МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Кафедра прикладної математики

**КУРСОВА РОБОТА**

із дисципліни «Програмування»

на тему

Генерація перестановок множини .

|  |  |
| --- | --- |
|  | |
|  |

Виконав:

студент I курсу групи КМ-73,

спеціальність 113 – прикладна

математика

САМОДРИГА О.А.

Керівник:

ГРОМОВА В.В.

Національна оцінка:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_

Оцінка ECTS: \_\_\_\_\_

Київ — 2018

Зміст

[ВСТУП 2](#_Toc515133312)

[1.ПОСТАНОВКА ЗАДАЧІ 3](#_Toc515133313)

[2.ВИБІР МЕТОДУ РОЗВ’ЯЗАННЯ 4](#_Toc515133314)

[3.АЛГОРИТМ 6](#_Toc515133315)

[4.ОПИС ПРОГРАМИ 8](#_Toc515133316)

[5.РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ 10](#_Toc515133317)

[ВИСНОВКИ 12](#_Toc515133318)

[ЛІТЕРАТУРА 13](#_Toc515133319)

[ДОДАТОК А.ТЕКСТ ПРОГРАМИ НА МОВІ С# 14](#_Toc515133320)

# ВСТУП

Метою даної курсової роботи є генерація перестановок множини.

Об’єкт дослідження – множини.

Предмет дослідження – знаходження усіх можливих варіацій перестановок заданих чисел.

# 1.ПОСТАНОВКА ЗАДАЧІ

У даній роботі потрібно згенерувати усі можливі комбінації множини.

Комбінація: {a1,a2…an}, де n-будь-яке натуральне число.

a1-ввідповідно перший член.

an-останній член.

Матиме n! Можливих варіацій

# 2.ВИБІР МЕТОДУ РОЗВ’ЯЗАННЯ

1. Код Грея.

Будемо будувати код Грея для довжини n = k. Припустимо, що нам відомий код Грея для перестановок довжиною k-1. Візьмемо першу перестановку з відомого нам коду. Вона має такий вигляд: {a1, a2, a3, ..., ak-1}

Спочатку запишемо число k в початок цієї перестановки, після чого будемо рухати його вправо елементарними транспозиція (підкреслені пари переставляються елементів).

* {K, a1, a2, a3, ..., ak-1}
* {A1, k, a2, a3, ..., ak-1}
* {A1, a2, k, a3, ..., ak-1}
* {A1, a2, a3, k, ... , ak-1}
* {A1, a2, a3, ..., k, ak-1 }
* {A1, a2, a3, ..., ak-1, k}

Отримаємо k різних перестановок, що відрізняються однією елементарною транспозицією. Маємо таку перестановку з коду Грея для перестановок довжини k-1 і припишемо в кінці число k. Ця перестановка відрізняється на одну елементарну транспозицію (останні елементи збігаються, а префікси довжини k-1 відрізняються на елементарну транспозицію). Нехай вона має такий вигляд:

{B1, b2, b3, ..., bk-1}

Елемент k записуємо в кінець і починаємо "рухати" його вліво:

* {B1, b2, b3, ..., bk-1, k}
* {B1, b2, b3, ..., k , bk-1}
* {B1, b2, b3, k, ..., bk-1}
* {B1, b2, k, b3, ..., bk-1}
* {B2, k, b1, b3, ..., bk-1}
* {K, b1, b2, b3, ..., bk-1}

Продовжуємо аналогічно. Для кожної перестановки дописуємо k в один кінець (по черзі), і за допомогою елементарних транспозиція рухаємо в інший кінець, при цьому додаючи кожну проміжну перестановку в список.

Таким чином отримуємо для кожної перестановки довжиною k-1 (всього (k-1)! Штук) по k нових перестановок, в сумі k⋅ (k-1)! = K! перестановок. Всі вони різні, так як для будь-яких двох перестановок з нового коду Грея елемент k варто на різних позиціях, а якщо k стоїть на одній і тій же позиції, то ці перестановки утворені від різних перестановок довжиною k-1. Так само всі сусідні перестановки відрізняються рівно в одній елементарної транспозиції. Разом, ми отримали список з k! різних перестановок довжиною k, причому сусідні відрізняються в одній елементарної транспозиції.

