KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
TAIKOMOSIOS INFORMATIKOS KATEDRA

DISKREČIOSIOS STRUKTŪROS (P170B008)

KURSINIS DARBAS

Užduoties nr. A28 Ataskaita

Atliko:
IF-1/9 gr. studentas
Martynas Kuliešius
Priėmė:
Lekt. Audrius Nečiūnas

KAUNAS
2023

Turinys

Užduotis 4
Užduoties analizė 4
Sprendimo idėja 4
Programos paaiškinimas 5
Duomenų įvedimas: 5
Programos kodas 9
Testavimai 13
Išvados 19
Literatūros sąrašas 20

Nuotraukos

pav. 1 Pirmasis pavyzdys su sugeneruotais kėliniais 4
Pav. 2 Programos main metodas 5
Pav. 3 GautiKelinius metodas, kuris išgauna kėlinius 6
Pav. 4 Metodas skirtas išvesti kėliniams ir pakeisti jų elementų spalvas. 7
Pav. 5 RastiSekanti metodo kodas 7
Pav. 6 Elementų sukeitimo masyve metodas 7
Pav. 7 Raidėm nudažyti skirtas metodas 8
Pav. 8 faktorialo skaičiavimo metodas 8
Pav. 9 Pirmasis testas 13
Pav. 10 Antrasis testas 13
Pav. 11 Trečias testas 14
Pav. 12 Ketvirtas testas 15
Pav. 13 Penktasis testas 16
Pav. 14 Šešto testo pirma dalis 17
Pav. 15 Šešto testo antra dalis 17
Pav. 16 Septintas testas 1 dalis 18
Pav. 17 Septintas testas 2 dalis 18

Užduotis

a. A28 - Sugeneruoti kėlinių seką leksikografine tvarka.

Užduoties analizė

Sąvokos:

- Kėlinys – gretinys, sudarytas iš visų turimų elementų.
- Gretinys – elementų junginys, kuris nuo kito skiriase bent vienu elementu.
- Leksikografinė tvarka – rikiavimas nuo mažiausio iki didžiausio, arba rikiavimas abėcėlės tvarka a-z.
- Seka – kažkokia tvarka išrašyta skaičių arba kitų elementų aibė.
- Aibė – tam tikrų objektų rinkinys, sujungtas pagal kokį nors požymį.

Pasirinktai aibei elementų reikia sugeneruoti visus galimus kėlinius ir šiuos pavaizduoti leksikografine tvarka.

Sprendimo idėja

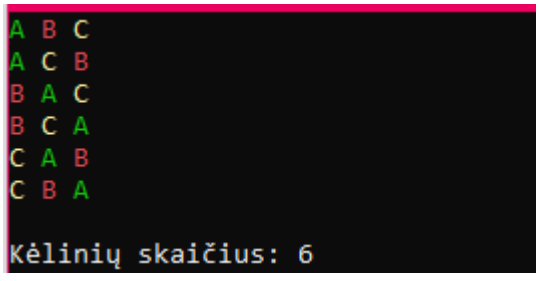
Pradžioje turime pradinę elementų aibę, kurioje yra n elementų. Žinoma, kad visų įmanomų kėlinių kiekis yra apskaičiuojamas pagal formulę $\mathbf{P_n = n!}$, kur n! – skaičiaus n faktorialas.

Pradžioje yra nustatoma elementų, iš kurių bus generuojami kėliniai, aibė, tada kviečiamas metodas kuris šiuos kėlinius generuos. Pradžioje, pirminė aibė surikiuojama tam, kad būtų kėlinių pradžia leksikografine tvarka (a-z) ir išvedama pirmoje eilutėje. Toliau naudojamas ciklas trunkantis n! kartų kad būtų išvedamas tinkamas kiekis kėlinių. Kiekvienos ciklo iteracijos metu surandamas sekantis tinkantis aibės elementas sukeitimui, jis sukeičiamas su i-tuoju elementu ir sukeitus aibės elementai perrikiuojami nuo sukeisto elemento iki aibės galo, tuomet naujas kėlinys yra išvedamas ir prasideda nauja ciklo iteracija, kur kartojasi žingsniai iki kol pasiekiamas n! skaičius.

Pirmąjam pavyzdžiui ir testui buvo naudojama aibė, kurios elementai yra {A B C}.

Atlikus n! skaičiavimą gauname, kad turime gauti 6 kėlinius iš viso, įskaitant ir pradinį darinį.

Kadangi aibėje yra 3 elementai, tai n! = 3*2*1 = 6



pav. 1 Pirmasis pavyzdys su sugeneruotais kėliniais

Programai atlikus skaičiavimus išvedamas rezultatas ir kiekvienas aibės elementas gauna kitokią spalvą, kad būtų lengviau matyti ar teisingai sugeneruotas sekantis kėlinys.

