VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATIKOS IR INFORMATIKOS METODIKOS KATEDRA

Sigita Mažvylaitė

REKURENTINIAI SĄRYŠIAI IR JŲ TAIKYMAI

Magistro baigiamasis darbas

Vadovas
Prof. dr. Eugenijus Stankus

Leidžiu ginti: ____________________
(Vadovo parašas)

VILNIUS 2011

TURINYS

ĮVADAS ... 3
1. REKURENTINIO SĄRYŠIO SĄVOKA, PAVYZDŽIAI ... 5
2. REKURENTINIŲ SĄRYŠIŲ SPRENDIMAS ... 10
2.1. Homogeninių rekurentinių sąryšių sprendimas ... 11
2.2. Nehomogeninių rekurentinių sąryšių sprendimas ... 15
2.3. Atskiro tiesinio nehomogeninio rekurentinio sąryšio su pastoviais koeficientais sprendinio parinkimas ... 16
3. REKURENTINIAI SĄRYŠIAI IR GENERUOJANČIOS FUNKCIJOS ... 20
4. REKURENTINIŲ SĄRYŠIŲ TAIKYMAS ... 24
4.1. Integralų skaičiavimuose ... 24
4.2. Geometrijoje ... 28
4.3. Kombinatorikos ir tikimybių teorijos uždaviniuose ... 30
4.4. Sudėtinės palūkanos. Diskontavimas. ... 32
5. PRAKTINĖ DALIS ... 35
IŠVADOS ... 37
REZIUMĖ ... 38
LITERATŪROS SĄRAŠAS ... 39
Priedas Nr.1 ... 41
Priedas Nr. 2 ... 42

ĮVADAS

Šiame baigiamajame darbe nagrinėjamos pagrindinės rekurentinio sąryšio sąvokos, rekurentinių sąryšių sprendimas, jų ryšys su generuojančiomis funkcijomis. Aptariamos rekurentinių sąryšių taikymo matematikoje, informatikoje (tiksliau, algoritmavime bei programavime) ir ekonomikoje galimybės. Darbe, naudojantis „JAVA" programavimo kalba, sukurtos programa kai kurių uždavinių sprendimų paspartinimui bei programa jų skaičiavimo principų grafiniam vaizdavimui.

Temos aktualumas. Rekurentinių sąryšių taikymas matematikoje ir informatikoje yra labai platus. Sprendžiant daugelį kombinatorikos uždavinių naudojantis rekurentiniu sąryšiu, uždavinį su n objektų galime pakeisti uždaviniu su (n - 1) objektu, o šį – uždaviniu su (n - 2) objektu ir t.t. Paeiliui mažindami objektų skaičių, galų galiausiai gauname uždavinį, kurį nėra sudėtinga išspręsti. Rekurentinių sąryšių taikymas palengvina ir supaprastina uždavinių sprendimą, todėl jų naudojimas priimtinas daugeliui. Programavime rekurentiniai sąryšiai (rekursija) yra apibūdinama kaip funkcija, kuri kreipiasi pati į save. Ji naudinga, kai reikia programuoti veiksmus su nežinomu operacijų skaičiumi, įgalina supaprastinti algoritmo formulavimą. Apskritai rekursija neretai sutinkama įvairiose gyvenimo srityse. Pavyzdžiui, televizoriaus ekrane matome tą patį televizorių, kuriame vėl rodomas televizorius ir t. t. Rekurentinių sąryšių taikymą ir jo aktualumą pagrindžia ir nuolatinis jų naudojimas bankuose: sudėtinių palūkanų skaičiavimas, diskontavimas ir pan.

Tikslai. Pirmasis šio darbo tikslas – apžvelgti ir išnagrinėti bendrąją rekurentinių sąryšių teoriją. Šiam tikslui pasiekti, naudojant skirtingus literatūros šaltinius, nagrinėjama:

- pagrindinės rekurentinių sąryšių sąvokos, pavyzdžiai;
- rekurentinių sąryšių sprendimas: homogeninių ir nehomogeninių sąryšių sprendimas, atskiro tiesinio nehomogeninio rekurentinio sąryšio su pastoviais koeficientais sprendinio parinkimas, pavyzdžiai;
- rekurentiniai sąryšiai ir generuojančios funkcijos, jų ryšys, pavyzdžiai.

Antrasis šio darbo tikslas – išanalizuoti rekurentinių sąryšių taikymą šiose srityse:

- integralų skaičiavimuose;
- kombinatorikos ir tikimybių teorijos uždaviniuose;
- geometrijoje;
- ekonomikoje: sudėtinių palūkanų skaičiavimas; diskontavimas.

Trečiasis šio darbo tikslas – naudojantis „Java“ programavimo kalba, sukurti programą, kurią taikant būtų galima suskaičiuoti daugiakampio K_n, gautą iš n viršūnių (taškų), iš kurių kiekvienos nubrėžtos atkarpos į kiekvieną kitą viršūnę, briaunų skaičių; taip pat grafiškai pavaizduoti uždavinio sprendimą.

Rezultatai. Rašant magistro baigiamąjį darbą buvo pasiekti suformuluoti tikslai, kurių rezultatai, atitinkamai panaudoti, gali turėti aiškią vertę: tai yra išsamiai paruošta teorinė medžiaga apie rekurentinius sąryšius bei jų taikymus, kuri yra naudinga sprendžiant kombinatorikos, diskrečiosios matematikos, tikimybių teorijos ir kitokius uždavinius.

Darbo struktūra. Baigiamąjį darbą sudaro įvadas, penki skyriai, išvados ir literatūros sąrašas, kurį sudaro 25 šaltiniai. Bendra darbo apimtis – 40 puslapių. Darbo pabaigoje - du priedai, kuriuose pateikiamos praktinės dalies programos.

1. REKURENTINIO SĄRYŠIO SĄVOKA

Sprendžiant daugelį kombinatorikos, diskrečiosios matematikos, tikimybių teorijos, ekonomikos, geometrijos ir kitokių uždavinių, yra naudojami rekurentiniai sąryšiai[1]. Programavime užrašomos rekursijos, kurios leidžia supaprastinti algoritmo formulavimą. Taip pat jų taikymą galima pastebėti loginių žaidimų ir galvosūkių sprendimuose.

Šiame skyrelyje susipažinsime su pagrindinėmis rekurentinio sąryšio sąvokomis, pateiksime pavyzdžius.

Apibrėžimas. *k-osios eilės tiesiniu rekurentiniu sąryšiu vadinsime formulę, siejančią skaičių a_n su skaičiai $a_{n-1}, a_{n-2}, \ldots, a_{n-k}$:*

$$c_0(n)a_n + c_1(n)a_{n-1} + \ldots + c_k(n)a_{n-k} = \varphi(n),\ n \geq k, \qquad 1.1$$

čia $c_0(n), c_1(n), \ldots, c_k(n)$ ir $\varphi(n)$ yra natūraliojo argumento funkcijos ir $c_0(n) \neq 0$, $c_k(n) \neq 0$.

Apibrėžimas. Tam, kad sąryšis (1.1) generuotų skaičių seką, reikia k pradinių reikšmių: $a_0, a_1, \ldots, a_{k-1}$. Jei $c_0(n), \ldots, c_k(n)$ yra konstantos, tai sakome, kad turime ***tiesinį rekurentinį sąryšį su pastoviais koeficientais***.

Jei $\varphi(n) = 0$, tai rekurentinis sąryšis vadinamas ***homogeniniu***, priešingu atveju – ***nehomogeniniu***.

Pavyzdžiui:

a) $a_n - a_{n-} + 5a_{n-} + a_{n-} - 3a_{n-} = 0$; d) $a_n - a_{n-} + 4a_{n-} - 1a_{n-} = 1n$;

b) $4a_n - 7a_{n-} + a_{n-} = n^2 +$; e) $a_n - 6(a_{n-1})^2 + 7a_{n-2} =$;

c) $a_n - (n-\ )a_{n-} - (n-\ )a_{n-} = 0$; f) $a_n^2 + (a_{n-})^2 = -$.

Turime, kad

1. a), b), c), d) rekurentinės išraiškos yra tiesinės. Išraiškos e) ir f) nėra tiesinės, kadangi turi narį $(a_{n-})^2$.
2. a), b) ir d) yra tiesiniai sąryšiai su pastoviais koeficientais.
3. a) ir c) sąryšiai yra homogeniniai, o b) ir d) – nehomogeniniai.

Nagrinėjant rekurentinius sąryšius labai svarbi yra skaičių sekos sąvoka.

Apibrėžimas. *Skaičių seka vadinama skaitinė funkcija, apibrėžta natūraliųjų skaičių aibėje* ***N****.*

Jeigu kiekvienam natūraliajam skaičiui *n* tam tikru būdu priskiriamas realusis skaičius x_n, tai sakoma, kad nusakyta skaičių $x_1, x_2, \ldots x_n, \ldots$ seka; x_1 – pirmasis sekos narys, ..., x_n – *n*-asis sekos narys. Seką žymėsime simboliu $\{x_n\}$. Tada $x_n = f(n)$ – sekos bendrasis narys.

---

[1] Skirtinguose literatūros šaltiniuose rekurentiniai sąryšiai gali būti vadinami ir skirtuminėmis lygtimis.

Seka gali būti išreikšta formule, nurodančia, kaip apskaičiuoti tos sekos n – tąjį narį.

1 pavyzdys. Tegul $x_n = \frac{n^3}{n^3 +}$.

▶ Tada $x_1 = \frac{1}{1+} = \frac{1}{4};\ x_2 = \frac{8}{11};\ x_3 = \frac{9}{10};\ldots$ ◀

Arba seka kartais apibrėžiama rekurentiniu būdu:

a) nurodomas pirmasis sekos narys (arba keli pirmieji nariai),

b) nurodoma formulė, pagal kurią n – tasis sekos narys apskaičiuojamas iš prieš jį esančių narių.

Apibrėžimas. *Seka, kurios kiekvienas narys* x_n *yra prieš einančių kelių narių funkcija, vadinama rekurenčiąja seka.* Jeigu

$$x_{n+} = \varsigma(x_{n+-}\ x_{n+-}, \ldots, x_n)$$

su visomis n reikšmėmis, tai seka $\{x_n\}$ vadinama *k-osios eilės rekurenčiąja seka*. Pavyzdžiui, seka $\{x_n\}$, apibrėžta sąryšiu $x_{n+} = x_n + \frac{1}{x_n}$, $n \in N$, yra pirmosios eilės rekurenčioji seka, o seka $\{y_n\}$, apibrėžta sąryšiu $y_{n+} = y_{n+}^2 + \frac{2}{y_{n+}} - y_n, n = 0,1,2,3,\ldots$, yra trečiosios eilės rekurenčioji seka.

Toliau panagrinėsime pavyzdžius, kuriuose sudaromi rekurentiniai sąryšiai uždavinių sprendimui.

1 pavyzdys. Fibonačio skaičiai.

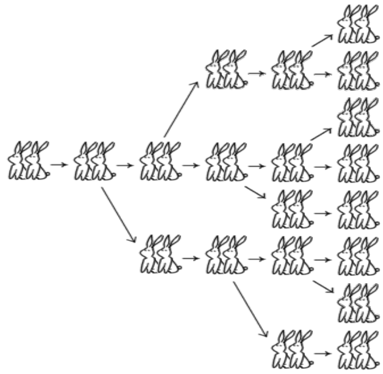
Iš triušių poros kas mėnesį gaunamas dviejų triušiukų prieauglis (patinėlio ir patelės), o iš atvestųjų triušiukų po dviejų mėnesių jau gaunamas naujas prieauglis. Kiek triušių porų turėsime po metų, jei metų pradžioje turėjome vieną porą?[2]

Sprendimas. Matome, kad po pirmojo mėnesio turėsime dvi triušių poras. Po antrojo mėnesio tris poras - prieauglį davė tik pirmoji pora. Dar po mėnesio prieauglį duos ir pradinė pora, ir pora, atvesta prieš du mėnesius. Viso – 5 poros, ir t.t. 1 pav.)

---

[2] Uždavinys pateiktas 1202 metais "Liber abaci" knygoje.

Laikas = 0 1 2 3 4 5



Poros = 1 1 2 3 5 8

1 pav.

Spręsdami šį uždavinį, sudarome rekurentinį sąryšį. Simboliu $F(n)$ pažymime triušių porų skaičių po n mėnesių, skaičiuojant nuo metų pradžios. n - ojo mėnesio pabaigoje turėsime $F(n-1)$ porą ir dar tiek pat naujų porų, kiek jų buvo $(n-2)$- ojo mėnesio pabaigoje, t.y. dar $F(n-2)$. Vadinasi,

$$F(n) = F(n-1) + F(n-2)\ .$$

Kadangi iš sąlygos matyti, kad $F(0) = 1$, o $F(1) = 2$, tai rekurentinis sąryšis su pradinėmis sąlygomis apibrėš triušių porų skaičių: $F_0 = 1$, $F_1 = 2$, $F_2 = 3$, $F_3 = 5$, $F_4 = 8$,..., $F_{11} = 233$,...

Gauti skaičiai vadinami Fibonačio skaičiais, o jų seka – Fibonačio skaičių seka.

Pastaba. Tą pačią seką gausime ir su šiomis pradinėmis sąlygomis - $F_0 = 0$, $F_1 = 1$. Todėl tolimesniuose nagrinėjimuose naudosime šias pradines sąlygas.

2 pavyzdys. Rekurentinių sąryšių taikymas kompiuterių moksle.