2.Алгоритм Джонсона-Троттера.

Порівняємо кожному елементу перестановки p [i] напрямок d [i]. Будемо вказувати напрямок за допомогою стрілок ← ( "вліво") або → ( "вправо"). Назвемо елемент рухомим, якщо у напрямку стрілки варто елемент менше за нього. Наприклад, для p = {1,3,2,4,5}, d = {←, →, ←, →, ←}, рухливими є елементи 3 і 5. На кожній ітерації алгоритму будемо шукати найбільший рухливий елемент і міняти місцями з елементом, який стоїть у напрямку стрілки. Після чого поміняємо напрямок стрілок на протилежне у всіх елементів великих поточного. Спочатку p = {1, ..., n}, d = {←, ..., ←}.

Приклад роботи алгоритму для n = 3

* p = {1,2,3} d = {←, ←, ←}
* p = {1,3,2} d = {←, ←, ←}
* p = {3,1,2} d = {←, ←, ←}
* p = {3,2,1} d = {→, ←, ←}
* p = {2,3,1} d = {←, →, ←}
* p = {2,1,3} d = {←, ←, →}

# 3.АЛГОРИТМ

Рішення задачі на генерацію комбінацій множин за Алгоритм Джонсона-Троттера поділяється на декілька етапів.

* На першому етапі заданий масив множини сортується за зростанням і кожному елементу цієї множини надається напрямок "вліво".
* На наступному кроці шукається рухомий елемент.
* Рухомий елемент міняється містами з сусіднім елементом відповідно до напрямку стрілки
* Шукаються усі елементи, що більші за значенням ніж рухомий. Якщо такий елемент знаходиться його напрямок змінюється на протилежний.
* Алгоритм виконується n! разів, відповідно до кількості можливих комбінацій.

Блок схема алгоритму Джонсона-Троттера

Введення множини n елементів і задання їх напряму

Сортування множини

Перебіг номеру комбінації від I до n!

Знаходження максимального рухомого елементу

Заміна напрямку елементів більших ніж максимальний рухомий

Заміна місця максимального рухомого елементу відповідно до напрямку руху

Вивід отриманої множини

# 4.ОПИС ПРОГРАМИ

1.Інструкція для кінцевого користувача.

При запуску програми відкривається програмне вікно, в якому користувачу надається можливість ввести розмірність множини. На головному меню знаходяться три кнопки «Запуск», «Теорія», «Вихід».

При введені розмірності і нажаті «Запуск» у випливаючому вікні надається можливість ввести відповідний елемент множини. При нажаті «відміни» введення припиняється-розмірність та значення елементів множини необхідно задавати спочатку. Після введення усіх елементів програма виводить отримані комбінації у вигляді таблиці.

При нажаті «Теорія» відкривається вікно з кнопкою «Повернутися» та підрозділами теорії на тему «Генерація перестановок множини», при нажаті на котрі відкривається відповідний теоретичний матеріал.

При нажаті «Вихід» програма закривається

2.Інформація для програміста.

Програма написана на мові C#.

Програма складається з проекту Windows Forms, в якому ініціалізація програми відбувається після натискання кнопок button1\_click, button2\_click.

В програмі знаходяться два вікна в котрі можна потрапити через один до одного.