Taigi, iš aibės {A B C} buvo gauti tokie kėliniai (įskaitant ir pradinį darinį):

{ A B C }

{ A C B }

{ B A C }

{ B C A }

{ C A B }

{ C B A }

Programos paaiškinimas

Duomenų įvedimas:

Pradžioje, apsirašome pradinę elementų aibę „**aibe**“ simbolių masyvo tipu, kuri bus naudojama atlikti kėlinių generavimo metode „**GautiKelinius**“. Laiko matavimui naudojame integruotą C# programavimo kalbos priemonę „**Stopwatch**“ – laikmatį, kuris pradės skaičiuoti laiką prieš apsirašant aibę ir nustos skaičiuoti laiką kai „**GautiKelinius**“ metodas pabaigs savo darbą. Kai visas darbas yra pabaigiamas ir laikmatis sustabdomas, Į konsolę išvedama kiek turi būti kėlinių ir kiek laiko vyko skaičiavimai. (pav. 1)

```
0 references
public static void Main(string[] args)
{
    var watch = new System.Diagnostics.Stopwatch();
    watch.Start();

    char[] aibe = { 'A', 'B', 'C' };
    GautiKelinius(aibe);

    watch.Stop();

    Console.WriteLine("\nKėlinių skaičius: " + fac(aibe.Length));
    Console.WriteLine($"Programos veikimo laikas: {watch.ElapsedMilliseconds} ms");
}
```

Pav. 2 Programos main metodas

Iškvietus „***GautiKelinius***“ (pav. 3) metodą prasideda visas programos darbas. Pradžioje programos paprastumui išsisaugome dvi reikšmes: ***dydis*** ir ***skaiciuoklis***. Čia saugomas aibės dydis ir iteracijų skaičiavimui papildomas skaitliukas. Šie dydžiai išsaugomi siekiant šiek tiek supaprastinti programą. Išsaugojus reikiamą informaciją, surikiuojama elementų aibė a-z rikiavimu. Tai bus pirmasis kėlinių sąrašo elementas. Pagrindinio darbo ciklo skaičiavimo valdymui sukuriame boolean tipo vėliavą, kuri bus pakeičiama, kai pasieksime n! kėlinių. Darbo ciklo pradžioje iškviečiame metodą „***spausdinti***“ (pav. 4) , tam kad išvestume pirmąjį kėlinį (pavyzdžio atveju: A B C) ir iškarto padidiname skaitliuką. Susikuriame naują kintamąjį i, kuriame bus saugomas dešiniausio ir mažiausio simbolio indeksas, kurį išgauname panaudodami papildomą ***for*** ciklą tikrinant nuo priešpaskutinio elemento iki pirmojo ir tikrinant ar tas elementas yra mažesnis už dešiniau randamus elementus. Jei randamas toks elementas jo indeksas išsaugomas ***i*** kintamajame. Toliau turime ***if*** bloką, kuriame tikrinama ar yra toks elementas ir ar nėra pasiektas n! skaičius, gaunamas kviečiant

metodą „***fac***“ (pav. 8), saugomas ***skaiciuoklyje***. Jeigu nėra tokio elemento arba pasiektas faktorialo skaičius, boolean vėliava pakeičia reikšmę ir darbas baigiamas. Jeigu dar nepasiekta faktorialo reikšmė ir yra sekantis skaičius einame toliau ir ieškome sekančio elemento, kuris bus sukeičiamas su dabartiniu mažiausiu dešiniausiu elementu. Iškviestas metodas „***RastiSekanti***“ (pav. 5) ieško sekančio, didesnio už ***i*** indekse esantį elementą, kurio indeksą gražins, o šio indeksas bus naudojamas elementų sukeitimui masyve su metodu „***sukeisti***“ (pav. 6). Gavus sekančių elementų indeksus ir jų vietas sukeitus aibėje, ši dar kartą perrikiuojama nuo i-tojo elemento iki aibės galo. Taip išgaunamas naujasis kėlinys atitinkantis leksikografinę seką. Toliau kartojamas pagrindinis darbo ciklas, išvedamas naujasis kėlinys ir ieškomas naujas elementas, kurį reikia sukeisti, kad gauti naują kėlinį. Darbas kartojasi iki kol neberandamas dešiniausias mažiausias elementas arba pasiekiamas reikiamas kiekis kėlinių.

Išvedant kėlinius, išvedimo metode kiekvienam elementui yra pritaikoma jau iš anksto nustatyta spalva, tam, kad būtų lengviau žiūrėti ar teisingai sugeneruojami kėliniai. (pav. 7). Taip pat po visų išvestų kėlinių parodomas kėlinių kiekis ir juos sugeneruoti užtrukęs laikas, kaip matoma pav.1

```
1 reference
static void GautiKelinius(char[] aibe)
{
    int dydis = aibe.Length;
    int skaiciuoklis = 0;

    Array.Sort(aibe);

    bool isFinished = false;
    while (!isFinished)
    {
        spausdinti(aibe);
        skaiciuoklis++;

        int i;

        for (i = dydis - 2; i >= 0; --i)
        {
            if (aibe[i] < aibe[i + 1]) { break; }
        }

        if (i == -1)
        {
            if (skaiciuoklis == fac(aibe.Length))
            {
                isFinished = true;
            }
        }
        else
        {
            int sekantisID = RastiSekanti(aibe, aibe[i], i + 1, dydis - 1);

            sukeisti(aibe, i, sekantisID);

            Array.Sort(aibe, i + 1, dydis - i - 1);
        }
    }
}
```