Programavimo kalboje skaičių išraiškos sudarytos iš dešimties simbolių: 0, 1, 2, ..., 9 ir keturių aritmetinių veiksmų: +, –, ×, ÷. Reikia apskaičiuoti skaičių n ilgio išraiškų su sąlyga, kai, šios kalbos sintaksė reikalauja, kad kiekviena išraiška baigtųsi skaitmeniu, ir dvi išraiškos gali būti jungiamos, naudojant aritmetinės operacijos simbolį.

Norėdami apskaičiuoti tokių n ilgio išraiškų skaičių, būtent ir pasinaudosime rekurentiniu sąryšiu.

Simboliu a_n pažymime nagrinėjamų n ilgio išraiškų skaičių. Visas tas išraiškas išskaidome į dvi klases. Pirmąją klasę sudarys išraiškos, kurios $(n-1)$– ojoje pozicijoje turi skaitmenį, antrąją klasę - kurių $(n-1)$ – ojoje pozicijoje yra aritmetinės operacijos simbolis.

Kadangi išraiška turi baigtis skaitmeniu, tai pirmojoje klasėje bus $10 \cdot a_{n-1}$ išraiškų, t.y. kiekviena iš $(n-1)$ ilgio išraiškų gali būti pratęsta vienu iš 10-ties būdų.

Kadangi dvi aritmetinės operacijos negali eiti kartu, tai antrojoje klasėje bus $40 \cdot a_{n-}$ išraiškų, t.y. $(n- \ )$ilgio išraiška gali būti pratęsta 4 aritmetinių operacijų simboliai $(n- \ )-$ oje pozicijoje ir 10 skaitinių simbolių $n-$tojoje pozicijoje. Vadinasi,

$$a_n = 10 \cdot a_{n-} \ + 40 \cdot a_{n-} \ .$$

Tam, kad galėtume pasinaudoti šia išraiška, turime a_0 ir a_1 reikšmes. Akivaizdžiai matyti, kad $a_0 = 0$, o $a_1 = 1$, nes ilgio 1 išraiška gali būti tik skaitmuo.

Turime

$$a_2 = \ 0a_1 + \cdot 0a_0 = \ 0 \cdot 10 + \ \cdot 0 \cdot 0 = \ 00,$$
$$a_3 = \ 0a_2 + \cdot 0a_1 = \ 0 \cdot 100 + \ \cdot 0 \cdot 10 = \ 400,$$
$$a_4 = \ 0a_3 + \cdot 0a_2 = \ 0 \cdot 1400 + \ \cdot 0 \cdot 100 = \ 8000.$$

Ir taip tęsiant skaičiavimus naudojant rekurentinį sąryšį, galime apskaičiuoti norimą skaičių n ilgio išraiškų, sudarytų iš dešimties simbolių: 0, 1, 2, ..., 9 ir keturių aritmetinių veiksmų: +, –, ×, ÷. $N >$

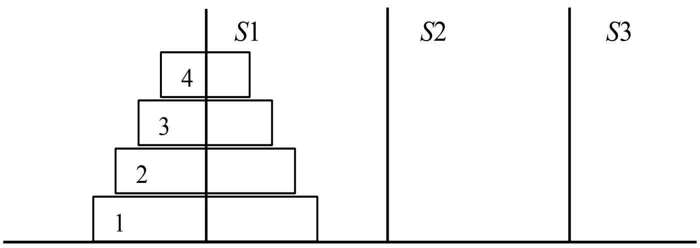
3 pavyzdys. Hanojaus bokštų uždavinys[3].

Duoti trys strypai: $S1, S2$ *ir* $S3$. *Ant pirmojo strypo užmauta n skirtingo skersmens* $d_1, d_2, ..., d_n$ *diskų, tenkinančių sąlygą:* $d_1 > d_2 > ... > d_n$ *(žr. 1.2 pav.). Šiuos diskus reikia perkelti nuo strypo* $S1$ *ant kito strypo, pvz.* $S3$, *laikantis taisyklių:*

1) *vienu kėlimu galima perkelti tik vieną diską, t.y. viršutinį diską;*
2) *didesnio skersmens diskas negali būti dedamas ant mažesnio skersmens disko;*
3) *visi diskai turi būti užmauti ant vieno iš strypų, t.y. disko padėjimas į šalį - negalimas.*

Kiek reiks atliktų perkėlimų, norint visus diskus nuo strypo $S1$ *perkelti ant kito strypo, pvz.* $S3$*?*

Simboliu $h(n)$ pažymėkime disko perkėlimų skaičių.



1.2 pav. Hanojaus bokštų uždavinys

Tarkime, kad mokame perkelti $(n- \ )$diską. Tuomet n diskų perkeliame šiais trimis žingsniais:

[3] Hanojaus bokštų uždavinį 1883 metais suformulavo prancūzų matematikas Eduardas Lukas (Edouard Lucas).

1. $(n-1)$ viršutinių diskų perkeliame nuo pirmojo strypo ant antrojo, naudodamiesi laisvu trečiuoju strypu;

2. apatinį n - ąjį diską uždedame ant trečiojo strypo;

3. $(n-1)$ diskų nuo antrojo strypo perkeliame ant trečiojo naudodamiesi laisvu pirmuoju strypu.

Dabar galime sudaryti rekurentinį sąryšį, nusakantį $h(n)$. Gauname, kad

$$h(n) = 2h(n-1) + 1,$$

t.y. norint perkelti n diskų, reikės du kartus perkelti $(n-1)$ diskų ir dar vieną n-ąjį diską.

- Šio rekurentinio sąryšio pradinė sąlyga yra $h(1) = 1$.
- Rekurentinis sąryšis $h(n) = 2h(n-1) + 1$ generuos seką: $h(1) = 1$;

$$h(2) = 2h(1) + 1 = 2 \cdot 1 + 1 = 3,$$

$$h(3) = 2h(2) + 1 = 2 \cdot 3 + 1 = 7,$$

$$h(4) = 2h(3) + 1 = 2 \cdot 7 + 1 = 15 \text{ ir t.t.}$$

Išvada: Iš pateiktų pavyzdžių galime padaryti išvadas, kad visi rekurentiniai sąryšiai generuoja skaičių a_n sekas, kurios priklauso tiek nuo rekurentinio sąryšio, tiek ir nuo pradinių sąlygų.

2. REKURENTINIŲ SĄRYŠIŲ SPRENDIMAS

Prieš pradedant nagrinėti rekurentinių sąryšių sprendimą, pirmiau apibrėšime rekurentinio sąryšio eilę, kuri lygi *k*, kai jis sieja $f(n+k)$ su $f(n)$, $f(n+1)$, ..., $f(n+k-1)$. Pavyzdžiui:

$$f(n+2)=4f(n)f(n+1)-7f^3(n)+1$$

yra antros eilės rekurentinis sąryšis, o

$$f(n+3)=5f(n)f(n+1)+2f^2(n+2)+1$$

yra trečios eilės rekurentinis sąryšis.

Toliau nagrinėkime rekurentinių sąryšių sprendimą. Turimą *k*-tosios eilės rekurentinį sąryšį

$$f(n)=a_1f(n-1)+a_2f(n-2)+\ldots+a_kf(n-k)$$

tenkina be galo daug skirtingų sekų. *k* pirmųjų sekos narių (pradines sąlygas) galima parinkti laisvai, jie nėra susieti ryšiais. Tačiau, kai *k* pirmųjų narių pasirinkta, visi kiti nariai apskaičiuojami vienareikšmiškai: narys $f(k+1)$ pagal rekurentinį sąryšį išreiškiamas nariais $f(1)$, $f(2)$, ..., $f(k)$, narys $f(k+2)$ – nariais $f(2)$, $f(3)$, ..., $f(k+1)$ ir t.t.

Remiantis rekurentiniu sąryšiu ir pirmaisiais sekos nariais, galima vieną po kito apskaičiuoti sekos narius ir anksčiau ar vėliau surasti bet kokį sekos narį. Tačiau tokiu atveju turime apskaičiuoti ir visus pirmesniuosius narius: juk jų nežinodami, negalime rasti tolesnių narių. Tačiau dažnai reikalingas tik vienas konkretus sekos narys, o kitų narių nereikia. Todėl labai svarbu žinoti rekurentinio sąryšio sprendinį: natūraliojo argumento funkciją, kuri generuoja tą pačią seką, kaip ir rekurentinis sąryšis.

Rekurentinio sąryšio sprendinį galima apibrėžti taip: natūraliojo argumento funkcija $f(n)$ yra rekurentinio sąryšio sprendinys, jei ją įrašę į sąryšį, gausime tapatybę.

Pavyzdys. Imkime rekurentinį sąryšį

$$f(n+2)=6f(n+1)-8f(n)$$

su pradinėmis reikšmėmis $f(0)=1$, $f(1)=2$.

Sprendinys yra seka

$$2,\ 4\ ,\ 8,\ ...,\ 2^n,\ ...$$

Iš tikrųjų bendrasis šios sekos narys yra $f(n)=2^n$. Todėl $f(n+2)=2^{n+2}$, $f(n+1)=2^{n+1}$. Kadangi lygybė $2^{n+2}=6\cdot2^{n+1}-8\cdot2^n$ yra teisinga su bet kuria natūraliąja *n* reikšme, tai 2^n yra to sąryšio sprendinys. Jis vadinamas atskiruoju rekurentinio sąryšio sprendiniu.

Apibrėžimas. *k*-osios eilės rekurentinio sąryšio sprendinys vadinamas ***bendruoju sprendiniu***, jei jis priklauso nuo *k* laisvųjų konstantų $C_1,...,C_k$, kurias pasirinkus galima gauti bet kurį to sąryšio sprendinį.

Pavyzdžiui, rekurentinio sąryšio

$$f(n+2) = 4f(n+1) - 3f(n)$$

bendrasis sprendinys yra

$$f(n) = C_1 1^n + C_2 3^n .$$

Nesudėtinga patikrinti, kad $f(n) = C_1 1^n + C_2 3^n$ funkcija $f(n+2) = 4f(n+1) - 3f(n)$ sąryšį paverčia tapatybe. Reikia parodyti, kad bet kokioms $f(0)$ ir $f(1)$ reikšmėms egzistuoja konstantos C_1 ir C_2.

Įrašę $f(n) = C_1 1^n + C_2 3^n$ į $f(n+2) = 4f(n+1) - 3f(n)$, kai $n = 0$ ir $n = 1$, gausime:

$$\begin{cases} C_1 + C_2 = f(0), \\ C_1 + 3C_2 = f(1). \end{cases}$$

Kadangi sistemos determinantas nelygus nuliui, tai ši sistema visada turi sprendinį su bet kuriomis $f(0)$ ir $f(1)$ reikšmėmis.

2.1. Homogeninių rekurentinių sąryšių sprendimas

Rekurentiniams sąryšiams spręsti bendrų taisyklių nėra. Dažnai naudojami sprendimo metodai:

1. charakteristinių šaknų;
2. generuojančių funkcijų;
3. neapibrėžtųjų koeficientų.

Nagrinėkime tiesinį homogeninį sąryšį su pastoviais koeficientais

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \ldots + a_k f(n-k),$$

čia $a_1, a_2, \ldots, a_k$ – kokie nors skaičiai.

Iš pradžių aptarsime tiesinių homogeninių rekurentinių su pastoviais koeficientais, kai $k = 2$ sprendimo metodą. Šis metodas taikomas ir aukštesnės eilės analogiškiems sąryšiams.

Rekurentinių sąryšių su pastoviais koeficientais sprendimas yra pagrįstas dviem tokiais teiginiais. Aptarkime.

Imkime 2-osios eilės su pastoviais koeficientais rekurentinį homogeninį sąryšį:

$$f(n) = a_1 f(n-1) + a_2 f(n-2),$$

a_1 ir a_2 – realieji skaičiai.

1 teiginys. Jei $f_1(n)$ ir $f_2(n)$ yra tiesiškai nepriklausomi $f(n) = a_1 f(n-1) + a_2 f(n-2)$ rekurentinio sąryšio sprendiniai, tai bet kokiems skaičiams – konstantoms C_1 ir C_2 funkcija

$$f(n) = C_1 f_1(n) + C_2 f_2(n)$$

yra $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sąryšio sprendinys.

Įrodymas. $f(n) = C_1 f_1(n) + C_2 f_2(n)$ įstatykime į $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sąryšį. Kairioji $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sąryšio pusė bus lygi

$$C_1 f_1(n) + C_2 f_2(n).$$

Apskaičiuokime kairiąją $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sąryšio pusę.

$$\begin{aligned} & a_1(C_1 f_1(n-1) + C_2 f_2(n-1)) + a_2(C_1 f_1(n-2) + C_2 f_2(n-2)) = \\ & = C_1(a_1 f_1(n-1) + a_2 f_1(n-2)) + C_2(a_1 f_2(n-1) + a_2 f_2(n-2)) = \\ & = C_1 f_1(n) + C_2 f_2(n), \end{aligned}$$

nes $f_1(n)$ ir $f_2(n)$ yra atskiri $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sąryšio sprendiniai.

2 teiginys. Jei skaičius r_1 yra kvadratinės lygties $r^2 = a_1 r + a_2$ sprendinys[4], tai seka

$$1, r_1, r_1^2, \ldots, r_1^{n-1}, \ldots$$

yra rekurentinio sąryšio $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sprendinys. T.y. $f(n) = r_1^n$

Įrodymas. Kadangi r_1 yra charakteristinės lygties sprendinys, tai

$$r_1^2 = a_1 r_1 + a_2.$$

Šią lygybę padauginę iš r_1^{n-2}, gausime:

$$r_1^n = a_1 r_1^{n-1} + a_2 r_1^{n-2},$$

t.y. $f(n) = r_1^n$ yra $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sprendinys.