Опис основних функцій:

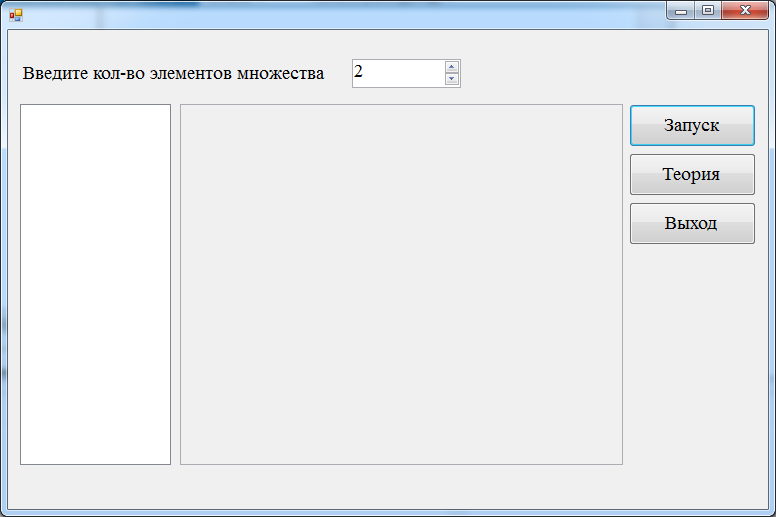
|  |  |  |
| --- | --- | --- |
| Назва функції | Опис формальних параметрів | Призначення(результат роботи) |
| button1\_Click() | object sender,  EventArgs e | Ініціалізація кнопки та дії, що вона виконує |
| button2\_Click() | object sender,  EventArgs e | Перехід до екрану з теорією. |
| button3\_Click() | object sender,  EventArgs e | Вихід. |
| Fractal() | int n | Підрахунок кількості комбінацій n елементів |

Опис основних даних:

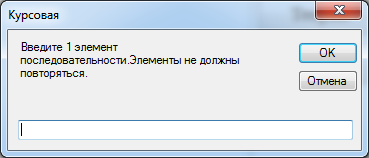
|  |  |  |
| --- | --- | --- |
| Назва змінної | Опис | Призначення |
| array[] | Масив даних типу double | Масив, що містить введену множину. |
| move[] | Масив даних типу int | Масив, що містить напрямок елементів множини. |
| movend | Елемент типу double | Рухомий елемент |
| indexl | Ціле число | Індекс максимального лівого рухомого елементу |
| left | Елемент типу double | Лівий максимальний рухомий елемент |
| right | Елемент типу double | Правий максимальний рухомий елемент |
| indexr | Цілі числа | Індекс правого рухомого елементу |
| int i, j,,l | Цілі числа | Коефіцієнти масивів |

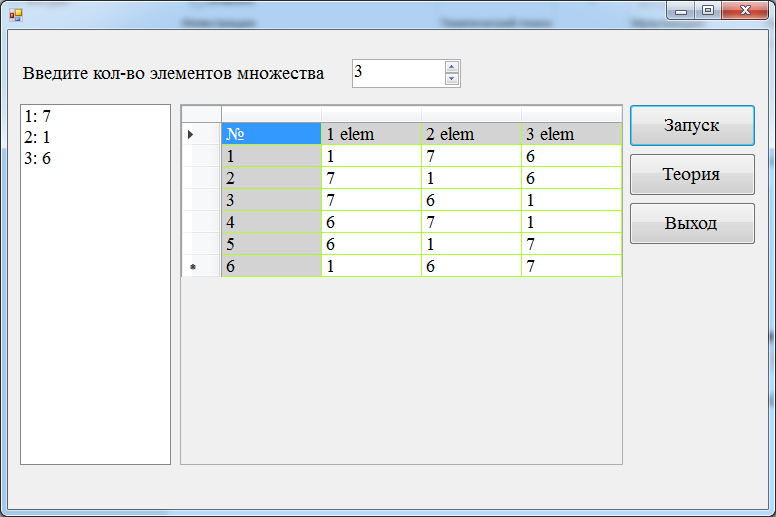
# 5.РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

