (1 5 2) = (2 3 4)(3 5 4)(1 3 2).

Funktiolla `FreeGroup` saadaan luotua joukko kuvaavia sanoja, jotka voidaan yhdistää haluttuihin permutaatioihin homomorfismilla. Tämän homomorfismin avulla nämä kuvaavat sanat saadaan liitettyä alkioiden esityksiin ryhmän generaattoreiden avulla.

```
gap> generaattorit:=FreeGroup("GEN-1", "GEN-2", "GEN-3");
<free group on the generators [ GEN-1, GEN-2, GEN-3 ]>
gap> kuvaus:=GroupHomomorphismByImages(generaattorit, A5,
GeneratorsOfGroup(generaattorit), GeneratorsOfGroup(A5));
[ GEN-1, GEN-2, GEN-3 ] -> [ (1,2,3), (2,3,4), (3,4,5) ]
gap> PreImagesRepresentative(kuvaus, (1,5,2));
GEN-1*GEN-2^-1*GEN-3*GEN-1^-1
```

Siis myös (1 5 2) = (1 3 2)(3 4 5)(2 4 3)(1 2 3). Permutaation tuloesitys ei ole yksiselitteinen eivätkä nämä funktiot anna välttämättä lyhyintä esitystä.

4 Sovellukset

4.1 Korttipakka

Tavallisin korttipakka on sisältää 52 erilaista korttia. Kortit on jaettu neljään maahan joista kussakin on 13 eriarvoista korttia. Toinen puoli kustakin kortista sisältää arvon ilmaisevat merkinnät ja toinen puoli on kaikissa korteissa samanlainen.

4.1.1 Teoreettinen tarkastelu

Korttipakan kaikki kortit ovat erilaisia, joten korttipakan järjestely vastaa symmetrisen ryhmän S_{52} permutaatioita. Kun korttipakan kortit pidetään arvopuoli samaan suuntaan, saadaan korttipakan korteille $52! > 8 \times 10^{67}$ eri järjestystä.

4.1.2 GAP-sovellus

Tarkastellaan korttipakan sekoittamista. Tavallinen tapa sekoittaa kortteja on jakaa pakka keskeltä kahteen yhtäsuureen osaan ja sijoittaa korttipakan kortit limittäin vuorotellen.

Kun limittäminen tehdään siten että alemman puoliskon päällimmäinen kortti tulee pakan ylimmäiseksi, kortit siirtyvät pakassa seuraavan funktion mukaisesti.

```
gap> sekoita1:=function(kortti)
> if kortti<=26 then
> return kortti*2;
> else
> return (kortti-26)*2-1;
> fi;
> end;
function( kortti ) ... end
```

Luodaan lista johon sijoitetaan funktion $sekoita1(x)$ arvot, kun $x = \{1, 2, \ldots, 52\}$.

```
gap> sekoitusLista1:=List([1..52], sekoita1);
[ 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38,
 40, 42, 44, 46, 48, 50, 52, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23,
25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51 ]
```

Kunkin paikan indeksin permutaatiota vastaavilla arvoilla täytetyistä listasta voidaan luoda permutaatio funktion `PermList` avulla.

```
gap> sekoitus1:=PermList(sekoitusLista1);
(1,2,4,8,16,32,11,22,44,35,17,34,15,30,7,14,28,3,6,12,24,48,43,33,13,26,
52,51,49,45,37,21,42,31,9,18,36,19,38,23,46,39,25,50,47,41,29,5,10,20,
40,27)
gap> Order(sekoitus1);
52
```

Tällä tavalla sekoitettaessa voidaan siis saada korteille 52 erilaista järjestystä.

Kun taas limitys tehdään siten, että ylemmän puoliskon päällimmäinen kortti tulee ylimmäiseksi, eli sekä ylin että alin kortti pysyvät sekoituksessa samoina, saadaan vastaavalla tavalla seuraava permutaatio.

```
gap> sekoita2:=function(kortti)
> if kortti<=26 then
> return kortti*2-1;
> else
> return (kortti-26)*2;
> fi;
> end;
function( kortti ) ... end
gap> sekoitusLista2:=List([1..52], sekoita2);
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37,
39, 41, 43, 45, 47, 49, 51, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24,
26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52 ]
gap> sekoitus2:=PermList(sekoitusLista2);
(2,3,5,9,17,33,14,27)(4,7,13,25,49,46,40,28)(6,11,21,41,30,8,15,29)
(10,19,37,22,43,34,16,31)(12,23,45,38,24,47,42,
32)(18,35)(20,39,26,51,50,48,44,36)
gap> Order(sekoitus2);
8
```

Tässä tilanteessa erilaisia järjestyksiä saadaan vain kahdeksan, joten se on epäkäy-

tännöllinen, jos kortit halutaan sekoittaa, mutta se voi olla hyödyllinen, jos halutaan aikaansaada sama järjestys aina kahdeksan sekoituksen jälkeen.