Apibendrinimas. Rekurentinio sąryšio $f(n) = a_1 f(n-1) + a_2 f(n-2)$ bendrasis sprendinys apskaičiuojamas taip:

1) randame charakteristinės lygties $r_1^2 = a_1 r_1 + a_2$ sprendinius r_1 ir r_2,

2) užrašome bendrąjį sprendinį:

$$f(n) = C_1 r_1^n + C_2 r_2^n, \text{ jei } r_1 \neq r_2,$$

$$f(n) = C_1 r^n + C_2 n r^n, \text{ jei } r_1 = r_2 = r.$$

Šio apibendrinimo teisingumas pagrįstas aukščiau minėtu pirmu ir antru teiginiu, bei faktu, kad nr^n yra $f(n) = a_1 f(n-1) + a_2 f(n-2)$ sprendinys, jei $r_1 = r_2 = r$.

Tiesiniai rekurentiniai sąryšiai su pastoviais koeficientais, kurių eilė yra aukštesnė už du, sprendžiami analogiškai. Sakykime, turime tokį sąryšį:

$$f(n+k) = a_1 f(n+k-1) + \ldots + a_k f(n).$$

Tuomet jo charakteristinė lygtis yra

---

[4] Ši kvadratinė lygtis yra vadinama rekurentinio sąryšio $f(n) = a_1 f(n-1) + a_2 f(n-2)$ charakteristine lygtimi.

$$r^k = a_1 r^{k-1} + ... + a_k .$$

Jei k- ojo laipsnio lygtis turi skirtingus sprendinius $r_1,...r_k$, tai nagrinėjamo sąryšio bendrasis sprendinys yra

$$f(n) = C_1 r_1^{n-1} + C_2 r_2^{n-1} + ... + C_k r_k^{n-1}$$

Jei $r_1 = r_2 = ... = r_s$, tai tą sprendinį atitinka tokie $f(n+k) = a_1 f(n+k-1) + ... + a_k f(n)$ rekurentinio sąryšio sprendiniai:

$$f_1(n) = r_1^{n-1}, f_2(n) = n r_1^{n-1}, f_3(n) = n^2 r_1^{n-1}, ..., f_s(n) = n^{s-1} r_1^{n-1} .$$

Todėl tą sprendinį atitinka tam tikra bendrojo sprendinio dalis, būtent,

$$r_1^{n-1}(C_1 + C_2 n + C_3 n^2 + ... + C_s n^{s-1})$$

Sudarę tokius reiškinius visoms šaknims ir sudėję, gauname bendrąjį nagrinėjamo rekurentinio sąryšio sprendinį.

Toliau panagrinėkime homogeninių rekurentinių sąryšių sprendimo pavyzdžius.

1 pavyzdys. Išspręskime rekurentinį sąryšį, nusakantį Fibonačio seką:

$$F_n = F_{n-1} + F_{n-2},$$

su pradinėmis sąlygomis $F_0 = 0$, $F_1 = 1$.

Sudarome charakteristinę lygtį

$$r^2 = r + 1 .$$

Šios lygties sprendiniai yra

$$r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2} .$$

Vadinasi, bendrasis sprendinys, yra

$$F_n = C_1\left(\frac{1+\sqrt{5}}{2}\right)^n + C_2\left(\frac{1-\sqrt{5}}{2}\right)^n .$$

Konstantas C_1 ir C_2 parenkame taip, kad jos atitiktų pradines sąlygas: $F_0 = 0$, $F_1 = 1$. Turime:

$$C_1 + C_2 = 0,$$

$$C_1 \frac{1+\sqrt{5}}{2} + C_2 \frac{1-\sqrt{5}}{2} = 1 .$$

Išsprendę šią lygčių sistemą, gauname:

$$C_1 = -C_2 = \frac{1}{\sqrt{5}} .$$

Vadinasi, Fibonačio seką generuoja štai tokia natūraliojo argumento funkcija.

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right).$$

Toliau panagrinėkimekaip siejasi seka F_n su aukso pjūviu santykiu ϕ. Pirmiausiai apibrėžkime aukso pjūvio santykio sąvoką.

Apibrėžimas: Atkarpa aukso pjūvio santykiu daloma į dvi dalis *a* ir *b* taip, kad

$$\frac{a}{b} = \frac{a+b}{a}.$$

2.1 pav. Aukso pjūvio santykis

Kai $b = 1$, gauname

$$\frac{a}{1} = \frac{a+1}{a} = \phi,$$

$$\phi^2 = \phi + 1,$$

$$\phi = \frac{1 \pm \sqrt{5}}{2}.$$

Kadangi ϕ dalo atkarpą, tai $\phi = \dfrac{1+\sqrt{5}}{2}$.

Simboliu $\hat{\phi}$ žymėsime $1-\phi$, t.y.

$$\hat{\phi} = 1 - \phi = 1 - \frac{1}{2} - \frac{\sqrt{5}}{2} = \frac{1-\sqrt{5}}{2}.$$

Vadinasi, Fibonačio seką galima užrašyti taip:

$$F_n = \frac{1}{\sqrt{5}}\left(\phi^n - \hat{\phi}^n\right).$$

Kadangi $|\hat{\phi}| < 1$, tai $F_n \approx \dfrac{1}{\sqrt{5}}\phi^n$.

Pastaba: kai $n \geq 3$, $F_n = \dfrac{1}{\sqrt{5}}\phi^n$, turime apvalinti iki artimiausiojo sveikojo skaičiaus.

Pavyzdžiui: Fibonačio skaičių seka:

$$F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8 \text{ ir t.t.}$$

Paėmę $n = 3$, gausime: $\dfrac{1}{\sqrt{5}}\phi^3 \cong 1{,}89$. Suapvalinę gausime 2. Kai $n = 4$, gausime $\dfrac{1}{\sqrt{5}}\phi^4 \cong 3{,}07$.

Suapvalinę gauname 3. Paėmę $n = 5$, gausime $\dfrac{1}{\sqrt{5}}\phi^5 \cong 4{,}96$. Suapvalinę, turime 5, ir t.t.

2 pavyzdys. Išspręskime trečiosios eilės rekurentinį homogeninį sąryšį:

$$f(n) - 9f(n-1) + 26f(n-2) - 24f(n-3) = 0, \text{ kai } f(0) = 0,\ f(1) = 1,\ f(2) = 3.$$

Sąryšio charakteristinė lygtis yra

$$r^3 - 9r^2 + 26r - 24 = 0.$$

Šios lygties sprendiniai yra: $r_1 = 2$, $r_2 = 3$ ir $r_3 = 4$. Tada sąryšio bendrasis sprendinys yra

$$f(n) = C_1 2^n + C_2 3^n + C_3 4^n.$$

Apskaičiuojame konstantas:

$$\begin{cases} C_1 + C_2 + C_3 = 0, \\ 2C_1 + 3C_2 + 4C_3 = 1, \\ 4C_1 + 9C_2 + 16C_3 = 3. \end{cases}$$

Išsprendę lygčių sistemą gausime:

$$C_1 = -2, \; C_2 = 3 \text{ ir } C_3 = -1.$$

Vadinasi, nagrinėjamo rekurentinio sąryšio su nurodytomis pradinėmis sąlygomis sprendinys yra

$$f(n) = -2 \cdot 2^n + 3 \cdot 3^n - 4^n = -2^{n+1} + 3^{n+1} - 4^n.$$

2.2. Nehomogeninių rekurentinių sąryšių sprendimas

Šiame skyrelyje apžvelgsime tiesinių nehomogeninių rekurentinių sąryšių su pastoviais koeficientais sprendimą (NHRS), kuris yra analogiškas tiesinių nehomogeninių diferencialinių lygčių su pastoviais koeficientais sprendimui.

Nagrinėkime rekurentinį sąryšį

$$f(n+2) - a_1 f(n+1) - a_2 f(n) = p(n)$$

Šio sąryšio bendrasis sprendinys yra homogeninio sąryšio $f(n+2) - a_1 f(n+1) - a_2 f(n) = 0$ bendrojo sprendinio ir $f(n+2) - a_1 f(n+1) - a_2 f(n) = p(n)$ sąryšio atskirojo sprendinio suma.

Tiesinės ***nehomogeninės*** *lygties* ***bendrasis*** *sprendinys*	=	Tiesinės ***homogeninės*** *lygties* ***bendrasis*** *sprendinys*	+	Tiesinės ***nehomogeninės*** *lygties* ***atskirasis*** *sprendinys*

Apibendrinimas. Tarkime, kad nehomogeninis rekurentinis sąryšis turi pavidalą

$$f(n) + a_1 f(n-1) + \ldots + a_k f(n-k) = p(n).$$

Ir sąryšis $f(n) + a_1 f(n-1) + \ldots + a_k f(n-k) = 0$ yra nagrinėjamo nehomogeninio rekurentinio sąryšio homogeninis sąryšis. Tada nehomogeninio rekurentinio sąryšio bendrasis sprendinys ieškomas taip:

1) apskaičiuojamas HRS bendrasis sprendinys $f_H(n)$,

2) apskaičiuojamas NHRS atskirasis sprendinys $f_N(n)$,

3) šių sprendinių suma ir yra bendrasis NHRS sprendinys:

$$f(n) = f_H(n) + f_N(n).$$

Toliau panagrinėkime nehomogeninių rekurentinių sąryšių sprendimo pavyzdžius.

1 pavyzdys. Raskime bendrąjį tiesinio nehomogeninio rekurentinio sąryšio su pastoviais koeficientais $f(n) - 7f(n-1) + 10f(n-2) = 4^n$, $n \ge 2$, sprendinį.

Pirmiausia rasime šio homogeninio rekurentinio sąryšio $f(n) - 7f(n-1) + 10f(n-2) = 0$ sprendinį. Šio homogeninio rekurentinio sąryšio charakteristinė lygtis yra

$$r^2 - 7r + 10 = 0,$$

o sprendiniai $r_1 = 2$, $r_2 = 5$. Tada

$$f_H(n) = C_1 2^n + C_2 5^n.$$

Apskaičiuosime atskirąjį nehomogeninio rekurentinio sąryšio sprendinį. Sprendinio ieškosime laisvojo nario $\varphi(n)$ pavidale:

$$f_N(n) = C4^n.$$

Įstatę $f_N(n)$ į nehomogeninį rekurentinį sąryšį, gauname

$$C4^n - 7C4^{n-1} + 10C4^{n-2} = 4^n,$$

$$C \cdot 4^{n-2}(4^2 - 7 \cdot 4 + 10) = 4^n,$$

$$C(16 - 28 + 10) = 4^2,$$

$$-2C = 16,$$

$$C = -8.$$

Vadinasi, $f_N(n) = -8 \cdot 4^n = -2 \cdot 4^{n+1}$.

Tada NHRS bendrasis sprendinys yra

$$f(n) = C_1 2^n + C_2 5^n - 2 \cdot 4^{n+1}.$$

Toliau skaičiuosime C_1 ir C_2 su pradinėmis sąlygomis - $f(0) = 8$, $f(1) = 36$. Į $f(n) = C_1 2^n + C_2 5^n - 2 \cdot 4^{n+1}$ įstatę $n = 0$ ir $n = 1$, gauname

$$\begin{cases} 8 = C_1 + C_2 - 8, \\ 36 = 2C_1 + 5C_2 - 32. \end{cases}$$

Išsprendę lygčių sistemą, gausime: $C_1 = 4$, o $C_2 = 12$.

Vadinasi, $f(n) = 4 \cdot 2^n + 12 \cdot 5^n - 2 \cdot 4^{n+1}$.

2.3. Atskiro tiesinio nehomogeninio rekurentinio sąryšio su pastoviais koeficientais sprendinio parinkimas

Atskiro NHRS sprendinio ieškosime pavidale, kuris priklauso nuo nehomogeninio rekurentinio sąryšio laisvojo nario formos $\varphi(n)$ ir nuo HRS charakteristinės lygties sprendinių.

Todėl toliau nagrinėkime nehomogeninį rekurentinį sąryšį

$$f(n) + a_1 f(n-1) + a_2 f(n-2) + \ldots + a_k f(n-k) = \varphi(n),$$

su $f(0), f(1), \ldots, f(k-1)$ reikšmėms.

NHRS atskirojo sprendinio pavidalas, kai $\varphi(n) = Da^n$, čia D ir a – konstantos, randamas taip:

$$f_N(n) = \begin{cases} Ca^n, & \text{jei } CH(a) \neq 0, \\ Cn^m a^n, & \text{jei } a \text{ yra kartotinė } m \text{ kartų charakteri stinio} \\ & \text{polinomo } CH(n) \text{ šaknis,} \end{cases}$$

čia $CH(n)$ - HRS, atitinkančio nagrinėjamąjį NHRS, charakteristinis polinomas, t.y..

$r^k + a_1 r^{k-1} + \ldots + a_{k-1} r + a_k = 0$

1 pavyzdys. Išspręskime nehomogeninį rekurentinį sąryšį

$$f(n) - 5f(n-1) + 6f(n-2) = 9, \quad f(0) = 10, \quad f(1) = 25.$$

Charakteristinis polinomas yra

$$r^2 - 6r + 8 = 0$$

ir jo sprendiniai yra: $r_1 = 4$, $r_2 = 2$.