Pav. 3 GautiKelinius metodas, kuris išgauna kėlinius

```
1 reference
static void spausdinti(char[] aibe)
{
    for (int i = 0; i < aibe.Length; i++)
    {
        string raide = aibe[i].ToString();
        keistiSpalva(raide);
        Console.Write(raide + " ");
    }
    keistiSpalva("reset");
    Console.WriteLine();

}
```

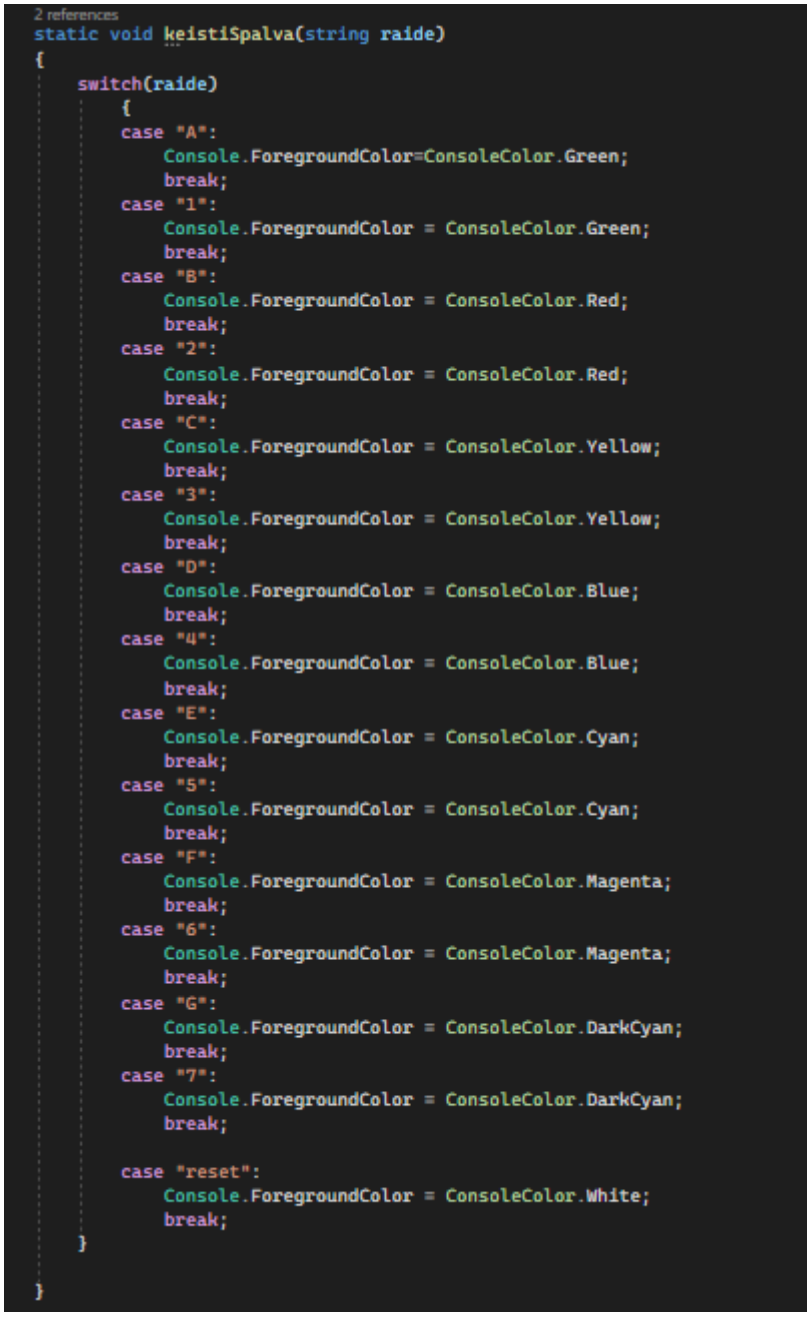
Pav. 4 Metodas skirtas išvesti kėliniams ir pakeisti jų elementų spalvas.

```
1 reference
static int RastiSekanti(char[] aibe, char pirmesnis, int l, int h)
{
    int ceilIndex = l;

    for (int i = l + 1; i <= h; i++)
        if (aibe[i] > pirmesnis && aibe[i] < aibe[ceilIndex])
            ceilIndex = i;

    return ceilIndex;
}
```

Pav. 5 RastiSekanti metodo kodas

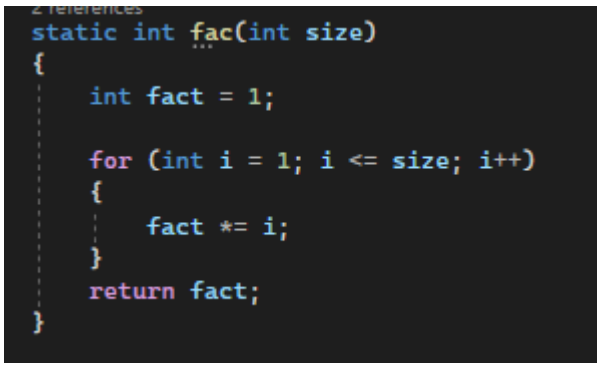
```
1 reference
static void sukeisti(char[] aibe, int i, int j)
{
    char t = aibe[i];
    aibe[i] = aibe[j];
    aibe[j] = t;
}
```