Luodaan nyt permutaatioryhmä käyttäen generaattoreina kahta edellä kuvattua permutaatiota.

```
gap> sekoitukset:=Group(sekoitus1,sekoitus2);
<permutation group with 2 generators>
gap> Order(sekoitukset);
27064431817106664380040216576000000
gap> LogInt(Order(sekoitukset),10);
34
```

Näitä kahta sekoitustapaa satunnaisesti vuorotellen voidaan siis saada yli 2×10^{34} eri järjestystä. Nämä permutaatiot riittänevät ainakin satunnaisesti pelaavien tarpeisiin.

Tutkitaan vielä kuinka suuri osa tämä on kaikista mahdollisista korttijärjestyksistä.

```
gap> Factorial(52);
80658175170943878571660636856403766975289505440883277824000000000000
gap> last/Order(sekoitukset);
29802279137433108747262291939218 75
gap> LogInt(last,10);
33
```

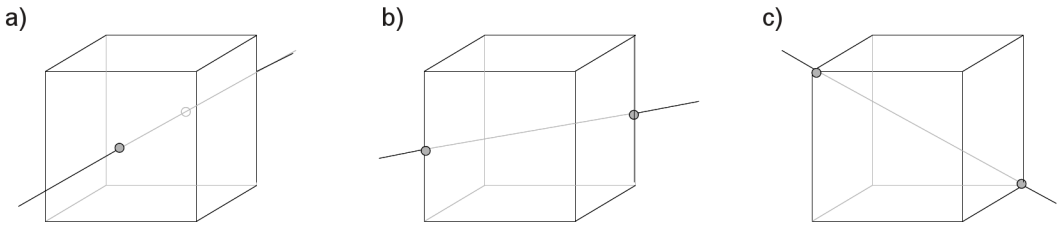
Mahdollisista järjestyksistä käytetään vain alle $\frac{1}{10^{33}}$, kun pakka jaetaan aina tasan puoliksi ja puoliskojen kortit limitetään vuorotellen. Vaikka siinäkin osassa on riittävästi järjestysvaihtoehtoja normaaleihin tarpeisiin, yleensä päädytään myös ryhmän ulkopuolisiin permutaatioihin, mikäli pakka halutaan sekoittaa eikä tasan jakamisen tai vuorotellen limittämisen suhteen olla tarkkoja. Tavallisesti korteilla myös pelataan silloin tällöin sekoitusten välissä, ja pelin jälkeen kortteja kerättäessä kortit päätyvät hyvin todennäköisesti ryhmän ulkopuoliseen järjestykseen.

4.2 Kuution tahkojen väritys

4.2.1 Teoreettinen tarkastelu

Tarkastellaan kuutiota jonka kukin tahko on väritetty jollain N eri väristä. Jos kuution tahkot voidaan erottaa toisistaan, eli ne on merkitty esimerkiksi numeroin, erilaisia kuution värityksiä on N^6 kappaletta.

Oletetaan tästä eteenpäin, ettei kuution tahkoja voida erottaa toisistaan, vaan kuutioiden A ja B väritys on sama, mikäli kuutioiden väritys saadaan vastaamaan toisiaan kuutioita kääntelemällä.



Kuva 1: Kuution symmetria-akselit

Kuution permutaatiot voidaan jakaa viiteen eri luokkaan.

I) Identiteettikuvauksia on vain yksi ja se säilyttää jokaisen N^6 mahdollisen kuution värityksen sellaisenaan.

II) 90 asteen kierto vastakkaisten tahkojen keskipisteiden suhteen (Kuva 1 a). Tällaisia permutaatioita on 6 erilaista ja niistä jokainen säilyttää kääntöakselin määrittelevien tahkojen värityksen ja muiden tahkojen värityksen vain mikäli kaikki neljä muuta tahkoa ovat samanvärisiä. Tällaisia kuution värityksiä on N^3 kappaletta.

III) 180 asteen kierto vastakkaisten tahkojen keskipisteiden suhteen (Kuva 1 a). Tällaisia permutaatioita on 3 erilaista ja niistä jokainen säilyttää kääntöakselin määrittelevien tahkojen värityksen ja muiden tahkojen värityksen vain mikäli niiden vastakkaisilla puolilla olevat parit ovat samanvärisiä. Tällaisia kuution värityksiä on N^4 kappaletta.

IV) 120 asteen kierto kuution vastakkaisten kulmien suhteen (Kuva 1 c). Tällaisia permutaatioita on 8 erilaista ja ne säilyttävät värityksen mikäli niiden kulmien, joiden suhteen kierto tehdään, viereiset tahkot ovat samanvärisiä. Tällaisia värityksiä on N^2 kappaletta.

V) 180 asteen kierto vastakkaisten särmien keskipisteiden suhteen (Kuva 1 b). Tällaisia permutaatioita on 6 erilaista ja ne säilyttävät niiden kuutioiden värityksen, joiden ne sivut joiden kummankin kierron määrittelevän särmän viereiset tahkot ovat samanvärisiä, sekä kaksi muuta tahkoa ovat samanvärisiä. Tällaisia värityksiä on N^3 kappaletta.