Tada $f_H(n) = C_1 4^n + C_2 2^n$.

Kadangi $a = 1$ ir a nėra charakteringojo polinomo sprendinys, tai $f_N(n)$ ieškomas pavidale $f_N(n) = C$.

Įstatę $f_N(n)$ į NHRS, gauname $C = 3$. Bendrasis NHR sprendinys

$$f(n) = C_1 4^5 + C_2 2^n + 3.$$

Atsižvelgiant į pradines sąlygas - $f(0) = 10$, $f(1) = 25$, gauname:

$$\begin{cases} C_1 + C_2 + 3 = 10, \\ 4C_1 + 2C_2 + 3 = 25, \end{cases}$$

Išsprendę lygčių sistemą, gauname sprendinius $C_1 = 4$, $C_2 = 3$.

Vadinasi, $f(n) = 4 \cdot 4^n + 3 \cdot 2^n + 3$.

Teorema. Tarkime, kad $f_{1N}(n)$ yra atskirasis NHRS

$$f(n) + a_1 f(n-1) + \ldots + a_k f(n-k) = P_1(n)$$

sprendinys, o $f_{2N}(n)$ yra atskirasis NHRS

$$f(n) + a_1 f(n-1) + \ldots + a_k f(n-k) = P_2(n)$$

sprendinys, $f_N(n) = f_{1N}(n) + f_{2N}(n)$ yra atskirasis NHRS

$$f(n) + a_1 f(n-1) + \ldots + a_k f(n-k) = P_1(n) + P_2(n)$$

sprendinys.

2 pavyzdys. Raskime nehomogeninio rekurentinio sąryšio

$$f(n) - 7f(n-1) + 10f(n-2) = 7 \cdot 3^n + 4^n$$

atskirąjį sprendinį.

Pirmiausiai ieškosime $f(n) - 7f(n-1) + 10f(n-2) = 7 \cdot 3^n$ atskirojo sprendinio $f_{1N}(n)$, tada – rekurentinio sąryšio $f(n) - 7f(n-1) + 10f(n-2) = 4^n$ atskirojo sprendinio $f_{2N}(n)$ ir galų gale juos sudėsime..

Nėra sudėtinga įsitikinti, kad

$$f_{1N}(n) = -\frac{63}{2} \cdot 3^n,$$

$$\text{o } f_{2N}(n) = -8 \cdot 4^n.$$

Gauname, kad nagrinėjamo NHRS atskirasis sprendinys yra

$$f_N(n) = -\frac{63}{2} \cdot 3^n - 8 \cdot 4^n.$$

Nehomogeninio rekurentinio sąryšio atskirasis sprendinys, kai $\varphi(n)$ yra polinomo ir eksponentės sandauga, yra pavidale:

$$\varphi(n) = (p_0 + p_1 n + \ldots + p_s n^s) \cdot a^n ,$$

čia $p_i, i = \overline{0,s}$ ir a – konstantos.

Šiuo atveju atskirojo sprendinio pavidalas

$$f_N(n) = \begin{cases} (q_0 + q_1 n + \ldots + q_s n^s)a^n, \text{ jei } CH(a) \neq 0, \\ n^m (q_0 + q_1 n + \ldots + q_s n^s)a^n, \text{ jei } a \text{ yra } m \text{ kartų kartotinė} \\ \qquad \text{charakteristinio polinomo } CH(n) \text{ šaknis,} \end{cases}$$

čia - $CH(n) = r^k + a_1 r^{k-1} + \ldots + a_{k-1} r + a_k = 0$.

3 pavyzdys. Išspęskime NHRS $f(n) - 5f(n-1) + 6f(n-2) = n^2 4^n$, $n \geq 2$.

Charakteristinio polinomo $r^2 - 5r + 6 = 0$ sprendiniai yra $r_1 = 2$, $r_2 = 3$.

Kadangi 4 nėra $CH(n)$ sprendinys, tai

$$f_N(n) = (q_2 n^2 + q_1 n + q_0) \cdot 4^n .$$

Įstatę šį sprendinį į NHRS, gausime:

$$(q_2 n^2 + q_1 n + q_0) \cdot 4^n - 5(q_2 (n-1)^2 + q_1 (n-1) + q_0) \cdot 4^{n-1} +$$
$$+ 6(q_2 (n-2)^2 + q_1 (n-2) + q_0) \cdot 4^{n-2} = n^2 4^n,$$

$$4^{n-2}((q_2 n^2 + q_1 n + q_0) \cdot 16 - 5(q_2 (n^2 - 2n + 1) + q_1 (n-1) + q_0) \cdot 4 +$$
$$+ 6(q_2 (n^2 - 4n + 4)^2 + q_1 (n-2) + q_0) = n^2 4^n.$$

Sudauginę ir sutraukę panašius narius, gauname:

$$2q_2 n^2 + (16q_2 + 2q_1)n + 4q_2 + 8q_1 + 2q_0 = 16n^2 .$$

Tuomet palyginę koeficientus prie tų pačių n laipsnių, gauname sistemą

$$\begin{cases} 2q_2 = 16, \\ 16q_2 + 2q_1 = 0, \\ 4q_2 + 8q_1 + 2q_0 = 0. \end{cases}$$

Išsprendę šią sistemą, randame sprendinius - $q_2 = 8$, $q_1 = -64$, $q_0 = 240$.

Vadinasi, gautas bendrasis sprendinys - $f_N(n) = (8n^2 - 64n + 240) \cdot 4^n$.

Išvada: Iš nagrinėtų pavyzdžių galime daryti išvadą, kad NHRS atskirojo sprendinio $f_N(n)$ išraiška priklauso nuo NHRS laisvojo nario – funkcijos $\varphi(n)$, tačiau $f_N(n)$ turi būti tiesiškai nepriklausomas nuo HRS sprendinio $f_H(n)$.

Nehomogeninių rekurentinių sąryšių atskirieji sprendiniai:

$\varphi(n)$	$CH(n)$ – charakteristinis polinomas	NHRS atskirojo sprendinio pavidalas
Da^n	$CH(a) \neq 0$	Ca^n
Da^n	a yra m kartų kartotinė $CH(n)$ šaknis	$Cn^m a^n$
$Dn^s a^n$	$CH(a) \neq 0$	$(q_0 + q_1 n + ... + q_s n^s)a^n$
$Dn^s a^n$	a yra m kartų kartotinė $CH(n)$ šaknis	$n^m(q_0 + q_1 n + ... + q_s n^s)a^n$
Dn^s	$CH(1) \neq 0$	$q_0 + q_1 n + ... + q_s n^s$
Dn^s	1 yra m kartų kartotinė $CH(n)$ šaknis	$n^m(q_0 + q_1 n + ... + q_s n^s)$

3. REKURENTINIAI SĄRYŠIAI IR GENERUOJANČIOS FUNKCIJOS

Rekurentinių sąryšių metodu galima išspęsti daugelį kombinatorikos uždavinių. Tačiau daugeliu atvejų rekurentinius sąryšius būna labai sunku sudaryti, o dar sunkiau juos išspręsti. Dažnai tą sunkumą pavyksta išvengti, naudojantis generuojančiomis funkcijomis. Generuojančių funkcijų teorija glaudžiai siejasi su rekurentiniais sąryšiais.

Prieš aptariant generuojančių funkcijų ryšį su rekurentiniais sąryšiais, apibrėšime generuojančios funkcijos sąvoką.

Tarkime, kad $\{a_n\} = \{a_0, a_1, \ldots\}$ yra skaičių seka. Sudarome laipsninę eilutę:

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots = \sum_{n=0}^{\infty} a_n x^n ,$$

kurią vadiname sekos $\{a_n\}$ *generuojančiąja funkcija*.

▶ Baigtinės skaičių sekos generuojančioji funkcija yra polinomas:

$$\{a_0, a_1, a_2, \ldots, a_n, 0, 0, \ldots\} \leftrightarrow a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_n x^n .$$

Begalinės skaičių sekos generuojančioji funkcija:

$$\{1, 1, \ldots, 1, 1, 1, \ldots\} \leftrightarrow 1 + 1 \cdot x + 1 \cdot x^2 \ldots = \sum_{n=0}^{\infty} x^n = \frac{1}{1-x}. \;◀$$

Rekurentiniai sąryšiai generuoja skaitines sekas, kurioms galima užrašyti generuojančias funkcijas. Nagrinėsime, kaip turint tiesinį rekurentinų sąryšį, sudaryti jam atitinkančią generuojančią funkciją.

Tarsime, kad $\frac{f(x)}{\varphi(x)}$ - taisyklinga algebrinė trupmena, čia $f(x)$ ir $\varphi(x)$ – atitinkamai n-osios ir m-osios ($n < m$) eilės polinomai:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n ,$$

$$\varphi(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_m x^m ,$$

ir $b_0 \neq 0$.

Tuo atveju, kai

$$\frac{f(x)}{\varphi(x)} = c_0 + c_1 x + c_2 x^2 + \ldots + c_k x^k + \ldots$$

turi būti teisinga tapatybė

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = (b_0 + b_1 x + b_2 x^2 + \ldots + b_m x^m) \cdot$$
$$\cdot (c_0 + c_1 x + c_2 x^2 + \ldots + c_k x^k + \ldots).$$

Padauginus tapatybės dešinėje pusėje esantį daugianarį iš laipsninės eilutės ir palyginę koeficientus prie vienodų x laipsnių kairėje ir dešinėje, gausime:

$$
\begin{aligned}
&b_0c_0 = a_0 \\
&b_0c_1 + b_1c_0 = a_1, \\
&b_0c_2 + b_1c_1 + b_2c_0 = a_2, \qquad\qquad (3.1) \\
&- \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \\
&b_0c_{m-} + b_1c_{m-} + \ .+ b_{m-}c_0 = a_{m-}\ .
\end{aligned}
$$

Visi tolesni sąryšiai bus vienodo pavidalo:

$$b_0c_{m+k} + b_1c_{m+k-} + \ .+ b_mc_k = 0, \quad k = 0,1,2,\dots$$

Vadinasi, $\frac{f(x)}{\varphi\ x)} = c_0 + c_1x + c_2x^2 + \ ..+ c_kx^k + \ ..$eilutės koeficientai $c_0, c_1, \dots, c_n, \dots$ tenkina tiesinį $b_0c_{m+k} + b_1c_{m+k-} + \ .+ b_mc_k = 0$ rekurentinį sąryšį. Sąryšio koeficientai priklauso nuo trupmenos vardiklio. Trupmenos skaitiklis naudojamas pirmiesiems rekurentinės sekos nariams $c_0, c_1, \dots, c_{m-}$ apskaičiuoti.

Panagrinėkime atvejį, kai duotas m-osios eilės $b_0c_{m+k} + b_1c_{m+k-} + \ .+ b_mc_k = 0$ rekurentinis sąryšis ir pradinės sąlygos $c_0, c_1, \dots, c_{m-}$. Tada skaičių seką $c_0, c_1, \dots, c_k, \dots$ generuojanti funkcija bus algebrinė trupmena:

$$\frac{f(x)}{\varphi\ x)} = \frac{a_0 + a_1x + \dots + a_{m-1}x^{m-1}}{b_0 + b_1x + \dots + b_mx^m}$$

čia $b_0, b_1, \dots, b_m$ – rekurentinio sąryšio koeficientai, o $a_0, a_1, \dots, a_{m-}$ – koeficientai, apskaičiuoti iš (3.1) lygčių.

Pastaba: Iš pirmo žvilgsnio atrodytų, kad mažai turėsime naudos, rekurentinį sąryšį pakeisdami generuojančia funkcija: nes vis tiek skaitiklį reikės dalyti iš vardiklio, o iš to gausime tą patį rekurentinį sąryšį. Tačiau $\frac{f(x)}{\varphi\ x)} = \frac{a_0 + a_1x + \dots + a_{m-1}x^{m-1}}{b_0 + b_1x + \dots + b_mx^m}$ trupmeną galima tapačiai pertvarkyti, o tai palengvina skaičių c_k radimą.

Toliau nagrinėsime pavyzdžius, kaip rasti generuojančiajai funkcijai atitinkančio rekurentinio sąryšio sprendinį bei išspręsime rekurentinį sąryšį generuojančių funkcijų metodu.

1 pavyzdys. Tarkime, generuojanti funkcija yra

$$\frac{x^3 - \ x^2 + \ x +}{x^4 - \ 0x^2 + \ }.$$

Išdėstykime šią trupmeną paprastesnėmis trupmenomis. Pirmiausiai randame vardiklio polinomo šaknis. Kadangi:

$$x^4 - \ 0x^2 + \ = \ x^2 - \ )(x^2 - \ ) = \ x - \ )(x + \ )(x - \ )(x + \ ).$$

Vadinasi,

$$\frac{x^3 - \ x^2 + \ x +}{x^4 - \ 0x^2 + \ } = \frac{A}{x -} + \frac{B}{x +} + \frac{C}{x -} + \frac{D}{x +}.$$

Abi lygybės puses padauginę iš bendro vardiklio, gauname:

$$x^3-3x^2+2x+1=A(x+1)(x-3)(x+3)+$$
$$+B(x-\ )(x-\ )(x+\ )+C(x-\ )(x+\ )(x+\ )+D(x-\ )(x+\ )(x-\ )\cdot$$

Šią lygybę turi tenkinti bet kokios kintamojo x reikšmės. Kai $x=\ $, visi dešinėje pusėje parašyti dėmenys, išskyrus pirmąjį, lygūs nuliui; kai $x=-\ $ visi dešinėje pusėje parašyti dėmenys, išskyrus antrąjį, lygūs nuliui ir t.t. Vietoj x paeiliui rašydami 1, -1, 3, -3 suskaičiuojame:

$$\mathrm{A}=-\frac{1}{16},\ \mathrm{B}=-\frac{5}{16},\ \mathrm{C}=\frac{7}{48},\ \mathrm{D}=\frac{5}{4}.$$

Trupmenos $\dfrac{A}{x-\ }$ eilutė užrašoma pagal Niutono formulę[5].