Pav. 6 Elementų sukeitimo masyve metodas



```
2 references
static void keistiSpalva(string raide)
{
    switch(raide)
        {
        case "A":
            Console.ForegroundColor=ConsoleColor.Green;
            break;
        case "1":
            Console.ForegroundColor = ConsoleColor.Green;
            break;
        case "B":
            Console.ForegroundColor = ConsoleColor.Red;
            break;
        case "2":
            Console.ForegroundColor = ConsoleColor.Red;
            break;
        case "C":
            Console.ForegroundColor = ConsoleColor.Yellow;
            break;
        case "3":
            Console.ForegroundColor = ConsoleColor.Yellow;
            break;
        case "D":
            Console.ForegroundColor = ConsoleColor.Blue;
            break;
        case "4":
            Console.ForegroundColor = ConsoleColor.Blue;
            break;
        case "E":
            Console.ForegroundColor = ConsoleColor.Cyan;
            break;
        case "5":
            Console.ForegroundColor = ConsoleColor.Cyan;
            break;
        case "F":
            Console.ForegroundColor = ConsoleColor.Magenta;
            break;
        case "6":
            Console.ForegroundColor = ConsoleColor.Magenta;
            break;
        case "G":
            Console.ForegroundColor = ConsoleColor.DarkCyan;
            break;
        case "7":
            Console.ForegroundColor = ConsoleColor.DarkCyan;
            break;

        case "reset":
            Console.ForegroundColor = ConsoleColor.White;
            break;
    }

}
```

Pav. 7 Raidėm nudažyti skirtas metodas



```
2 references
static int fac(int size)
{
    int fact = 1;

    for (int i = 1; i <= size; i++)
    {
        fact *= i;
    }
    return fact;
}
```

Pav. 8 faktorialo skaičiavimo metodas

Programos kodas

```
//Martynas Kuliešius IFF-1/9
using System;
using System.Collections.Generic;

class DiskretDarbs {

    public static void Main(string[] args)
    {
        var watch = new System.Diagnostics.Stopwatch();
        watch.Start();

        char[] aibe = { 'A', 'B', 'C' };
        GautiKelinius(aibe);

        watch.Stop();

        Console.WriteLine("\nKėlinių skaičius: " + fac(aibe.Length));
        Console.WriteLine($"Programos veikimo laikas:
{watch.ElapsedMilliseconds} ms");
    }

    /// <summary>
    /// Metodas, kuris atlieka pagrindinius keitimus, rikiavimus ir išvedimą.
    /// </summary>
    /// <param name="aibe">elementų masyvas/aibė</param>
    static void GautiKelinius(char[] aibe)
    {
        // Išsaugomas aibės dydis
        int dydis = aibe.Length;
        int skaiciuoklis = 0;

        // su bazine programinės kalbos funkcija surikiuojama pradinė
aibė/masyvas
        Array.Sort(aibe);

        // Kuria ir spausdina kėlinius
        bool isFinished = false; // pagrindiniam ciklui valdyti naudojamas
boolean kintamasis
        while (!isFinished) {
            // Išspausdina kėlinį
            spausdinti(aibe);
            skaiciuoklis++;

            // Suranda dešiniausią simbolį, kuris yra mažesnis už šale esančius
simbolius. jį laikau pirmesniu
            int i;

            for (i = dydis - 2; i >= 0; --i)
            {
                if (aibe[i] < aibe[i + 1]) // kai randamas mažesnis mažiausias
simbolis, jo indeksas išsaugomas ir pabaigiamas ciklas
                {
                    break;
                }
            }

            // Jeigu nėra dešiniausio simbolio, kuris yra mažesnis už paskiau
einantį simbolį, vadinasi kad visi galimi kėliniai buvo sukurti.
            if (i == -1)
            {
```