Yhteensä permutaatiota on 24 ja erilaisia värityksiä on Burnsiden lemman (Lemma 1.35) nojalla

$$\frac{N^6 + 6N^3 + 3N^4 + 8N^2 + 6N^3}{24} = \frac{N^6 + 3N^4 + 12N^3 + 8N^2}{24}. \tag{4}$$

4.2.2 GAP-sovellus

Luodaan permutaatioryhmä käyttäen generaattoreina kahta erisuuntaista edellisen luvun II-tyyppistä permutaatiota, eli 90 asteen rotaatiota.

```
gap> kuutio:=Group((1,2,3,4),(1,5,3,6));
Group([ (1,2,3,4), (1,5,3,6) ])
gap> Order(kuutio);
24
```

Todetaan että kuution permutaatioryhmä on isomorfinen symmetrisen ryhmän S_4 kanssa.

```
gap> S4:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> IdGroup(kuutio)=IdGroup(S4);
true
```

Jaetaan kuution permutaatioryhmä konjugointiluokkiin ja valitaan kustakin luokasta edustaja.

```
gap> luokat:=ConjugacyClasses(kuutio);
[ ()^G, (2,4)(5,6)^G, (2,5,4,6)^G, (1,2)(3,4)(5,6)^G, (1,2,5)(3,4,6)^G ]
gap> ed:=List(luokat,Representative);
[ (), (2,4)(5,6), (2,5,4,6), (1,2)(3,4)(5,6), (1,2,5)(3,4,6) ]
```

Tutkitaan kuinka monta väritystä kukin konjugointiluokan edustaja säilyttää, kun käytössä on kaksi eri väriä, ja lasketaan Burnsiden lemman avulla erilaisten väritysten kokonaismäärä.

```
gap> tahkot:=[[1],[2],[3],[4],[5],[6]];
[ [ 1 ], [ 2 ], [ 3 ], [ 4 ], [ 5 ], [ 6 ] ]
gap> sailyy:=List(ed,i->2^Length(Orbits(Group(i),tahkot,OnSets)));
[ 64, 16, 8, 8, 4 ]
gap> Sum(List([1..5],i->Size(luokat[i])*sailyy[i]))/Size(kuutio);
10
```

Väritysten määräksi saatu 10 on sama joka saadaan kaavaa 4 käyttämällä.

Yhdistetään edellä lasketut operaatiot funktioksi, jonka ainoana argumenttina on käytössä olevien värien määrä.

```
gap> variaatiot:=function(varienMaara)
> local kuutio, tahkot, luokat, ed, sailyy, summa;
> kuutio:=Group((1,2,3,4),(1,5,3,6));
> tahkot:=[[1],[2],[3],[4],[5],[6]];
> luokat:=ConjugacyClasses(kuutio);
> ed:=List(luokat,Representative);
> sailyy:=List(ed,i->varienMaara^Length(Orbits(Group(i),tahkot,OnSets)));
> summa:=Sum(List([1..5],i->Size(luokat[i])*sailyy[i]))/Size(kuutio);
> return summa;
> end;
function( varienMaara ) ... end
```

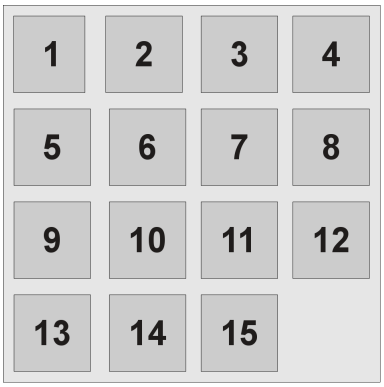
Lasketaan vielä kuutioiden arvot värien määrillä yhdestä kymmeneen.

```
gap> for i in [1..10] do
> Print(variaatiot(i),"\n");
> od;
1
10
57
240
800
2226
5390
11712
23355
43450
```

Tulokset ovat samat kuin kaavalla 4 laskettaessa.

4.3 15-puzzle

15-puzzle on logiikkapeli, jossa 15 numeroa on sijoitettu 4×4 ruudukkoon, ja niitä voidaan liikuttaa yksi kerrallaan siten että jokin tyhjän paikan viereisistä numeroista siirretään tyhjänä olleeseen paikkaan. Tavoitteena on saada numerot järjestettyä ruudukkoon alla kuvatulla tavalla.



Kuva 2: 15-puzzle

Alunperin pelilauta oli pahvinen tai puinen laatikko, jossa oli puisia liikuteltavia kuutioita. Tällaisen pelin kuutiot oli helppo järjestää haluttuun järjestykseen. Myöhemmät pelin toteutukset ovat muovisia ja palat on uritettu, joten niitä ei voi irroittaa ja järjestää yhtä helposti.