Tokiu būdu, gausime:

$$\frac{x^3-3x^2+2x+1}{x^4-10x^2+9}=\frac{1}{16}(1+x+x^2+\ldots+x^n+\ldots)-\frac{5}{16}(\ -x+x^2-\ldots+(-1)^n x^n+\ldots)-$$
$$-\frac{7}{144}\left(\ +\frac{x}{3}+\frac{x^2}{3^2}+\ .+\frac{x^n}{3^n}+\ .\right)+\frac{5}{12}\left(\ -\frac{x}{3}+\frac{x^2}{3^2}-\ .+\ -\ )^n\frac{x^n}{3^n}+\ .\right)$$

Apskaičiavę koeficientą prie x^n, gausime c_n:

$$c_n=\frac{1}{16}-\frac{5}{16}(-\ )^n-\frac{7}{144\cdot 3^n}+\frac{5\cdot(-\ )^n}{12\cdot 3^n}.$$

Tuo pačiu gavome generuojančiai funkcijai atitinkančio rekurentinio sąryšio sprendinį.

Algebrinės trupmenos išreiškimas laipsnine eilute yra ekvivalentus atitikmuo rekurentinio sąryšio su nurodytomis pradinėmis sąlygomis sprendimui. Trupmenas išreikšdami elementariųjų trupmenų suma ir tas elementariąsias trupmenas išdėstydami laipsninėmis eilutėmis, galime spręsti tiesinius rekurentinius sąryšius su pastoviais koeficientais.

Sprendžiant rekurentinį sąryšį, panaudojant generuojančias funkcijas, turime:

1) sudaryti rekurentiniam sąryšiui generuojančią funkciją $\dfrac{f(x)}{\varphi\ x)}$,
2) $\dfrac{f(x)}{\varphi\ x)}$ išreikšti elementariųjų trupmenų suma, o jas – laipsninėmis eilutėmis (pagal Niutono formulę),
3) gautos eilutės koeficientas c_n prie x^n ir bus ieškomasis sprendinys.

2 pavyzdys. Išspręskime rekurentinį sąryšį

$$f(n)-5f(n-\ )+6f(n-2)=0,\ f(0)=\ ,\ f(1)=-\ .$$

Šiuo atveju $b_0=\ $, $b_1=-\ $ ir $b_2=6$.

[5] Lygybė $(a+x)^n=C_n^0a^n+C_n^1a^{n-1}x+\ldots+C_n^ka^{n-k}x^k+\ldots+C_n^nx^n$ vadinama Niutono formule.

Pagal (3.1) formulę apskaičiuojame a_0 ir a_1:

$$a_0 = b_0 c_0 = \ ,$$
$$a_1 = b_0 c_1 + b_1 c_0 = - \ .$$

Vadinasi, trupmenos $\dfrac{f(x)}{\varphi(x)} = c_0 + c_1 x + c_2 x^2 + \ldots + c_k x^k + \ldots$ skaitiklis lygus $- \ x + \ $. Gauname rekurentinį sąryšį atitinkančią generuojančią funkciją

$$\frac{f(x)}{\varphi(x)} = \frac{- \ x + \ }{x^2 - 5x + 6}.$$

Ši trupmena elementariosiomis trupmenomis išreiškiama taip:

$$\frac{- \ x + \ }{x^2 - 5x + 6} = \frac{13}{x-2} - \frac{20}{x-3},$$

ir, išskaidę eilute, gausime:

$$-\frac{13}{2}\left(1 + \frac{x}{2} + \frac{x^2}{2^2} + \ldots + \frac{x^n}{2^n} + \ldots\right) + \frac{20}{3}\left(1 + \frac{x}{3} + \ldots + \frac{x^4}{3^n} + \ldots\right).$$

Todėl

$$f(n) = -\frac{13}{2}\frac{1}{2^n} + \frac{20}{3}\frac{1}{3^n} = -\frac{13}{2^{n+}} + \frac{20}{3^{n+}}.$$

Išvada: Rekurentinių sąryšių sprendimas charakteristinių šaknų metodu yra žymiai efektyvesnis nei generuojančių funkcijų metodas.

4. REKURENTINIŲ SĄRYŠIŲ TAIKYMAS

Rekurentinių sąryšių (rekursijos) taikymas matematikoje ir informatikoje yra labai platus. Šiame skyriuje apžvelgsime rekurentinių sąryšių taikymą įvairiuose matematinių uždavinių sprendimuose, kitame skyriuje – rekursiją taikysime programavime.

Pirmiausiai panagrinėsime rekurentinių sąryšių taikymą integralų skaičiavimuose.

4.1. Integralų skaičiavimas taikant rekurentinius sąryšius

Prieš pradedant nagrinėti rekurentinių sąryšių taikymą integralų skaičiavimuose, apibrėšime skaičių sekos ribą.

Apibrėžimas. Jei $\forall \varepsilon > 0\ \exists N(\varepsilon): n > N \Rightarrow |a_n - a| < \varepsilon$, tai skaičių a vadiname **sekos riba**. Jei tokio skaičiaus nėra – seka ribos neturi.

Kitaip tariant, jeigu egzistuoja toks sekos narys a_n, nuo kurio pradedant, skirtumas tarp visų tolimesnių narių ir kažkokio skaičiaus a yra mažesnis, nei kažkoks iš anksto nustatytas skaičius (jis gali būti kiek norima mažas), tai sakome, kad a yra šios sekos riba.

4.1.1 pavyzdys. Naudodamiesi rekurentiniu sąryšiu apskaičiuokime integralo $I_n = \int_0^1 \frac{x^n}{x+5}dx$ *reikšmę, kai* $n = 7$.

Skaičiuojame šį integralą naudodamiesi rekurentiniu sąryšiu

$$I_n + 5I_{n-1} = \frac{1}{n}, \qquad (4.1)$$

nes

$$I_n + 5I_{n-1} = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5}dx = \int_0^1 x^{n-1}dx = \frac{1}{n}.$$

Kiekviena gauta reikšmė suapvalinama 15 skaitmenų po kablelio tikslumu. Kai $n = 0$, gauname:

$$I_0 = \int_0^1 \frac{dx}{x+5} = \ln(x+5)\Big|_0^1 = \ln 6 - \ln 5 \cong 0,182321556793954.$$

Toliau taikydami rekurentinį sąryšį, gauname:

$I_1 = 1 - 5I_0 \cong 0,088392216030230$, $\quad I_2 = \frac{1}{2} - 5I_1 \cong 0,058038919848850$,

$I_3 = \frac{1}{3} - 5I_2 \cong 0,043138734089082$, $\quad I_4 = \frac{1}{4} - 5I_3 \cong 0,034306329554591$,

$I_5 = \frac{1}{5} - 5I_4 \cong 0,028468352227045$, $\quad I_6 = \frac{1}{6} - 5I_5 \cong 0,024324905531440$,

$I_7 = \frac{1}{7} - 5I_6 \cong 0,021232615199941$, $\quad I_8 = \frac{1}{8} - 5I_7 \cong 0,018836924000296$.

Atsakymas: $I_7 \cong 0,021232615199941$.

Tačiau toliau panagrinėkime atvejį, kuomet skaičiuojant tą patį integralą, skaičiavimus apvalinsime 6 skaitmenų po kablelio tikslumu. Gauname:

$$I_0 = \int_0^1 \frac{dx}{x+5} = \ln(x+5)\Big|_0^1 = \ln 6 - \ln 5 \cong 0.182322.$$

$I_1 = 1 - 5I_0 = 1 - 0.911610 = 0.088390$, $I_2 = \frac{1}{2} - 5I_1 = 0.500000 - 0.441950 = 0.058050$,

$I_3 = \frac{1}{3} - 5I_2 = 0.333333 - 0.290250 = 0.043083$, $I_4 = \frac{1}{4} - 5I_3 = 0.250000 - 0.215415 = 0.034585$,

$I_5 = \frac{1}{5} - 5I_4 = 0.200000 - 0.172925 = 0.027075$, $I_6 = \frac{1}{6} - 5I_5 = 0.166667 - 0.135375 = 0.031292$,

$$I_7 = \frac{1}{7} - 5I_6 = 0.142857 - 0.156460 = -0.013603.$$

Pastaba: lengva pastebėti, kad $I_6 > I_5$ ir atrodytų visiškai nelogiška, kad $I_7 < 0$. Viso to priežastis – $I_0 = 0.18232156\ldots$ reikšmės apvalinimo ribojimas iki 6 skaitmenų po kablelio tikslumu. Vadinasi, naudojant mažesnio tikslumo reikšmę netikslumai auga ir todėl gauname: $I_6 > I_5$ ar $I_7 < 0$.

Naudojant didesnį tikslumą, pavyzdžiui, 15 skaitmenų po kablelio, klaidingi rezultatai pastebimi vėliau.

$$I_0 = \int_0^1 \frac{dx}{x+5} = \ln(x+5)\Big|_0^1 = \ln 6 - \ln 5 \cong 0{,}182321556793954,$$

$I_1 = 1 - 5I_0 = 0{,}088392216030230$, $I_2 = \frac{1}{2} - 5I_1 = 0{,}058038919848850$,

$I_3 = \frac{1}{3} - 5I_2 = 0{,}043138734089082$, $I_4 = \frac{1}{4} - 5I_3 = 0{,}034306329554591$,

$I_5 = \frac{1}{5} - 5I_4 = 0{,}028468352227045$, $I_6 = \frac{1}{6} - 5I_5 = 0{,}024324905531440$,

$I_7 = \frac{1}{7} - 5I_6 = 0{,}021232615199941$, $I_8 = \frac{1}{8} - 5I_7 = 0{,}018836924000296$,

$I_9 = \frac{1}{9} - 5I_8 = 0{,}016926491109632$, $I_{10} = \frac{1}{10} - 5I_9 = 0{,}015367544451838$,

$I_{11} = \frac{1}{11} - 5I_{10} = 0{,}014071368649901$, $I_{12} = \frac{1}{12} - 5I_{11} = 0{,}012976490083829$,

$I_{13} = \frac{1}{13} - 5I_{12} = 0{,}012040626503934$, $I_{14} = \frac{1}{14} - 5I_{13} = 0{,}011225438908903$,

$I_{15} = \frac{1}{15} - 5I_{14} = 0{,}010539472122154$, $I_{16} = \frac{1}{16} - 5I_{15} = 0{,}009802639389231$,

$I_{17} = \frac{1}{17} - 5I_{16} = 0{,}009810332465609$, $I_{18} = \frac{1}{18} - 5I_{17} = 0{,}006503893227513$,

$I_{19} = \frac{1}{19} - 5I_{18} = 0{,}020112112809804$, $I_{20} = \frac{1}{20} - 5I_{20} = 0{,}050560564049021.$

Gavome, kad $I_{17} > I_{16}$ ir $I_{20} < 0$.

Tad, taikydami rekurentinį sąryšį ir norėdami gauti kuo tikslesnę reikšmę, skaičiavimuose turime kiekvieną skaičių apvalinti kuo didesniu tikslumu.

Išveskime n-ojo nario formulę:

$$I_n = \frac{1}{n} - 5I_{n-1} = \frac{1}{n} - \frac{5}{n-1} + 5^2 I_{n-2} = \frac{1}{n} - \frac{5}{n-1} + \frac{5^2}{n-2} - 5^3 I_{n-3} =$$

$$= \frac{1}{n} - \frac{5}{n-1} + \frac{5^2}{n-2} - \frac{5^3}{n-3} + 5^4 I_{n-4} = \ldots = \frac{1}{n} - \frac{5}{n-1} + \frac{5^2}{n-2} - \frac{5^3}{n-3} + \ldots + \frac{(-1)^{n-1}5^{n-1}}{1} \cdot I_1 =$$

$$= \sum_{k=1}^{n-1} \frac{(-1)^{k-1}5^{k-1}}{n-k+1} + (-1)^n 5^n I_0 .$$

Naudodamiesi n-ojo nario formule apskaičiuokime integralo $I_n = \int_0^1 \frac{x^n}{x+5}dx$ *reikšmę, kai* $n = 7$.

Gauname:

$$I_7 = \frac{1}{7} - \frac{5}{7-1} + \frac{5^2}{7-2} - \frac{5^3}{7-3} + \frac{5^4}{7-4} - \frac{5^5}{7-5} + \frac{5^6}{7-6} \cdot (1 - 5 \cdot I_0).$$

Kai

$$I_0 = \int_0^1 \frac{dx}{x+5} = \ln(x+5)\Big|_0^1 = \ln 6 - \ln 5 \cong 0{,}182321556793954,$$

tai

$$I_7 = \frac{1}{7} - \frac{5}{6} + \frac{25}{5} - \frac{125}{4} + \frac{625}{3} - \frac{3125}{2} + 15625(1 - 5 \cdot 0{,}182321556793954) \cong 0{,}021232615199941,$$

t. y. gavome tą pačią reikšmę, kaip ir ankstesniuose skaičiavimuose (naudojant 15 skaitmenų po kablelio tikslumą).