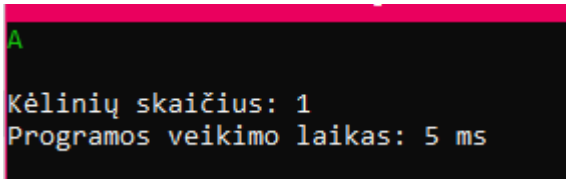
```
                if (skaiciuoklis == fac(aibe.Length)) // papildomas tikrinimas
tam atvejui, jeigu buvo praleistas koks nors kelinys
                {
                    isFinished = true;
                }
            }
            else
            {
                // Suranda pirmąjį simbolį eilėje sekantį simbolį.
                int sekantisID = RastiSekanti(aibe, aibe[i], i + 1, dydis - 1);

                //Sukeičia šiuos du simbolius vieną su kitu.
                sukeisti(aibe, i, sekantisID);

                Array.Sort(aibe, i + 1, dydis - i - 1); // Surikiuoja
masyvą/aibę nuo i-tojo elemento iki galo.
            }
        }
    }

    /// <summary>
    /// Metodas, kuris randa mažiausio, pirmąjį simbolį sekančio simbolio
indeksą masyve.
    /// </summary>
    /// <param name="aibe"> Masyvas kuriame saugomi visi simboliai</param>
    /// <param name="pirmesnis"> simbolis, pagal kurį ieškomi jį sekantys
simboliai</param>
    /// <param name="l"> pirmesniojo elemento indeksas</param>
    /// <param name="h"> aibes dydis. Iškviečiant metodą darašomas -1 todėl, kad
c# kalboje masyvų indeksacija prasideda nuo 0</param>
    /// <returns></returns>
    static int RastiSekanti(char[] aibe, char pirmesnis, int l, int h)
    {
        int ceilIndex = l;

        for (int i = l + 1; i <= h; i++)
            if (aibe[i] > pirmesnis && aibe[i] < aibe[ceilIndex])
                ceilIndex = i;

        return ceilIndex;
    }

    /// <summary>
    /// Papildomas metodas sukeisti simbolius vietomis
    /// </summary>
    /// <param name="aibe"></param>
    /// <param name="i"></param>
    /// <param name="j"></param>
    ///  A<->B

    static void sukeisti(char[] aibe, int i, int j)
    {
        char t = aibe[i];
        aibe[i] = aibe[j];
        aibe[j] = t;
    }

    /// <summary>
    /// Paskaičiuojamas kėlinių kiekis, kuris apsirašo n! (n faktorialu)
    /// </summary>
    /// <param name="size"> Aibės elementų kiekis </param>
```

```
    /// <returns></returns>
    static int fac(int size)
    {
        int fact = 1;

        for (int i = 1; i <= size; i++)
        {
            fact *= i;
        }
        return fact;
    }

    static void spausdinti(char[] aibe)
    {
        for (int i = 0; i < aibe.Length; i++)
        {
            string raide = aibe[i].ToString();
            keistiSpalva(raide);
            Console.Write(raide + " ");
        }
        keistiSpalva("reset");
        Console.WriteLine();

    }
    static void keistiSpalva(string raide)
    {
        switch(raide)
            {
            case "A":
                Console.ForegroundColor=ConsoleColor.Green;
                break;
            case "1":
                Console.ForegroundColor = ConsoleColor.Green;
                break;
            case "B":
                Console.ForegroundColor = ConsoleColor.Red;
                break;
            case "2":
                Console.ForegroundColor = ConsoleColor.Red;
                break;
            case "C":
                Console.ForegroundColor = ConsoleColor.Yellow;
                break;
            case "3":
                Console.ForegroundColor = ConsoleColor.Yellow;
                break;
            case "D":
                Console.ForegroundColor = ConsoleColor.Blue;
                break;
            case "4":
                Console.ForegroundColor = ConsoleColor.Blue;
                break;
            case "E":
                Console.ForegroundColor = ConsoleColor.Cyan;
                break;
            case "5":
                Console.ForegroundColor = ConsoleColor.Cyan;
                break;
            case "F":
                Console.ForegroundColor = ConsoleColor.Magenta;
                break;
```

```
            case "6":
                Console.ForegroundColor = ConsoleColor.Magenta;
                break;
            case "G":
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                break;
            case "7":
                Console.ForegroundColor = ConsoleColor.DarkCyan;
                break;

            case "reset":
                Console.ForegroundColor = ConsoleColor.White;
                break;
        }

    }
}
```

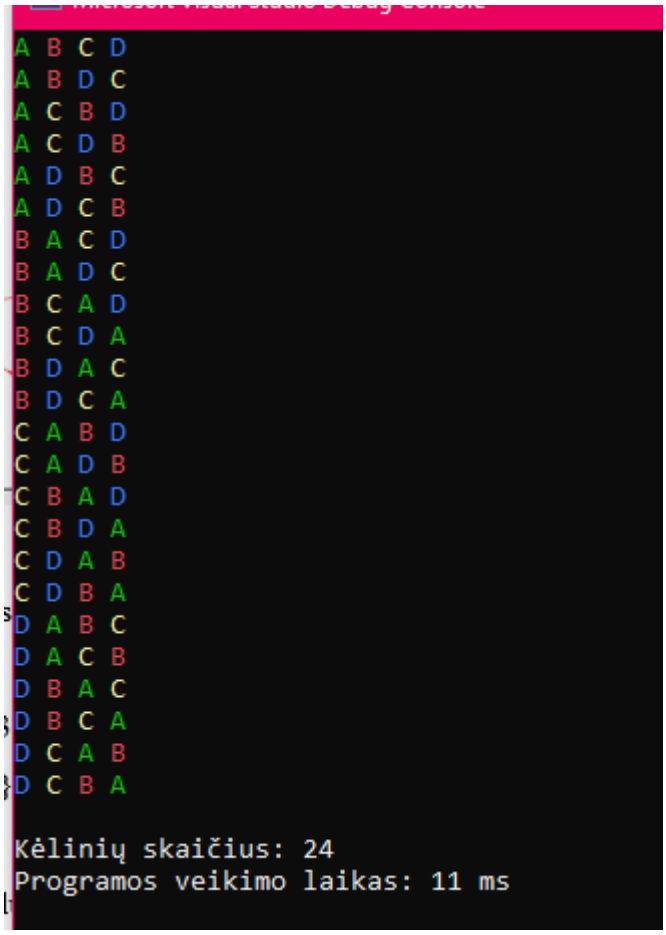
Testavimai

Pirmas testas
Testo tikslas: Išsiaiškinti kas bus jeigu paduodamas tik vienas elementas į aibę.
Duoti duomenys: aibe = { ‘A’}
Rezultatai:



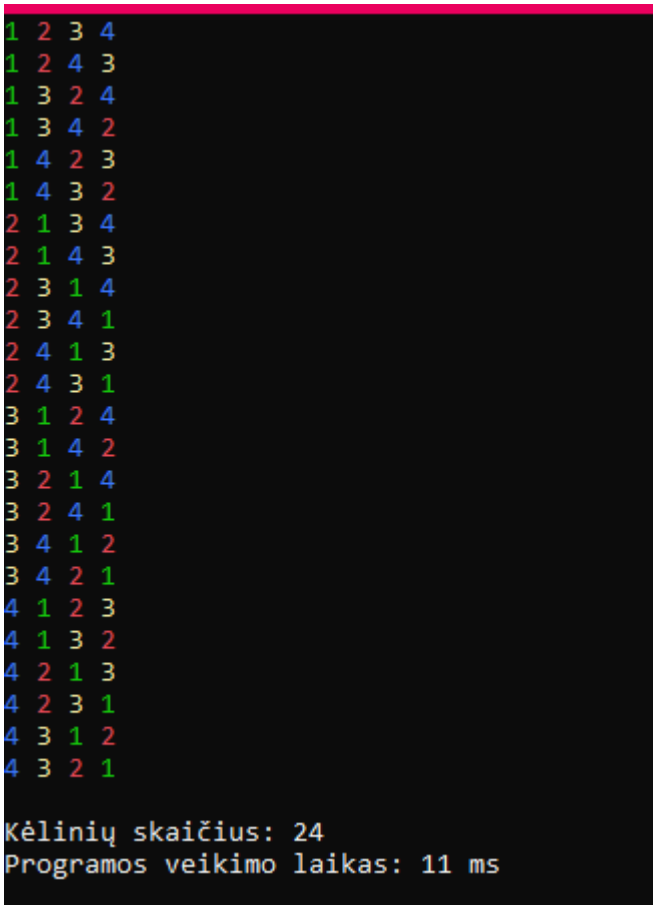
Pav. 9 Pirmasis testas

Antras testas
Testo tikslas: Patikrinti su daugiau nei dviem elementais
Duoti duomenys: aibe = { ‘A’, ‘B’, ‘C’, ‘D’}
Rezultatai:



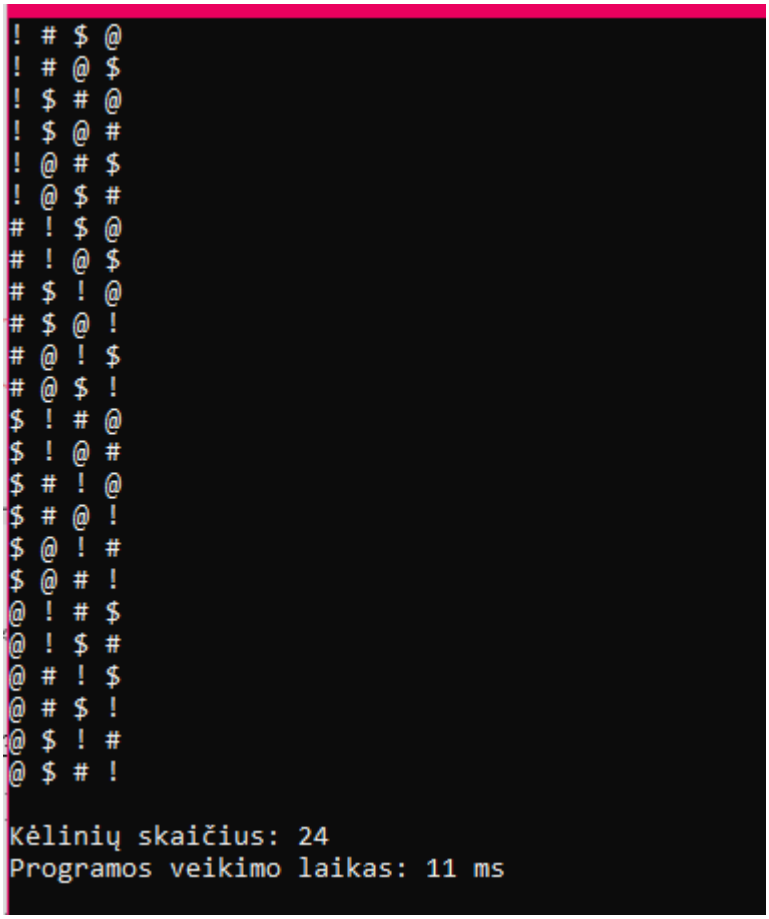
Pav. 10 Antrasis testas

Trečias testas
Testo tikslas: Patikrinti programos veikimą su skaitinėm elementų reikšmėm.
Duoti duomenys: aibe = { ‘1’, ‘2’, ‘3’, ‘4’}
Rezultatai:



Pav. 11 Trečias testas

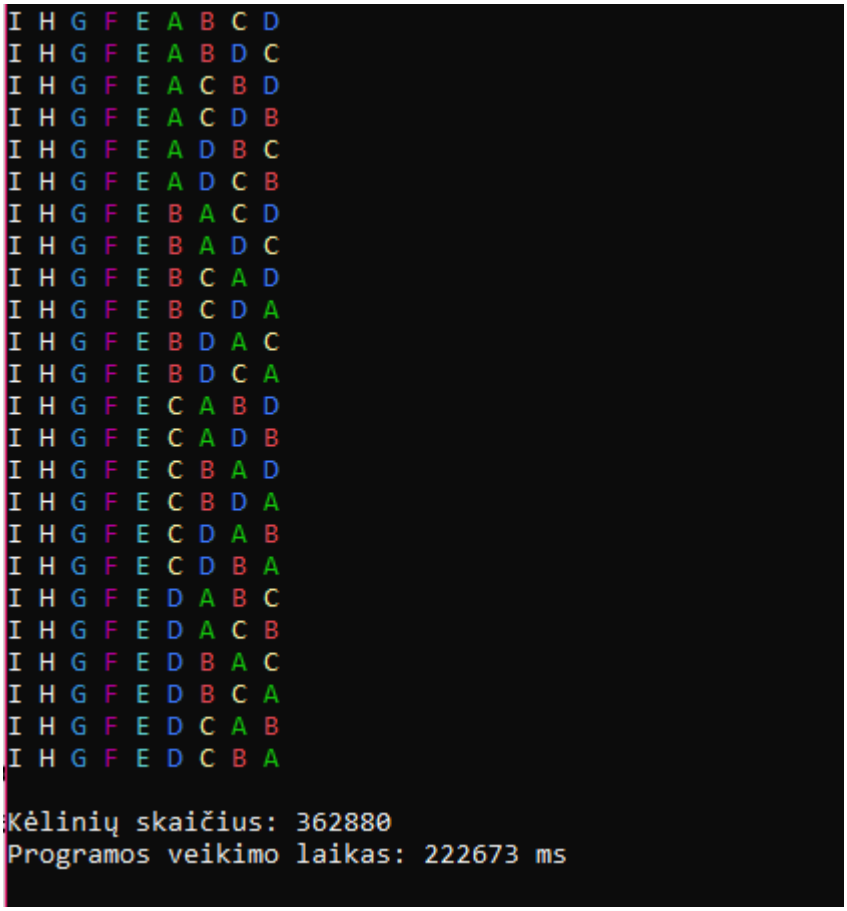
Ketvirtas testas
Testo tikslas: Patikrinti programos veikimą su neskaitinėm ir ne raidinėm elementų reikšmėm.
Duoti duomenys: aibe = { ‘!’, ‘@’, ‘#’, ‘$’}
Rezultatai:



Pav. 12 Ketvirtas testas

Kaip matoma, kadangi spalvos keitimui nėra numatytos ir nepaskirtos spalvos simboliams, spalva išlieka balta, tačiau generuoja kėlinių seką ir su simboliais.

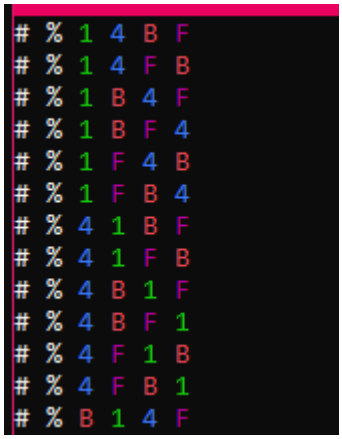
Penktas testas
Testo tikslas: Patikrinti programos veikimą didesniu kiekiu elementų.
Duoti duomenys: aibe = { ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, }
Rezultatai:



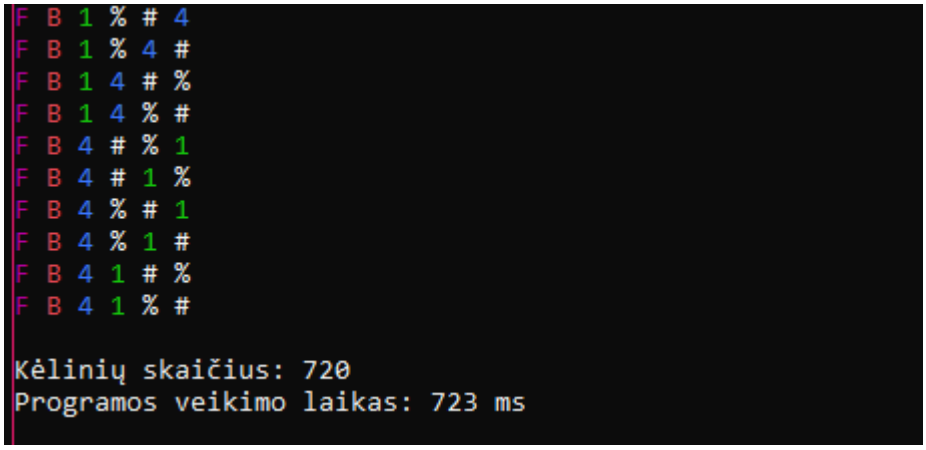
Pav. 13 Penktasis testas

Kaip matome iš kėlinių skaičiaus, tokios apimties ekrano iškarpos neišeis įkelt į ataskaitą neprikraunant papildomų 20 lapų. Tikėtasis kėlinių kiekis atitinka programos skaičiavimus ir matoma, kad jei nepadaryta papildomų case spalvos keitimams, jeigu yra daugiau elementų nei 7, visi sekantys elementai būna tiktai baltos spalvos. Tokį kiekį kėlinių sugeneravo per ~3 min. Jeigu būtų teisingai pritaikomas paralelizmas skaičiavimai galėtų vykti daug greičiau.