4.3.1 Historiaa

15 puzzle tuli markkinoille Yhdysvaltojen itärannikolla loppuvuodesta 1879 ja aiheutti suuren villityksen eri puolilla maailmaa vuoden 1880 aikana. Pelin suosio perustui ennen kaikkea siihen, että puolet sen tiloista se on suhteellisen yksinkertaista ratkaista, mutta loppujen tilojen ratkaisun mahdottomuudesta ei tuolloin ollut suuren yleisön keskuudessa yksimielisyyttä.[11]

Monet kertoivat ratkaisseensa pelin tilasta, jossa muut numerot olivat oikeilla paikoillaan, mutta numerot 14 ja 15 oli vaihdettu keskenään. Ensimmäisen julkisen palkkion ko. tehtävän ratkaisusta lupasi vuoden 1880 tammikuun 24. päivä tohtori Charles K. Pevey. Myöhemmin hän kertoi luvanneensa palkkion, koska eräs nuori nainen oli vakuuttunut ratkaisseensa ongelman, ja koska nuoren naisen sanoja ollut kohteliasta epäillä, hän tarvitsi keinon vakuuttaa hänet siitä, ettei ratkaisu ollut mahdollinen. Tohtori Pevey tarvitsi

siis koko kaupungin apua saadakseen nuoren naisen, joka tiesi ratkaisseensa ongelman, vakuuttumaan siitä, ettei tehtävän ratkaiseminen ollut mahdollista.[11]

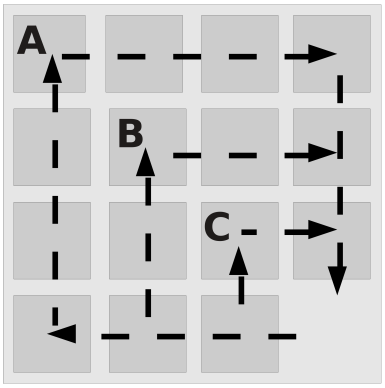
4.3.2 Teoreettinen tarkastelu

Jo joulukuussa 1879 julkaistiin myös artikkelipari[6], jossa todistettiin, että numeropalat voidaan järjestää palasia laudalla liikuttamalla vain parillisiin permutaatioihin joissa tyhjä paikka on sama kuin alkutilassa. Pelin sellaiset tilat, joissa tyhjä paikka on oikeassa alalaidassa ovat siis alternoivan ryhmän A_{15} sisältämät permutaatiot. Sellaisia tiloja on $\frac{1}{2}15!$ kappaletta. Jos tyhjän paikan 16 eri paikkaa otetaan huomioon, eri tiloja on $\frac{1}{2}16!$.

1970-luvulla Richard M. Wilson todisti yleisemmän teorian verkoille, jossa on yksi tyhjä paikka, jonka kanssa viereisten solmujen tile voidaan vaihtaa. Sen mukaan kaikkien tällaisten n solmun verkkojen tilojen lukumäärä, yhtä poikkeusta ja syklejä lukuunottamatta, on joko $n!$ tai $\frac{1}{2}n!$.

4.3.3 GAP-ohjelmiston soveltaminen tehtävään

Tutkitaan 15-puzzlen sellaisia järjestyksiä, joissa tyhjä paikka on oikeassa alakulmassa. Tällaisia järjestyksiä ovat kaikki alternoivan ryhmän A_{15} permutaatiot. Kokeillaan generoivatko Kuvan 3 mukaiset permutaatiot A, B ja C kaikkien 15-puzzlen permutaatioiden ryhmän.



Kuva 3: Kolme käytettävää permutaatiota

```
gap> A15:=DerivedSubgroup(SymmetricGroup(15));
<permutation group with 13 generators>
gap> puzzle:=Group((15,14,13,9,5,1,2,3,4,8,12),(15,14,10,6,7,8,12)
,(15,11,12));
Group([ (1,2,3,4,8,12,15,14,13,9,5), (6,7,8,12,15,14,10), (11,12,15) ])
gap> Order(puzzle);
653837184000
gap> Order(puzzle)=Order(A15);IsSubgroup(puzzle,A15);
true
true
```

Ryhmä on siis alternoivan ryhmän A_{15} aliryhmä jonka kertaluku on $|A_{15}|$, joten kyseessä on sama ryhmä.