Atsakymas: $I_7 \cong 0{,}021232615199941$.

Pastaba. Atkreipkime dėmesį, kad rekursija gali būti taikoma ir atvirkščiai, t.y. žinant kurį nors sekos narį, pavyzdžiui, I_{11} tam tikru tikslumu, galima gauti žemesnius narius, t. y. $I_{10}, I_9, I_8, \ldots I_0$.

Iš (4.1) rekurentinės formulės išreikškime I_{n-1}:

$$I_{n-1} = \frac{\frac{1}{n} - I_n}{5}. \qquad (4.2)$$

Laikydami, kad $I_{12} = I_{11}$, turėsime:

$$I_{11} + 5I_{11} \approx \frac{1}{12}, \; I_{11} \approx \frac{1}{72} \approx 0.013889.$$

Taikydami (4.2) rekurentinį sąryšį, gauname:

$$I_{10} = \frac{(\frac{1}{11} - 0.013889)}{5} = 0.015404, \qquad I_9 = \frac{(\frac{1}{10} - 0.015404)}{5} = 0.016919,$$

Toliau:

$$I_8 = 0.018838,\ I_7 = 0.021232,\ I_6 = 0.024325,\ I_5 = 0.028468,$$

$$I_4 = 0.034306,\ I_3 = 0.043139,\ I_2 = 0.058039,\ I_1 = 0.088392,$$

ir galų gale $I_0 = 0.182322$.

Vadinasi, skaičiuojant rekurentinės sekos narius, didelę įtaką turi gaunamų reikšmių tikslumas. Šiuo atveju, kai laikėme $I_{12} = I_{11}$, gavome gana netikslias pirmąsias I_{10}, I_9, I_8. reikšmes, tačiau tolesnės $I_7, I_6, I_5, ..., I_0$ reikšmės jau gautos 6 skaitmenų po kablelio tikslumu.

4.1.2 pavyzdys (Trigonometrinės funkcijos integralas).

Apskaičiuokime integralo $I_n = \int_0^{\frac{\pi}{2}} \sin^n x dx$ *reikšmę, naudodamiesi rekurentiniu sąryšiu, kai* $n = '$.

Integruodami dalimis gauname, kad

$$I_n = \int_0^{\frac{\pi}{2}} \sin^{n-1} x \sin x dx = -\sin^{n-1} x \cos x \Big|_0^{\frac{\pi}{2}} + (n-1)\int_0^{\frac{\pi}{2}} \sin^{n-2} x \cos^2 x dx$$

$$= (n-1)\int_0^{\frac{\pi}{2}} \sin^{n-2} x(1-\sin^2 x)dx = (n-1)\left[I_{n-2} - I_n\right].$$

Supaprastiname ir gauname rekurentinį sąryšį:

$$nI_n = (n-1)I_{n-2},$$

čia

$$I_n = \frac{n-1}{n} I_{n-2} = \frac{(n-1)(n-3)}{n(n-2)} I_{n-4} = ... = \frac{(n-1)(n-3)...}{n(n-2)...}[I_1 ar I_0].$$

Akivaizdu, kad $I_1 = 1$ ir $I_0 = \frac{\pi}{2}$.

Išvados:

1. Jei *n* lyginis, naudojama I_0 reikšmė; jei *n* nelyginis, naudojama I_1 reikšmė.

2. Kai *n* didiname, I_n sumažėja.

3.Naudodamiesi rekurentiniu sąryšiu sparčiai galime suskaičiuoti integralo $I_n = \int_0^{\frac{\pi}{2}} \sin^n xdx$ reikšmes.

Sprendimas. Naudodamiesi rekureniniu sąryšiu randame: (reikšmės skaičiavimo paspartinimui imsime intervalą kas 2)

$$I_3 = \frac{n-1}{n} I_1 = \frac{2}{3} I_1 = \frac{2}{3} = 0.66666,$$

$$I_5 = \frac{n-1}{n} I_3 = \frac{4}{5} I_3 = \frac{8}{15} = 0.53333,$$

$$I_7 = \frac{n-1}{n} I_5 = \frac{6}{7} I_5 = \frac{48}{105} = 0.45714.$$

Atsakymas: $I_7 \cong 0{,}45714$.

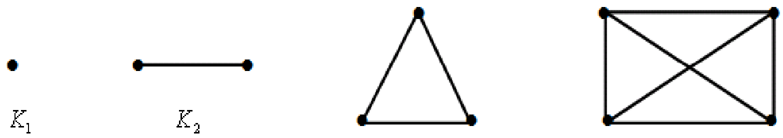
4.2. Rekurentinių sąryšių taikymas geometrijoje

Rekurentinių sąryšių taikymą taip pat galime pastebėti figūrų sudarymo ir brėžimo uždaviniuose. Tai galėtų būti tokio tipo uždaviniai kaip: keliais būdais galima padalyti iškiląjį $(n+1)$ - kampį į trikampius, nubrėžiant to daugiakampio viduje nesikertančias įstrižaines?

4.2.1 pavyzdys. Daugiakampio uždavinys.

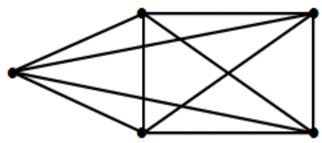
Turime daugiakampį K_n, gautą iš n viršūnių (taškų), iš kurių kiekvienos nubrėžtos atkarpos į kiekvieną kitą viršūnę. Reikia surasti daugiakampio K_n briaunų skaičiaus rekurentinį sąryšį.

Sprendimas. Daugiakampiai K_1, K_2, K_3 ir K_4 atitinkamai yra pavaizduoti žemiau pateiktuose paveikslėliuose:



K_1 K_2 K_3 K_4

Norėdami nubrėžti daugiakampį K_5, jis konstruojamas iš daugiakampio K_4 pridedant papildomai dar vieną viršūnę (tašką) ir iš šios viršūnės brėžiamos briaunos į visas kitas daugiakampio K_4 viršūnes:



K_5

Atkreipiame dėmesį, kad daugiakampio K_n:

briaunų skaičius $S_5 = 4 + S_4$ briaunų skaičius;

briaunų skaičius $S_4 = 3 + S_3$ briaunų skaičius;

briaunų skaičius $S_3 = 2 + S_2$ briaunų skaičius;

briaunų skaičius $S_2 = 1 + S_1$ briaunų skaičius.

Taigi

$S_n =$ daugiakampio K_n briaunų skaičius, tada

$$S_n = S_{(n-1)} + (n-1), \text{ kai } n \geq 2$$

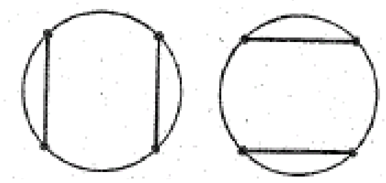
ir

$$S_1 = 0$$

yra daugiakampio K_n briaunų skaičiaus rekurentinis sąryšis.

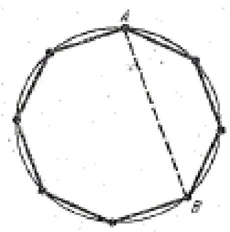
4.2.2 pavyzdys. Į apskritimą įbrėžtas taisyklingasis 2n – kampis. Kiek yra būdų sujungti jo viršūnes atkarpomis, jei nubrėžtosios atkarpos negali susikirsti viena su kita?

Sprendimas. Kai $n = 1$, yra tik vienas tokio sujungimo būdas[6]. Kai $n = 2$, yra du būdai, pavaizduoti 1 pav.



1 pav.

Norėdami rasti būdų skaičių $F(n)$, kai n – bet kuris natūrinis skaičius, sudarysime rekurentinį sąryšį, kurį tenkina $F(n)$. Tuo tikslu pasirinksime vieną daugiakampio viršūnę, pavyzdžiui, A. Ją galima sujungti su bet kokia kita viršūne B, tarp A ir B paliekant lyginį skaičių viršūnių. (2 pav.)



2 pav.

Visi viršūnių jungimo būdau suskirstomi į klases, atsižvelgiant į tai, kiek lieka viršūnių iš kairės nuo atkarpos, nubrėžtos iš taško A. Jei kairėje lieka $2s$ viršūnių, tai kitoje pusėje lieka $2(n-s-1)$ viršūnių. Šitaip $2n$ – kampis padalijamas į $2s$ – kampį ir $2(n-s-1)$ – kampį. Tačiau $2s$ – kampyje nubrėžti atkartas taip, kad jos viena su kita nesikirstų, yra $F(s)$ būdų. Tą patį padaryti $2(n-s-1)$ - kampyje yra $F(n-s-1)$ būdų. Pagal dauginimo taisyklę gauname, kad s – tajai klasei priklauso $F(s)F(n-s-1)$ atkarpų nubrėžimo variantų.

Vadinasi, visų galimų variantų skaičius lygus

$$F(0)F(n-1) + F(1)F(n-2) + \ldots + F(n-1)F(0).$$

Gavome rekurentinį sąryšį:

$$F(n) = F(0)F(n-1) + F(1)F(n-2) + \ldots + F(n-1)F(0).$$

---

[6] Skersmenį laikome „taisyklinguoju dvikampiu“.

4.3. Rekurentiniai sąryšiai kombinatorikos ir tikimybių teorijos uždaviniuose.

Sprendžiant kombinatorikos uždavinius yra taikomas metodas: tiriamąjį uždavinį pakeisti uždaviniu su mažesniu objektų skaičiumi. Taip gaunama formulė gretinių su pasikartojimais skaičiui rasti. Keitimo analogišku uždaviniu su mažesniu objektų skaičiumi metodas vadinamas rekurentinių sąryšių metodu.

Apžvelkime kombinatorikos uždavinių sprendimą taikant rekurentinių sąryšių metodą.

4.3.1 pavyzdys. Laimingieji troleibuso bilietai.

Kai kas šešiaženklį troleibuso bilieto numerį laiko „laimingu“, kai skaitmenų, parašytų lyginėse vietose, suma lygi sumai skaitmenų, parašytų nelyginėse vietose. Pavyzdžiui, bilietas 631 752 laikomas “laimingu”, nes 6 + 1 + 5 = 3 + 7 + 2 = 12. Reikia apskaičiuoti, kiek „laimingų“ numerių yra tarp 000 000 ir 999 999.

Sprendimas. Pirmiausia apskaičiuosime, kiek yra triženklių skaičių, kurių skaitmenų suma lygi tam tikram skaičiui N (triženkliais laikysime ir tokius skaičius, kaip 075, ir net 000). Šiuo atveju, dėmenų skaičius lygus 3, suma lygi N, o dėmenys – skaičiai nuo 0 iki 9. Jo sprendinių skaičių žymėsime $F(3,9;N)$. Tada galėsime parašyti tokį rekurentinį sąryšį:

$$F(3,9;N)=F(2,9;N)+F(2,9;N-\quad)+F(2,9;N-\quad)+F(2,9,N-\quad)+F(2,9;N-\quad)+$$
$$+F(2,9;N-\quad)+F(2,9;N-\quad)+F(2,9;N-\quad)+F(2,9;N-\quad)+F(2,9;N-\quad).$$

Panašiai

$$F(2,9;N)=F(1,9;N)+F(1,9;N-\quad)+\ .+F(1,9;N-\quad).$$

Akivaizdu, kad $F(1,9;N)=\quad$, kai $n\le N\le\quad$; kitais atvejais $F(1,9;N)=\quad$. Remdamiesi tais sąryšiais, užpildome šitokią lentelę:

k \ N	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
2	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5
3	1	3	6	10	15	21	28	36	45	55	63	69	73	75	75

k \ N	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	4	3	2	1	0	0	0	0	0	0	0	0	0
3	73	69	63	55	45	36	28	21	15	10	6	3	1

Norint sužinoti, kiek yra „laimingų“ bilietų, reikia skaičius, parašytus trečioje eilutėje, pakelti kvadratu ir gautuosius skaičius sudėti. Kiekviename „laimingame“ biliete skaitmenų, parašytų lyginėse vietose, suma lygi sumai skaitmenų, parašytų nelyginėse vietose. Sakykime, kad ta suma N. Trečioje eilutėje parašytasis skaičius rodo, kiek triženklių skaičių turi skaitmenų sumą, lygią *N.* Kitaip sakant, keliais būdais galima pasirinkti skaitmenis, rašomus lyginėse vietose (t. y. antroje, ketvirtoje ir šeštoje vietoje). Tiek pat būdų turime pasirinkti skaitmenis nelyginėms vietoms (pirmajai, trečiajai ir penktajai vietai) užpildyti. Kadangi vienas pasirinkimas nepriklauso nuo kito, tai “laimingų“ numerių, kurių skaitmenų, parašytų lyginėse vietose, suma lygi *N*. Visų „laimingų“ bilietų skaičių randame pagal sumavimo taisyklę:

$$2(1^2 + {}^2 + {}^2 + 0^2 + 5^2 + 1^2 + 8^2 + 6^2 + 5^2 + 5^2 + 3^2 + 9^2 + 3^2 + 5^2).$$

Atsakymas: 55 252.

4.3.2 pavyzdys. Lošimai.

Kitas pavyzdys susijęs su rekurenkiniais sąryšiais, gerai žinoma lošėjų pralaimėjimų problema.

Tarkime, lošėjas padaro lyginį (vienodą) skaičių lažybų (tikimybė laimėti $\frac{1}{2}$), kiekvieną kartą statydamas po vieną dolerį. Žaidimą lošėjas pradeda su a dolerių, o jo priešininkas su b dolerių. Kokia tikimybė, kad lošėjas pralaimės prieš savo priešininką?