Šeštas testas
Testo tikslas: Patikrinti programos veikimą kai yra ir skaitinių reikšmių, ir raidžių, ir simbolių.
Duoti duomenys: aibe = { `'1'`, `'B'`, `'#'`, `'4'`, `'%'`, `'F'` }
Rezultatai:



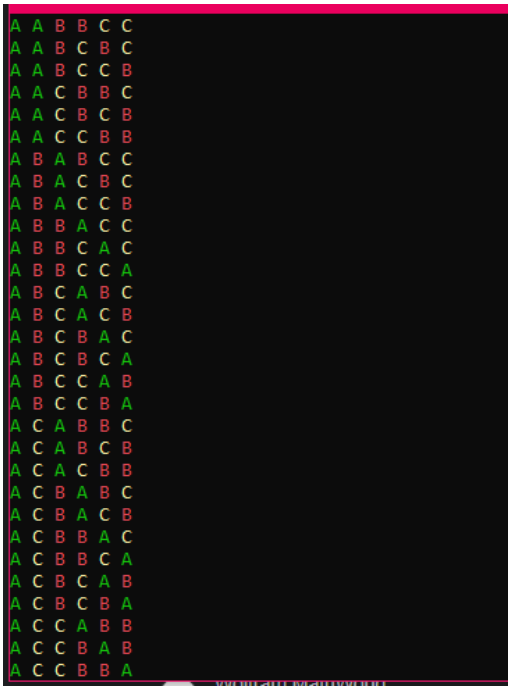
Pav. 14 Šešto testo pirma dalis



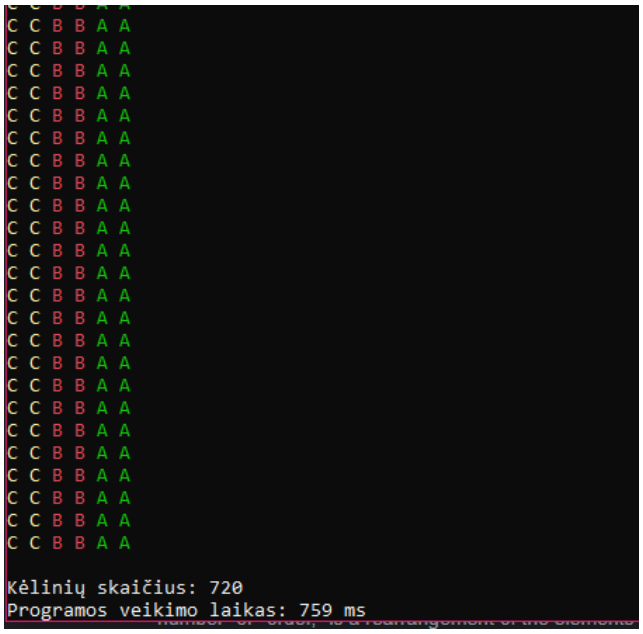
Pav. 15 Šešto testo antra dalis

Kaip matoma, pradžioje surikiuoja leksikografikai masyvą, todėl pirma simboliai, poto skaičiai, poto raidės, tačiau vistiek kėlinius generuoja ir sugeneruoja tiek kiek tikimasi kėlinių.

Septintas testas
Testo tikslas: Patikrinti programos veikimą kai pasikartojančių reikšmių aibėje.
Duoti duomenys: aibe = { `'A'`, `'A'`, `'B'`, `'B'`, `'C'`, `'C'` }
Rezultatai:



Pav. 16 Septintas testas 1 dalis



Pav. 17 Septintas testas 2 dalis

Išvados

Programa veikia sklandžiai ir labai greitai, nors ir išgavimo algoritmai nėra patys optimaliausi ir beveik kiekvienam metode reikia panaudoti bent po vieną ciklą. Greitaveika parodo šios programos efektyvumą ir tikslumą. Vienintelis tokio sprendimo minusas būtų tai, kad užduodant didesnius kiekius aibės elementų, skaičiavimai trunka ilgiau negu tikėtasi ir tai, kad yra saugomos papildomos reikšmės, kurių nebūtina keliskart saugoti, dėl to, kad jos dažniausiai nekinta. Taip pat ištestavus programos veikimą su pasikartojančiomis reikšmėmis, ciklai pasimeta ir toliau sukasi kol pasiekia tam tikrą kiekį kėlinių (šita dalis sutaisyta jau).

Literatūros sąrašas

https://moodle.ktu.edu/course/view.php?id=39

https://www.w3schools.com/cs/cs_switch.php

https://en.wikipedia.org/wiki/Permutation