Luodaan uusi ryhmä F, jonka generaattorit ovat A, B ja C, sekä homomorfismi $F \to puzzle$.

```
gap> F:=FreeGroup("A","B","C");
<free group on the generators [ A, B, C ]>
gap> homom:=GroupHomomorphismByImages(F, puzzle, GeneratorsOfGroup(F),
GeneratorsOfGroup(puzzle)); [ A, B, C ] -> [ (1,2,3,4,8,12,15,14,13,9,5),
 (6,7,8,12,15,14,10), (11,12,15) ]
```

Tutkitaan, millaisella permutaatioiden A, B ja C yhdistelmällä voidaan ratkaista 15-puzzle, joka on muuten järjestetty, paitsi palat 1 ja 6 on vaihdettu keskenään.

```
gap> alkutila:=(1,6);
(1,6)
gap> polku:=PreImagesRepresentative(homom, alkutila);
C*A*C*A^-1*B*A*C^-1*A^-1*C^-1*B*A*C^-1*B^-1*C^-1*A^-1*B^-1*A*B^-1*A^-1*B
^-1*A^-1*B*A*B*A^2*B^-1*A^-2*B^-3*A^-2*B*A^
2*B^2*A^3*B^-3*A^-3*B^-1*A^-3*B*A^3*B^-2*A^4*B^-2
gap> polku:=PreImagesRepresentative(homom, alkutila);
C*A*C*A^-1*B*A*C^-1*A^-1*C^-1*B*A*C^-1*B^-1*C^-1*A^-1*B^-1*A*B^-1*A^-1*B
^-1*A^-1*B*A*B*A^2*B^-1*A^-2*B^-3*A^-2*B*A^
2*B^2*A^3*B^-3*A^-3*B^-1*A^-3*B*A^3*B^-2*A^4*B^-2
```

```
gap> ratkaisu:=polku^-1;
B^2*A^-4*B^2*A^-3*B^-1*A^3*B*A^3*B^3*A^-3*B^-2*A^-2*B^-1*A^2*B^3*A^2*B*A
^-2*B^-1*A^-1*B^-1*A*B*A*B*A^-1*B*A*C*B*C*A^
-1*B^-1*C*A*C*A^-1*B^-1*A*C^-1*A^-1*C^-1
gap> Length(ratkaisu);
64
```

Ratkaisu ei ole lyhyin mahdollinen, eikä välttämättä edes lyhyin kolmella esitetyllä permutaatiolla saatava.

Kokeillaan vielä löytää ratkaisu vuonna 1880 paljon kiinnostusta osakseen saaneeseen tehtävään, jossa alkutilasta on vaihdettu palat 14 ja 15 keskenään. Tutkitaan siis permutaatiota (14 15) samassa ryhmässä.

```
gap> PreImagesRepresentative(homom, (14,15));
fail
```

Esittäminen generaattoreiden avulla epäonnistuu, koska ko. permutaatio on esitettävissä vain parittomalla määrällä viereisten paikkojen välisiä transpooseja, eikä täten sisälly ryhmään.

4.4 Rubikin kuutio

4.4.1 Historiaa

Tammikuun 30. päivä vuonna 1975 unkarilainen arkkitehtuurin professori Ernő Rubik sai patentin keksimälleen avaruudelliselle logiikkapelille (térbeli logikai játék) [2]. Rubikin kuutioksi myöhemmin nimetty logiikkapeli aiheutti 1970- ja 1980-lukujen taitteessa suurimman logiikkapelivillityksen sitten 15-puzzlen. Kuutioita on myyty maailmanlaajuisesti satoja miljoonia kappaleita.