Sprendimas. Tegul $f(n)$ pažymime tikimybę, kad lošėjas pralaimės prieš savo priešininką, kada lošėjas turi *n* dolerių ir priešininkas turi $a + \ - \ $ dolerių; tikimybė, kurią norime gauti yra $f(a)$, kur *a* yra pinigų kiekis su kuriuo lošėjas pradeda žaidimą.

Rekurentinį sąryšį $f(n)$ sudarome taip:

Kada lošėjas turi *n* dolerių, ir tikimybė pralaimėti prieš priešininką yra $f(n)$, žaidimo seka gali būti tęsiama vienu iš dviejų būdų: su tikimybe $\frac{1}{2}$ jis praranda kitą ėjimą, taip jam lieka $n -$ dolerių ir tada, su tikimybe $f(n - \ )$ jis pralaimi prieš savo priešininką. Kitas variantas, su tikimybe $\frac{1}{2}$ jis laimi kitą lošimą taip gaudamas $n +$ dolerių ir tada, su tikimybe $f(n + \ )$ jis pralaimi prieš savo priešininką.

Sudedant šiuos aptarpus faktus, matome, kad:

$$f(n) = \frac{1}{2} f(n - \ ) + \frac{1}{2} f(n + \ );$$

Tai ir yra mūsų ieškomas rekurentinis sąryšis.

Pastaba: Tačiau, norint pateikti išsamų šios problemos sprendimą, turime pridurti pradines sąlygas: $f(0) = \ $, kai lošėjas pralaimi pats sau, ir $f(a + b) = \ $, kai priešininkas bankrutuoja visais atvejais.

4.4. Sudėtinės palūkanos. Diskontavimas.

Apžvelgus rekurentinių sąryšių taikymą integralų skaičiavimuose, kombinatorikoje ir geometrijoje, galiausiai panagrinėsime dar vieną atvejį - rekurentinių sąryšių taikymą ekonomikoje. Skaičiuojant sudėtines palūkanas, vykdant diskontavimo procesą, neišvengiamai taikomi rekurentiniai sąryšiai.

Kadangi per praėjusiuosius 2010 metus Lietuvos gyventojams yra išduoda vien tik būsto paskolų virš 11% , panagrinėsime paskolų grąžinimo ir rekurentinių sąryšių ryšį.

4.4.1. Sudėtinės palūkanos.

Paskolos grąžinimas. Yra keli paskolos grąžinimo būdai: galima mokėti palūkanas kiekvieno laikotarpio pabaigoje, ir taip pat yra būdas, kuomet visas priklausančias palūkanas sumokėti. Panagrinėkime pavyzdžius paskolų grąžinimo, priklausomai nuo to, kokiu būdu bus mokamos palūkanos.

Pavyzdžiui, 1000 Lt. sumos mokėjimai 4 metų paskolai, esant 13% metinių palūkanų normai, kurias reikia mokėti kasmet (žr. 4. 4. 1 lentelė).

4.4.1 lentelė

Sudėtinių palūkanų skaičiavimas, kai palūkanos mokamos kasmet

Metai	Suma turima metų pradžioje	Palūkanos mokamos metų pabaigoje	Suma turima metų pabaigoje	Suma sumokama skolininko metų pabaigoje
1	1000	130	1130	130
2	1000	130	1130	130
3	1000	130	1130	130
4	1000	130	1130	1130

Jei palūkanos sumokamos kartu su pagrindine suma – palūkanos sudedamos. Palūkanos, priklausiančios ankstesniais metais, tampa dalimi pagrindinės šių metų sumos. Pavyzdžiui, 1000 Lt. paskola, esant 13% palūkanų normai, sudedamos kasmet. (žr. 4.4.2 lentelė) Jos 4 metų laikotarpiui duos efektą

4.4.2 lentelė

Sudėtinių palūkanų skaičiavimai

Metai	Suma metų pradžioje	Palūkanos pridedamos prie paskolos metų pabaigoje	Suma turima metų pabaigoje	Suma sumokama metų pabaigoje
1	1000	1000*0,13 = 130	1000*1,13 = 1130	00
2	1130	1130*0,13 = 146,90	$1000*1,13^2 = 1276,90$	00
3	1276,60	1276,90*0,13 = 165,60	$1000*1,13^3 = 1442,90$	00
4	1442,90	1442,90*0,13 = 187,58	$1000*1,13^4 = 1630,47$	1630,47

Abiem atvejais reikalaujama, kad palūkanos būtų mokamos nuo nemokėto likučio, tačiau šios dvi schemos duoda skirtingus efektus. Pirmuoju atveju mokėjimai palūkanų tuo laiku, kaip priklausė, išvengia palūkanų mokėjimo nuo palūkanų. Ir priešingai, kitoje mokėjimo schemoje – palūkanų skaičiavimas, naudojant procentų procentus. Taigi, sudėtinių palūkanų efektas priklauso nuo mokėjimų apimties ir nuo to, kada mokėjimai yra vykdomi.

Susiejant sudėtinių palūkanų skaičiavimą su nagrinėjama tema „rekurentinių sąryšių taikymai“, galima pastebėti, kad 4.4.2 lentelės skaičiavimai ir yra gaunami naudojant rekurentinį sąryšį

$$S_0 : U_{n+} = 1.16 U_n .$$

4.4.2. Diskontavimas.

Diskontavimas – finansų matematikoje naudojamas, jei reikia apskaičiuoti dabartinę pinigų vertę, kai žinoma būsimoji vertė po tam tikro laikotarpio. Šis būdas taikomas investuojant bei skolinantis pinigus.

Matematinė diskontavimo išraiška:

$$PV_0 = {}^{7}V_n * \frac{1}{(1+\ )^n} .$$

Šioje formulėje:

PV_0 – dabartinė vertė ; FV_n – būsimoji vertė n metais;

i – palūkanų norma, išreikšta dešimtaine trupmena; n – laiko periodas.

Nagrinėkime diskontavimo pavyzdį, kuriame taip pat galėsime įžvelgti rekurentinių sąryšių taikymą.

Pavyzdys. Banke norime padėti indėlį, kurio vertė po 7 metų bus 18 000 litų. Metinė palūkanų norma yra 6 % (priskaičiuojama kasmet). Kiek pinigų turime padėti šiandien, kad po 7 metų sutaupytume 18 000 litų?

Sprendimas. Šiuo atveju turime rekurentinį sąryšį $U_{n-} = \frac{U_n}{1.06}$, kurį naudojame skaičiuodami kiek pinigų turime padėti šiandien, kad po 7 metų sutaupytume 18 000 litų.

$$U_7 = 18\,000 \text{ Lt},$$

$$U_6 = \frac{U_n}{1.06} = \frac{180000}{1.06} = 16981.13 \text{ Lt},$$

$$U_5 = \frac{U_{n-}}{1.06} = \frac{16981.13}{1.06} = 16019.93 \text{ Lt},$$

Toliau gauname:

$U_4 = 15113.15$ Lt, $U_3 = 14\,257.68$ Lt, $U_2 = 13450.65$ Lt, ir galų gale U_1 = 12 689.29 Lt. T.y. suma, kurią turime padėti į banką, kad po 7 metų sutaupytume 18 000 litų, kuomet palūkanų norma yra 6 %.

5. PRAKTINĖ DALIS

Kaip jau minėta šio baigiamojo darbo 4 skyriuje, rekurentiniai sąryšiai taikomi ir programavime. Šio baigiamojo darbo praktinės dalies pagrindinis tikslas – uždaviniui

▶ *Daugiakampiui* K_n, *gautam iš n viršūnių (taškų) iš kurių kiekvienos nubrėžtos atkarpos į kiekvieną kitą viršūnę.* ◀

naudojantis „Java“ programavimo kalba:

1. užrašyti rekursiją, kuri suskaičiuotų daugiakampio K_n briaunų skaičių; (Priedas Nr. 1);

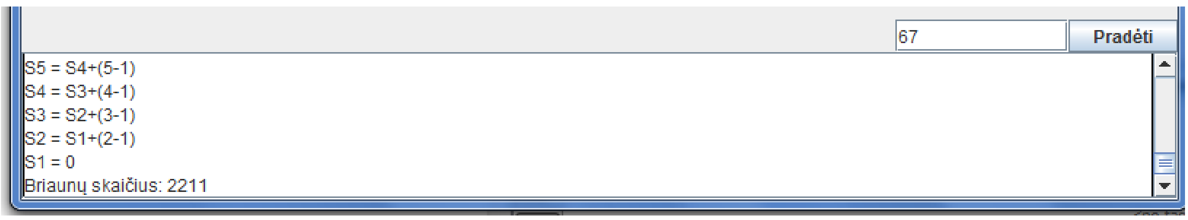
2. grafiškai pavaizduoti uždavinio sprendimą, t.y. nubrėžti figūrą, jos briaunas, pagal duotą sąlygą nuolat prijungiant vis po naują viršūnę. (Priedas Nr. 2).

Apibrėžimas. Programavime rekursija – programų ir algoritmų sudarymo metodas, kai programa kreipiasi pati į save, esant mažesnėms argumentų reikšmėms.

Grįždami prie uždavinio sprendimo, pirmiausiai užrašome rekursiją briaunų skaičiui gauti:

```
public int countEdges(int pointsNumber) {
     if(pointsNumber == 1 || pointsNumber == 0) {
          outputTextArea.append("S" + pointsNumber + " = 0\n");
          return 0;
        }
        else {
          outputTextArea.append("S" + pointsNumber + " = S" + (pointsNumber-1) + "+(" + pointsNumber + "-1)\n");
          return countEdges(pointsNumber-1) + (pointsNumber-1);
```

Norit sužinoti, kiek gaunama briaunų sujungus daugiakampio viršūnes su kiekviena esančia viršūne (pvz. viršūnių skaičius = 67), tai sukurtos programos pagalba įvedus 67 ir nuspaudus mygtuką <Pradėti>, pateikiamas reikšmės skaičiavimo algoritmas bei suskaičiuojamas briaunų skaičius, kuris šiuo atveju lygus 2211.



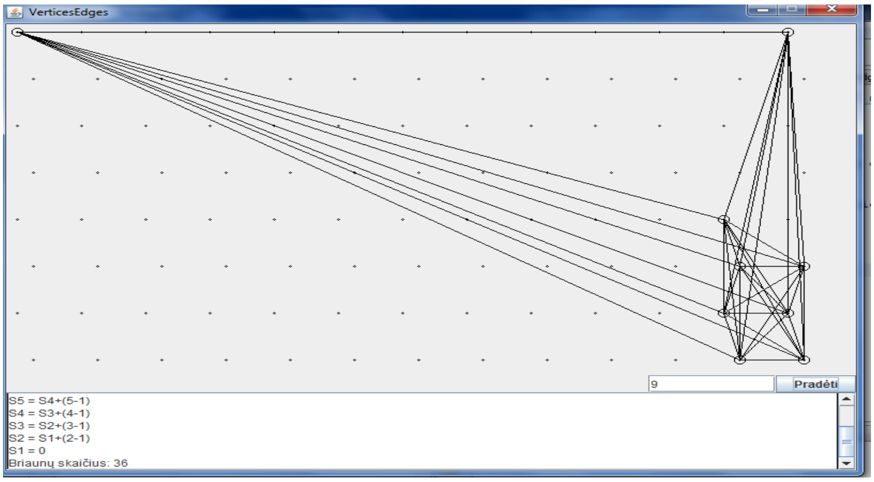
Taip įvedant bet kurią norimą reikšmę – viršūnių skaičių, gauname daugiakampio, gauto iš *n* viršūnių, iš kurių kiekvienos nubrėžtos atkarpos į kiekvieną viršūnę, briaunų skaičių.

Pastaba. Detalus uždavinio skaičiavimo algoritmo (rekursinė funkcija) kodas pateiktas 1. *Priede.*

Antra praktinės dalies užduotis - grafiškai atvaizduoti uždavinio sprendimą. Tam tikslui sukurta programa „Java“ programavimo kalba.

Tarkime, norime grafiškai pavaizduoti uždavinio sprendimą, kuomet viršūnių skaičius = 9 (1 pav.). Iš viso brėžiamos 36 briaunos.

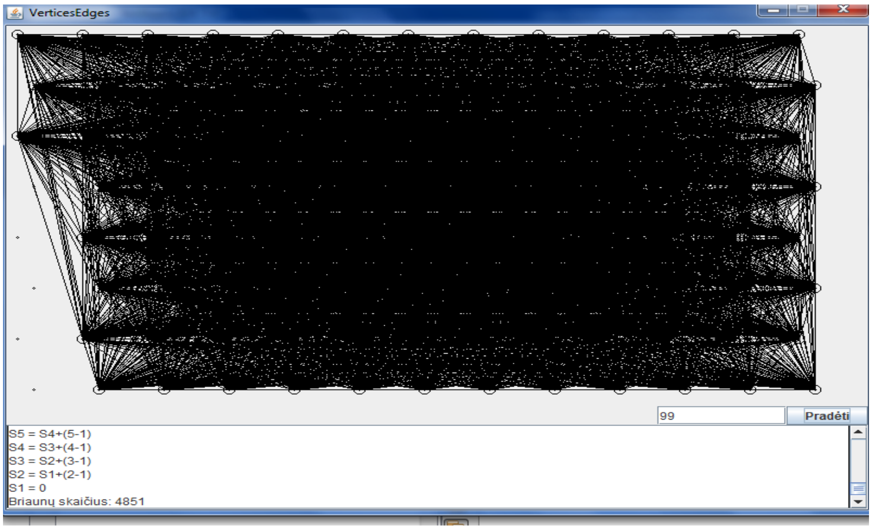
Turime:



1 pav.