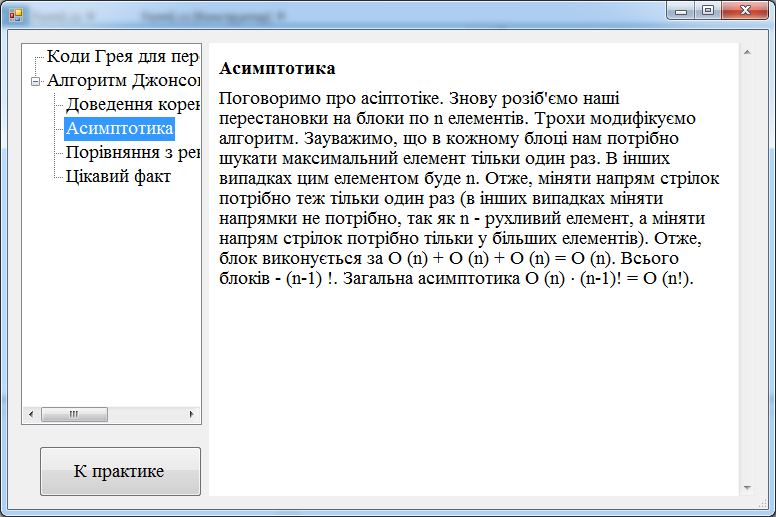
Початкове вікно.



Введення елементу



Теж вікно з введеною інформацією



Вікно з теоретичними відомостями

# ВИСНОВКИ

В ході виконання даної роботи, за для вирішення поставленої задачі був застосован алгоритм Джонсона-Троттера. Без посередньої перевагою даного алгоритму є можливість не зберігати всі попередні перестановки (з n-1 елемента), а тільки поточну. Отже, цей алгоритм споживає тільки O (n) пам'яті. Також, через нерекурсивність цей алгоритм працює швидше.

Нажаль один з головних недоліків цього алгоритму- попереднє сортування, що однозначно погіршує складність цього алгоритму.

За для покращення програми можна робити генерацію елементами, що повторюються.

# ЛІТЕРАТУРА

1. Романовский И.В. Дискретный Анализ - Санкт-Петербург, 2003. - стр. 39-41
2. Федоряева Т.И. Комбинаторные алгоритмы - Новосибирск, 2011. - стр. 36-46
3. Ананий Левитин, Алгоритмы. Введение в разработку и анализ - Москва. Санкт-Петербург. Киев, 2006. - стр. 226 - 229
4. https://msdn.microsoft.com

# ДОДАТОК А.ТЕКСТ ПРОГРАМИ НА МОВІ С#

Перша форма:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсовая

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

//Подсчет количества комбинаций

int Fractal(int n)

{

if (n == 1)

{

return 1;

}

else

{

return n \* Fractal(n - 1);

}

}

private void button1\_Click(object sender, EventArgs e)

{

double[]array;

int[]move;

int n, i, j, countl, countr, indexl = 0, indexr = 0, indem = 0, l, bufi;

double left = 0, right = 0, buf, bufb, moved = 0;

string elem;

n=(int)numericUpDown1.Value;

//Проверка на множество

while (n < 2)

{

MessageBox.Show("Не правильно введены данные.Кол-во элементов множесво должно быть больше 2.Иначе-не является множеством"); ;

return;

}

if (Convert.ToString(numericUpDown1.Value) == "-")

{

numericUpDown1.Value = 2;

}

array = new double[n];

move=new int[n];

listBox1.Items.Clear();

for (i = 0; i < n; i++ )

{

elem = Microsoft.VisualBasic.Interaction.InputBox("Введите " + (i + 1) + " элемент последовательности.Элементы не должны повторяться.");

if (elem.Length > 0)

{

while (true)

{

try

{

array[i] = Convert.ToDouble(elem);

break;

}

catch

{

MessageBox.Show("Не правильно введены данные.");

elem = Microsoft.VisualBasic.Interaction.InputBox("Введите " + (i + 1) + " элемент последовательности.Элементы не должны повторяться.");

}

}

}

else

return;

move[i] = 0;

listBox1.Items.Add(Convert.ToString(i + 1) + ": " + Convert.ToString(array[i]));

}

//Проверка на повторяемость

for (i = 0; i < n; i++)

{

for (j = 1; j < n; j++)

{

if (array[i] == array[j]&&i!=j)

{

MessageBox.Show("Элементы множества не должны повторятся.Введите элементы заново");

return;

}

}

}

//Предварительная сортировка

for (i = 0; i < n - 1; i++)