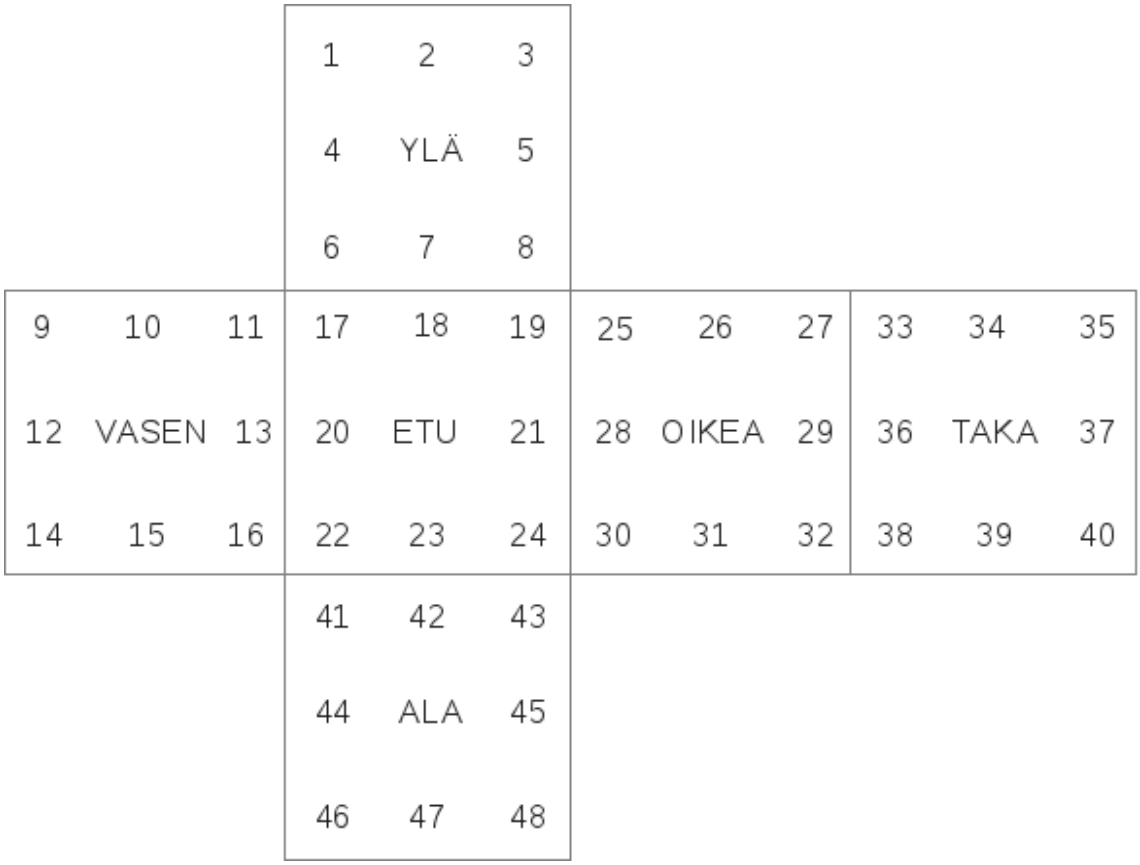
4.4.2 Teoreettinen tarkastelu

Rubikin kuutio on ulkopinnaltaan väritetyistä kuutiosta muodostuva $3 \times 3 \times 3$ kuutio. Pienempien kuutioiden näkyvissä olevat pinnat on väritetty kuudella eri värillä siten, että alkutilassaan kaikki suuren kuution samalla tahkolla sijaitsevat pinnat ovat keskenään samanvärisiä, ja erivärisiä kuin muilla tahoilla sijaitsevat pinnat.

Kaikkia samassa tasossa sijaitsevien osakuutioiden muodostamia $3 \times 3 \times 1$ kokoisia blokkeja voidaan pyörittää niiden keskipisteen ympäri, jolloin väritetyt pinnat siirtyvät koko kuution pinnalta toiselle. Tutkitaan näitä permutaatioita ja niiden avulla pinnoille muodostettuja väriyhdistelmiä. Koska pintojen keskimmäiset osapinnat pysyvät paikoillaan toistensa suhteen, voidaan muiden osapintojen paikat määritellä niiden mukaan. Numeroidaan osapinnat Kuvan 4 osoittamalla tavalla.

Valitaan käytettäviksi permutaatioiksi ne permutaatiot, jotka saadaan katsomalla kuutiota kunkin kuuden tahkon suunnasta ja kääntämällä ko. tahkon osakuutioiden muodostamaa osaa katselusuunnassa 90 astetta myötäpäivään. Merkitään näitä permutaatioita Kuvassa 4 tahkoille annettujen nimien alkukirjainten mukaan. Tällöin saadaan seuraavat permutaatiot.

$$
\begin{aligned}
E &= (6\ 25\ 43\ 16)(7\ 28\ 42\ 13)(8\ 30\ 41\ 11)(17\ 19\ 24\ 22)(18\ 21\ 23\ 20)\\
T &= (1\ 14\ 48\ 27)(2\ 12\ 47\ 29)(3\ 9\ 46\ 32)(33\ 35\ 40\ 38)(34\ 37\ 39\ 36)\\
O &= (3\ 38\ 43\ 19)(5\ 36\ 45\ 21)(8\ 33\ 48\ 24)(25\ 27\ 32\ 30)(26\ 29\ 31\ 28)\\
V &= (1\ 17\ 41\ 40)(4\ 20\ 44\ 37)(6\ 22\ 46\ 35)(9\ 11\ 16\ 14)(10\ 13\ 15\ 12)\\
A &= (14\ 22\ 30\ 38)(15\ 23\ 31\ 39)(16\ 24\ 32\ 40)(41\ 43\ 48\ 46)(42\ 45\ 47\ 44)\\
Y &= (1\ 3\ 8\ 6)(2\ 5\ 7\ 4)(9\ 33\ 25\ 17)(10\ 34\ 26\ 18)(11\ 35\ 27\ 19)
\end{aligned}
\tag{5}
$$



Kuva 4: Kuution pintojen numerointi

Kuutiota tarkastelemalla voidaan helposti havaita, että mikä tahansa kuvan numeroiduista pinnoista on siirrettävissä näillä permutaatioilla mille tahansa suuren kuution tahkoista. koska lisäksi kunkin tahkon kulmapalat voidaan permutoida keskenään kyseistä tahkoa kääntämällä, kaikki kulmapalojen pinnat kuuluvat samaan rataan. Samanlaisella tarkastelulla saadaan toiseksi radaksi kaikkien kahden näkyvän pinnan reunapalojen muodostama joukko.