Tarkime, norime grafiškai pavaizduoti uždavinio sprendimą, kuomet viršūnių skaičius = 99 (2 pav.). Iš viso brėžiamos 4851 briaunos. Esant tokiam dideliam briaunų skaičiui, stebimas briaunų vaizdo susiliejimas.

Turime:



2 pav.

Pastaba. Detalus grafinio vaizdavimo programos kodas yra pateiktas 2. *Priede*.

IŠVADOS

Magistro baigiamajame darbe apžvelgta ir išnagrinėta bendroji rekurentinių sąryšių teorija, tyrinėtas rekurentinių sąryšių sprendimas, jų ryšys su generuojančiomis funkcijomis bei analizuotos rekurentinių sąryšių taikymo matematikoje, programavime ir ekonomikoje galimybės.

Pagrindinis praktinės darbo dalies tikslas – pasinaudojant „Java“ programavimo kalba, užrašyti rekursiją bei grafiškai atvaizduoti kai kurių uždavinių, išnagrinėtų darbo teorinėje dalyje, sprendimus.

Apibendrinant baigiamąjį darbą galime daryti išvadas:

● Rekurentiniai sąryšiai (rekursija) plačiai naudojami tiek matematikoje, tiek informatikoje. Matematikoje dažnai vartojama, kai rekurentinio sąryšio pagalba labai paprastai ir akivaizdžiai apibrėžiamos priklausomybės tarp dydžių.

● Programuojant visada stengiamasi didesnį uždavinį skaidyti į mažesnius arba sudėtingesnį paversti keliais paprastesniais. Tam tikslui būtent naudojama rekursija.

● Rekursija programavime gali būti naudinga, kai reikia programuoti veiksmus su nežinomu operacijų skaičiumi, tai įgalina supaprastinti algoritmo formulavimą.

Bendrai tariant, rekurentinių sąryšių taikymas palengvina ir supaprastina kombinatorikos, diskrečiosios matematikos, tikimybių teorijos, ekonomikos ir kitokių uždavinių sprendimą.

REZIUMĖ

Summary

Recurrence relations and its application

Student: Sigita Mažvylaitė

Supervisor: Prof. dr. Eugenijus Stankus

Application of recurrence relations in mathematics and computer science is very widely used. To solve many task of combinatorics using recurrence relations, the task of n objects we can change with the task of the object (n - 1*)* and of this – task with object (*n - 2*) and etc. In turn reducing the number of the objects, finally we get the task that is not difficult to solve. When programming, always tries to split the task into more or less difficult to make a few simpler as well. For this purpose it is used to recursively functions.

In generally, the application of recurrence relations to facilitate and simplify the combinatorics, discrete mathematics, probability theory, economic and other problems of solving.

LITERATŪROS SĄRAŠAS

1. Balakrishnan V. K. Introductory discrete mathematics, United States of America: Dover Publications, 1991, p. 105 – 107.
2. Bathul S. Mathematical Foundations of computer science, New Delhi: Raj Press, 2004, p. 194 – 228.
3. Bjoerck A., Dahlquist G. Numerical mathematics and scientific computation: 1999, p. 8 – 22.
4. Chuan – Chong Ch., Khee – Meng K. Principles and techniques in combinatorics, Sinagore, 2004, p. 225 – 261.
5. Epp. S.S. Discrete Mathematics with Applications, Canada: Nelson Education, 2004, p. 304 - 311.
6. Gupta P.N. Comprehensive Differential equations, 2005, p. 62 – 67.
7. Krylovas A. Diskrečioji matematika, Vilnius: Vilniaus Gedimino technikos universitetas, 2008, p. 191 – 206.
8. Kumar K. R. Discrete Mathematics: Firewall Media, Laxmi publications, 2005, p. 23 – 37.
9. Lietuvos Jaunųjų matematikų mokykla. Jaunajam matematikui, Vilnius: Danieliaus leidykla, 2001, p. 17 – 23.
10. Lietuvos Jaunųjų matematikų mokykla. Jaunajam matematikui, Vilnius: Danieliaus leidykla, 2005, p. 57 – 71.
11. Manstavičius E. Analizinė ir tikimybinė kombinatorika, Vilnius: TEV, 2007, p. 86 – 107.
12. Page E. S., Wilson L. B. An introduction to computational Combinatorics, London: Fakenham and Reading, 1979, p. 4 – 19.
13. Pekarskas V. Diferencialinis ir integralinis skaičiavimas, Kaunas: Technologija, 2003.
14. Ram B. Discrete mathematics, India: India Binding House, 2011, p. 114 – 149.
15. Roberts F. S., Tesman B. Applied combinatorics, United States of America: CRC, 2009, p. 340 – 382.
16. Sedgewick R. Algorithms in JAVA, United States of America, 2004, p. 197 – 234.
17. Shanker Rao G. Discrete mathematical structures, India: Nisha Enterprises, 2003, p. 109 – 116.
18. Twesky V. Lecture on Applications – Oriented mathematics, Canada: Holden – Day, 1969, p. 110 – 119.
19. Vilenkinas N. Kombinatorika, Kaunas: Šviesa, 1979, p. 114-161.

20. Wallis W. D., George J. C. Introduction to Combinatorics, United States of America: CRC, 2009, p. 89 – 99.
21. Williamson S. G. Combinatorics for computer science, United States of America: University of California at San Diego, 1985, p.165 – 169.
22. http://e-stud.vgtu.lt/files/dest/12342/inzinerija%20(p.%201-83) [žiūrėta 2011 01 20].
23. http://en.wikipedia.org/wiki/Recurrence_relation [žiūrėta 2011 01 20].
24. http://ims.mii.lt/~rimga/vu/algoritminiai/medziaga/3.../Rekursija [žiūrėta 2011 02 24].
25. http://www.mif.vu.lt/~bastys/java/paskaitos/ [žiūrėta 2011 02 24].

PRIEDAI

Priedas Nr.1

Rekursija

```
package verticesedges;
import javax.swing.JTextArea;              /**Teksto išvedimui/**
public class VerticesEdgesLogic {
  JTextArea outputTextArea;
  public VerticesEdgesLogic(JTextArea outputTextArea) {
           this.outputTextArea = outputTextArea;

   /** Rekursinė f-ja briaunų skaičiui gauti.
   * @param pointsNumber  - taškų skaičius
   * @return - briaunų skaičius **/

  public int countEdges(int pointsNumber) {
    if(pointsNumber == 1 || pointsNumber == 0) {
             outputTextArea.append("S" + pointsNumber + " = 0\n");
             return 0;
           }
           else {
             outputTextArea.append("S"   +   pointsNumber   +   "   =   S"   +
(pointsNumber-1) + "+(" + pointsNumber  + "-1)\n");
             return countEdges(pointsNumber-1) + (pointsNumber-1);
```

Priedas Nr.2

Uždavinio grafinio vaizdavimo programa

```
package verticesedges;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import javax.swing.BorderFactory;
import javax.swing.JPanel;
public class VerticesEdgesDrawPanel extends JPanel {
       public static final long serialVersionUID = 6L;
           private ArrayList<Point> points;
           private static int step = 60;
           private static int secondXLine = 15;
           private Point minDimension;
           private Point maxDimension;

           public VerticesEdgesDrawPanel() {
             super();
             this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
             points = new ArrayList<Point>();}
/** Taškų, linijų paišymas.
* @param pointsNumber - taškų skaičius **/
           public      void      startDrawing(int      pointsNumber)      throws
NullPointerException {
      ArrayList<Point> pointsForDrawing = new ArrayList<Point>();
      Graphics2D g2 = (Graphics2D)this.getGraphics();
      Color oldColor = g2.getColor();
      g2.setColor(this.getBackground());
      g2.fillRect(0,0,this.getSize().width, this.getSize().height);
      g2.setColor(oldColor);
        for (Point p: points) {
                          g2.drawOval(p.x-1, p.y-1, 2, 2); }
     if (pointsNumber == 1) { // jei vienas taškas, tai jį nupiešiam
                g2.drawOval(minDimension.x-5, minDimension.y-5, 10, 10);
              } else if (pointsNumber != 0 && pointsNumber <= points.size()) {
//Priešingu atveju, ieškome tarp kokių taškų brėžti linijas
```

```
                        pointsForDrawing = findBestPoints(pointsNumber);
                        // Paišome linijas tarp atrinktų taškų.
                        for (Point p: pointsForDrawing) {
                                g2.drawOval(p.x-5, p.y-5, 10, 10);
                                for (Point p2: pointsForDrawing) {
                                  if(!p.equals(p2)) {
                                    g2.drawLine(p.x, p.y, p2.x, p2.y);}
                } else if (pointsNumber > points.size()) {
                                System.out.println("Per daug taškų");}
    /**Taškų suradimas paišymui.
     * @param pointsNumber - Taškų skaičius
     * @return Taškų masyvas, tarp kurių bus braižomos linijos. **/
    public ArrayList<Point> findBestPoints(int pointsNumber) {
           ArrayList<Point> pointsForDrawing = new ArrayList<Point>();
                      Point lastPoint, newPoint;
                      int selectedPoints = 0;
                      int same, maxSame, shift;
                      boolean xChanged, freePoint, pointSet;
                      Map<Integer, Integer> pointsNumberX, pointsNumberY;

                      pointsNumberX = new HashMap<Integer, Integer>();
                      pointsNumberY = new HashMap<Integer, Integer>();
                      // Pirmas taškas
        pointsForDrawing.add(points.get(0));
        lastPoint = points.get(0);
        pointsNumberX.put(new Integer(lastPoint.x), 1);
        pointsNumberY.put(new Integer(lastPoint.y), 1);
        selectedPoints++;
        maxSame = 2;
        same = 0;
        shift = 0;
        xChanged = false;
        freePoint = false;
        pointSet = false;
        newPoint = new Point();
        while    (selectedPoints    !=    pointsNumber    &&    points.size()    >
selectedPoints) {
          for (Point p: points) {
            if(!pointsForDrawing.contains(p)  &&  ((lastPoint.y + shift)  == p.y
|| (lastPoint.y - shift) == p.y) && !(xChanged)) {
      if ((pointsNumberX.get(p.x) == null || pointsNumberX.get(p.x) < maxSame)
   && (pointsNumberY.get(p.y) == null || pointsNumberY.get(p.y) < maxSame)) {
```

```
                                newPoint = p;
                              pointSet = true;}
           else  if(!pointsForDrawing.contains(p)  &&  (lastPoint.x  +  shift  ==
p.x || lastPoint.x + secondXLine + shift == p.x
           lastPoint.x - secondXLine + shift == p.x) && xChanged) {
            if  ((pointsNumberX.get(p.x)  ==  null  ||  pointsNumberX.get(p.x)  <
maxSame)
            &&  (pointsNumberY.get(p.y)  ==  null  ||  pointsNumberY.get(p.y)  <
maxSame)) {
                                    newPoint = p;
                                    pointSet = true;}
                      if(freePoint) {
                         if(!pointsForDrawing.contains(p)) {
                                newPoint = p;
                                freePoint = false;
                                pointSet = true;}
                      if(pointSet) {
                          selectedPoints++;
                          pointsForDrawing.add(newPoint);
                          pointsNumberX.put(newPoint.x,
(pointsNumberX.get(newPoint.x)==null?new Integer(1):new
Integer(pointsNumberX.get(newPoint.x).intValue()+1)));
                          pointsNumberY.put(newPoint.y,
(pointsNumberY.get(newPoint.y)==null?new Integer(1):new
Integer(pointsNumberY.get(newPoint.y).intValue()+1)));
                          lastPoint = newPoint;
                          pointSet = false;
                          xChanged = !(xChanged); }
                        else {
                                if(!xChanged) {
                                  xChanged = true;
                                } else {
            shift += step;
if ((shift > maxDimension.x) && (shift > maxDimension.y)) {
            maxSame++;
            shift = 0;
if (maxSame > maxDimension.x/step && maxSame > maxDimension.y/step) {
            maxSame = 2;
           freePoint = true;}
                      return pointsForDrawing;}
    /** Sukuriamas pradinis taškų masyvas. **/
    public boolean createPoints() {
```

```
        int panelHeight, panelWidth;
        int i, j;
        boolean lineSwitch;
        Dimension panelSize;
        int maxX;
        panelSize = this.getSize();
        panelHeight = panelSize.height;
        panelWidth = panelSize.width;
        i = 10;
        maxX = 10;
        j = 10;
        lineSwitch = false;
        minDimension = new Point (j, i);
if(panelHeight >= (step) && panelWidth >= (step)) {
          for (int z = i; z < panelHeight-10; z += step) {
            for (int w = j; w < panelWidth-10; w += step) {
                            if(maxX < w) {
                              maxX = w;}
      Point point = new Point(w, z);
      points.add(point);}
                   if (lineSwitch) {
                     j = 10;
                            lineSwitch = false;
                   } else {
                     j = 10 + secondXLine;
                     lineSwitch = true;}
          } maxDimension = new Point(maxX,
                            points.get(points.size()-1).y);
} else {
                 return false;
        return true; // viskas gerai}
      @Override
      public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics g2 = (Graphics2D)g;
        if (createPoints())
        {
          for (Point p: points) {
                    g2.drawOval(p.x-1, p.y-1, 2, 2);}
```