{

for (j = i + 1; j < n; j++)

{

if (array[i] > array[j])

{

bufb = array[i];

array[i] = array[j];

array[j] = bufb;

}

}

}

dataGridView1.ColumnCount = n+1;

dataGridView1.RowCount =Fractal(n)+1;

dataGridView1.Rows[0].Cells[0].Value = "№";

for(i=0;i<Fractal(n);i++)

{

countl = 0;

countr = 0;

//Поиск движущихся элементов

for (j = 0; j < n; j++)

{

//Движение влево

if (j != 0 && array[j] > array[j - 1] && move[j] == 0)

{

//Нахождение макс левого движущегося элемента

if (countl == 0)

{

left = array[j];

countl=1;

indexl = j;

}

else

if (array[j] > left)

{

left = array[j];

indexl = j;

}

}

else

//Движение вправо

if (j + 1 != n && array[j] > array[j + 1] && move[j] == 1)

{

//Нахождение макс правого движущегося элемента

if (countr == 0)

{

right = array[j];

countr=1;

indexr = j;

}

else

if (array[j] > right)

{

right = array[j];

indexr = j;

}

}

}

//Нахождение макс движущегося элеммента(сравнение левого и правого движущегося элемента)

if (left > right)

{

moved = left;

indem = indexl;

}

else

if (right > left)

{

moved = right;

indem = indexr;

}

//Замена направление движения

for (l = 0; l < n; l++)

{

if (array[l] > array[indem])

{

if (move[l] == 0)

{

move[l] = 1;

}

else

if (move[l] == 1)

{

move[l] = 0;

}

}

}

//Перемещение движущегося элемента

if (move[indem] == 0)

{

buf = array[indem - 1];

bufi = move[indem - 1];

array[indem - 1] = array[indem];

array[indem] = buf;

move[indem - 1] = move[indem];

move[indem] = bufi;

}

else

if (move[indem] == 1)

{

buf = array[indem + 1];

bufi = move[indem + 1];

array[indem + 1] = array[indem];

array[indem] = buf;

move[indem + 1] = move[indem];

move[indem] = bufi;

}

//Разметка

dataGridView1.Rows[i+1].Cells[0].Value = i+1;

dataGridView1.Rows[i + 1].Cells[0].Style.BackColor = System.Drawing.Color.LightGray;

//Вывод комбинаций

for (l = 0; l < n; l++)

{

dataGridView1.Rows[0].Cells[l + 1].Value = Convert.ToString(l+1)+" elem";

dataGridView1.Rows[0].Cells[l + 1].Style.BackColor = System.Drawing.Color.LightGray;

dataGridView1.Rows[i+1].Cells[l+1].Value = array[l];

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

frmTeor thirt\_form = new frmTeor();

thirt\_form.StartPosition = FormStartPosition.CenterScreen;

this.Hide();

thirt\_form.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

Друга форма:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсовая

{

public partial class frmTeor : Form

{

public frmTeor()

{

InitializeComponent();

}

private void treeView1\_AfterSelect(object sender, TreeViewEventArgs e)

{

string path = Application.StartupPath;

string st1 = treeView1.SelectedNode.Text;

if (st1 == "Коди Грея для перестановок")

webBrowser1.Navigate(path + "\\texts\\1.htm");

if (st1 == "Алгоритм Джонсона-Троттера")

webBrowser1.Navigate(path + "\\texts\\2.htm");

if (st1 == "Доведення коректності")

webBrowser1.Navigate(path + "\\texts\\3.htm");

if (st1 == "Асимптотика")

webBrowser1.Navigate(path + "\\texts\\4.htm");

if (st1 == "Порівняння з рекурсивним алгоритмом")

webBrowser1.Navigate(path + "\\texts\\5.htm");

if (st1 == "Цікавий факт")

webBrowser1.Navigate(path + "\\texts\\6.htm");

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 thirt\_form = new Form1();

thirt\_form.StartPosition = FormStartPosition.CenterScreen;

this.Hide();

thirt\_form.ShowDialog();

}

}

}