4.4.3 Tarkastelu GAP-ohjelmiston avulla

Tämä tarkastelu pohjautuu suurilta osin GAP-ohjelmiston dokumentaatiosta[4] löytyvään esimerkkiin "Analyzing Rubik's Cube with GAP" (`http://www.gap-system.org/Doc/Examples/rubik.html`). Ko. esimerkissä tarkastellaan myös muita Rubikin kuution permutaatio-ominaisuuksia kuin seuraavassa esiteltävät.

Luodaan Rubikin kuution permutaatioryhmä syöttämällä sen virittävät kuusi permutaatiota. Nämä permutaatiot ovat samat kuin edellisen kappaleen permutaatiojoukossa (5) esitellyt.

```
gap> rubik:=Group(
> ( 1, 3, 8, 6)( 2, 5, 7, 4)( 9,33,25,17)(10,34,26,18)(11,35,27,19),
> ( 9,11,16,14)(10,13,15,12)( 1,17,41,40)( 4,20,44,37)( 6,22,46,35),
> (17,19,24,22)(18,21,23,20)( 6,25,43,16)( 7,28,42,13)( 8,30,41,11),
> (25,27,32,30)(26,29,31,28)( 3,38,43,19)( 5,36,45,21)( 8,33,48,24),
> (33,35,40,38)(34,37,39,36)( 3, 9,46,32)( 2,12,47,29)( 1,14,48,27),
> (41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40));
<permutation group with 6 generators>
```

Tutkitaan permutaatioiden määrää ja ratoja.

```
gap> Size(rubik);
43252003274489856000
gap> Collected(Factors(Size(rubik)));
[ [ 2, 27 ], [ 3, 14 ], [ 5, 3 ], [ 7, 2 ], [ 11, 1 ] ]
gap> radat:=Orbits(rubik);
[ [ 1, 3, 17, 14, 8, 38, 9, 41, 19, 48, 22, 6, 30, 33, 43, 11, 46, 40,
24, 27, 25, 35, 16, 32 ],
  [ 2, 5, 12, 7, 36, 10, 47, 4, 28, 45, 34, 13, 29, 44, 20, 42, 26, 21,
  37, 15, 31, 18, 23, 39 ] ]
```

Rubikin kuutiossa on siis $43252003274489856000 = 2^{27} \times 3^{14} \times 5^3 \times 7^2 \times 11$ eri permutaatiota. Alkiot jakaantuvat kahteen erilliseen rataan, eli kuution kulmissa ja sivuilla oleviin osapintoihin.

Seuraavaksi määritellään ryhmä F, jonka generaattoreiksi määritellään kuution tahkojen lyhenteet, ja homomorfismi hom, joka kuvaa ryhmän F generaattorit kuution permutaatioryhmän generaatoreiksi, jolloin kirjaimet vastaavat kyseisen tahkon 90 asteen rotaatiota tahkon suunnasta katsoen.

```
gap> F := FreeGroup("Y","V","E","O","T","A");
<free group on the generators [ Y, V, E, O, T, A ]>
```

```
gap> hom:=GroupHomomorphismByImages(F, rubik, GeneratorsOfGroup(F),
GeneratorsOfGroup(rubik));
[ Y, V, E, O, T, A ] ->
[ (1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19),
(1,17,41,40)(4,20,44,37)(6,22,46,35)(9,11,16,14)(10,13,15,12),
(6,25,43,16)(7,28,42,13)(8,30,41,11)(17,19,24,22)(18,21,23,20),
(3,38,43,19)(5,36,45,21)(8,33,48,24)(25,27,32,30)(26,29,31,28),
(1,14,48,27)(2,12,47,29)(3,9,46,32)(33,35,40,38)(34,37,39,36),
(14,22,30,38)(15,23,31,39)(16,24,32,40)(41,43,48,46)(42,45,47,44) ]
```

Seuraavaksi luodaan satunnainen rubikin kuution permutaatio *randperm*. Tähän permutaatioon johtava polku esitetään homomorfismin *hom* alkukuvien, eli edellä määriteltyjen 90 asteen rotaatioiden avulla. Tämän polun käänteisarvona saadaan lista rotaatioista, joilla *randperm* voidaan palauttaa kuution alkutilaan.

```
gap> randperm:=Random(rubik);
(1,19,16,14,32,35,25,41,40,48,9,8,22,46,38)(2,34)(3,33,27)
(4,12,47,21,45,29,42,26,44,7,20,10,37,39,28,31,36,23,5,15,18,13)
gap> polku:=PreImagesRepresentative(hom,randperm);
V^-1*Y^-1*V*E*O*Y*O^-1*E^-1*V*Y*E*Y^-1*E^-1*V^-1*Y*V*Y^2*V^-1*Y^-1*V*Y^-1
*V^-1*Y^3*V^-1*Y^-1*T*V^-1*T^-1*V*Y*V*Y*V*Y*E*Y^-1*E^-1*V^-1*Y^-1*E*Y
*E^-1*Y^-1*V^-1*Y^-1*V*Y*V^-1*Y*E^-1*V^-1*E*V*Y^-1*V*Y^2*V^2*Y*V*E^-1*Y^2
*E*V^-1*A^-1*V^-1*A*E^-1*A*O*A^-1*T^-1*O*T*E^-1*O^-1*Y^-1*A^-1*O
gap> Length(polku);
84
gap> ratkaisu:=polku^-1;
O^-1*A*Y*O*E*T^-1*O^-1*T*A*O^-1*A^-1*E*A^-1*V*A*V*E^-1*Y^-2*E*V^-1*Y^-1
*V^-2*Y^-2*V^-1*Y*V^-1*E^-1*V*E*Y^-1*V*Y^-1*V^-1*Y*V*Y*E*Y^-1*E^-1*Y*V*E
*Y*E^-1*Y^-1*V^-1*Y^-1*V^-1*Y^-1*V^-1*T*V*T^-1*Y*V*Y^-3*V*Y*V^-1*Y*V*Y^-2
*V^-1*Y^-1*V*E*Y*E^-1*Y^-1*V^-1*E*O*Y^-1*O^-1*E^-1*V^-1*Y*V
```

Tämä ratkaisu ei kuitenkaan ole, 84 pituisena, lyhyin mahdollinen. Vuonna 2010 Tomas Rokicki, Herbert Kociemba, Morley Davidson ja John Dethridge kävivät, Googlen lahjoittaman laskentatehon avulla, läpi kaikki mahdolliset Rubikin kuution permutaatiot ja osoittivat, että kunkin niistä optimiratkaisu on enintään 20 siirtoa [10]. Kuution tilojen jakauma suhteessa niiden etäisyyteen alkutilasta on esitetty Taulukossa 1.

Ensimmäinen permutaatio joka vaati 20 siirtoa ratketakseen, löydettiin vuonna 1995 [9]. Yllä olevassa ratkaisussa on myös 90 asteen rotaation kolmas potenssi, joka myös osoittaa, ettei yo. ratkaisu varmasti ole optimaalinen, koska kolmannen potenssin voisi korvata potenssilla -1, ja siten lyhentää siirtoketjun pituutta.

Taulukko 1: Rubikin kuution tilojen jakautuminen suhteessa etäisyyteen alkutilasta [10]

Etäisyys alkutilasta	Eri tilojen määrä
0	1
1	18
2	243
3	3 240
4	43 239
5	574 908
6	7 618 438
7	100 803 036
8	1 332 343 288
9	17 596 479 795
10	232 248 063 316
11	3 063 288 809 012
12	40 374 425 656 248
13	531 653 418 284 628
14	6 989 320 578 825 358
15	91 365 146 187 124 313
16	noin 1 100 000 000 000 000 000
17	noin 12 000 000 000 000 000 000
18	noin 29 000 000 000 000 000 000
19	noin 1 500 000 000 000 000 000
20	noin 300 000 000

Viitteet

[1] Archer, Aaron F.: *A Modern Treatment of the 15-Puzzle.* American Mathematical Monthly, 106:793–799, 1999.

[2] Bandelow, C.: *Inside Rubik's Cube and Beyond.* Birkhäuser, 1982.

[3] Fraleigh, J.B. ja V.J. Katz: *A First Course in Abstract Algebra.* Addison-Wesley World Student Series. Addison-Wesley, 7. painos, 2003.

[4] The GAP Group: *GAP – Groups, Algorithms, and Programming, Version 4.5.6*, 2012. `http://www.gap-system.org`.

[5] Herstein, I.N.: *Abstract Algebra.* Prentice-Hall, 3. painos, 1996.

[6] Johnson, Wm. Woolsey ja William E. Story: *Notes on the "15"Puzzle.* American Journal of Mathematics, 2(4):397–404, 1879.

[7] Joyner, D.: *Adventures in Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys.* Adventures in Group Theory. Johns Hopkins University Press, 2008.

[8] Metsänkylä, T. ja M. Näätänen: *Algebra.* Yliopistopaino, 2. painos, 2009.

[9] Reid, Michael: *Cube Lovers: superflip requires 20 face turns*, tammikuu 1995. `http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/michael_reid__superflip_requires_20_face_turns.html`, vierailtu 25.12.2012 .

[10] Rokicki, Tomas, Herbert Kociemba, Morley Davidson ja John Dethridge: *God's Number is 20*, heinäkuu 2010. `http://www.cube20.org`, vierailtu 25.12.2012 .

[11] Slocum, J. ja D. Sonneveld: *The 15 Puzzle: How It Drove the World Crazy; The Puzzle That Started the Craze of 1880; How Amercia's Greatest Puzzle Designer, Sam Loyd, Fooled Everyone for 115 Years.* Slocum Puzzle Foundation, 2006.

[12] Wilson, Richard M: *Graph puzzles, homotopy, and the alternating group.* Journal of Combinatorial Theory, Series B, 16(1):86–96, 1974.